Malicious Users' Transactions: Tackling Insider Threat

Weihan Li, Brajendra Panda, and Qussai Yaseen

Department of Computer Science and Computer Engineering
University of Arkansas
Fayetteville, USA 72701
{wxl011,bpanda,qyaseen}@uark.edu

Abstract. This paper investigates the issues of malicious transactions by insiders in database systems. It establishes a number of rule sets to constrain the relationship between data items and transactions. A type of graph, called *Predictive Dependency Graph*, has been developed to determine data flow patterns among data items. This helps in foretelling which operation of a transaction has the ability to subsequently affect a sensitive data item. In addition, the paper proposes a mechanism to monitor suspicious insiders' activities and potential harm to the database. With the help of the Predictive Dependency Graphs, the presented model predicts and prevents potential damage caused by malicious transactions.

Keywords: Insider Threat, Transactions, Database, Security.

1 Introduction

Organizations face the continual possibility of outside (external) and inside (internal) attacks. In the current context that risks and chances of malicious attacks from intruders are limitless, it is very important that database systems that store critical data provide confidentiality, integrity and availability. Although a lot of work in security is related to the outsider threat problem, the growing reliance on technological infrastructures has made organizations increasingly vulnerable to threats from insiders. Identification of insider threats is a major challenge, because by definition [1], given a wide range of activities, insiders have been granted certain authority and trust. In addition, they have superior knowledge of the organization's inner control and security systems.

Although insiders can perform tasks as authorized users, they should not manipulate data items arbitrarily, but only in constrained ways that preserve or ensure the integrity of the data item. Since there are different sets of data items that are necessarily associated with a set of certain transactions permitted to manipulate them, even though the transactions could be from authorized users, any data item modifications should be constricted to the transactions assigned, and those transactions should be executed following the correct sequence [2].

This paper considers transactions that inject malicious attacks into the system and corrupt other transactions and data items. The threat mitigation technique presented in this paper attempts to remove the malicious effects of an insider and to provide mitigation service. The contribution of this paper is summarized as follows. First, Normal Activity Rules are established to constrain the relationship between data items and transactions that depend upon different security levels. Second, a graph called Predictive Dependency Graph is developed to investigate different patterns of data flows among the data items. Third, The Labeling Mechanism is adopted to flag any unverified data items in order to predict and stop any malicious data flow. Fourth, the idea of a Real Event Analyzer (REA) is used to check the transactions' effect actions matching to real world. The advantage of using the REA is that it permits monitoring of the suspicious users' activities and potential harm to the database.

The rest of the paper is organized as follows. Section 2 discusses some related work. Section 3 introduces the insider threat problem and describes the proposed model. Finally, section 4 provides the conclusions and mentions some future work.

2 Related Work

The increasing awareness of the insider threat problem has led to significant interest in techniques which can effectively detect malicious insiders' activities. Unfortunately, the techniques for mitigating insider threat are relatively immature and have not gained wide acceptance or use.

A significant amount of research on intrusion detection systems has been conducted for almost twenty years [3][4][5][6][7][8][9], however, most of them focus on the capability of identifying intrusions by outsiders. In recent years, several works have been done in the area of detecting insider threats [10][11]. However, they are not adequate for protecting the database from "denial of service" and they tend to generate low "false alarms". Yaseen and Panda [12][13] discussed how insiders can use dependencies to infer sensitive information and make malicious modifications in sensitive data items. In addition, they proposed models to detect and prevent those types of insider attacks.

Database Activity Monitoring [14][15] is an emerging technology that monitors and analyzes individuals or organizations' databases for potential leaks and compromises. Database Activity Monitoring operates independently of the database management system and does not rely on any form of native (DBMS-resident) auditing or native logs such as trace or transaction logs. Database Activity Monitors capture and record, at a minimum, all Structured Query Language (SQL) activity in real time or near real time, including database administrator activity across multiple database platforms, and can generate alerts on policy violations. There are several DAM products on the market. This paper adopts the concept of DAM to propose a new Real Event Analyzer with similar functionality. The architecture technique, management policy and other features are beyond the discussion of this work.

3 Insider Threat and Malicious Transactions

The goal of this paper is to design models that can detect and stop malicious transactions submitted to the DBMS by a malicious insider masquerading as a legitimate user. Since there can only be a finite set of tasks to be performed for inside users, this paper assumes that the “Normal Activity Rules” (discussed in the next section), which are predefined and established respectively by the application program and administrators /database experts, are stored separately for predicting any further damage. Before going further, let us introduce some definitions.

Definition 1. A *Critical Data Item (CDI)* is defined as a data item to which a change would induce significant effect. The CDIs therefore are usually the target of malicious attackers. Besides, any data items outside the CDI set are therefore termed as *Regular Data Item Set (RDI).*

Definition 2. A *Predefined Transaction* is an authorized transaction that is allowed to update data items to which it is assigned. A Predefined transaction is categorized into two different types:

1. **Predefined Trusted Transaction (PTT)** is the transaction that has authorization to access both critical data items and regular data items, and it is executed by trusted users such as DB Administrators or other high-level employees.

2. **Predefined Normal Transaction (PNT)** is the transaction that can access and update only regular data items, and it is executed by normal-level users who might occasionally spread the potential damage, or a malicious normal-level user who intends to carry out malicious activities by breaching the system security.

For each transaction that has access to a mount of data items, whether *Critical Data Item Set* and/or *Regular Data Item Set*, those data items in that transaction are categorized with relation pairs of *Read Set* and *Write Set*. *Data dependency* is defined as follow.

Definition 3. A write operation $w_i[x]$ of a transaction T_i is *dependent on* a read operation $r_i[y]$ of T_i if $w_i[x]$ is computed using the value obtained by $r_i[y]$. A data value v_1 is dependent on another data value v_2 if the write operation that wrote v_1 was dependent on a read operation on v_2. Notice that v_1 and v_2 may be two different versions of same data item.

The semantics information is insufficient; it is reasonable to presume that a write operation of a transaction depends on all read operations of the same transaction that precede the write operation in a particular pair in transaction *T*. For example, the Write Set containing write operation w[x] is dependent on a set of data items in the corresponding Read Set if *x*=f(Read Set), the values of data items in that Read Set are used in calculating the new value of *x* (Previous value of x is the value before current operation). There are three cases for the set of data items in Read Set, which are as follows.

Case1: Read Set = Ø. This means that no data item is used in calculating the new value of *x*. Such an operation is denoted as a fresh write. If w[*x*] is a fresh write and if the previous value of *x* is updated maliciously, the value of x will be refreshed after this write operation.

Case2: $x \notin$ Read Set. Then w[*x*] is called a blind write. If w[*x*] is a blind write and if the previous value of *x* is updated by a malicious transaction and none of the data items in Read Set are updated maliciously, then the value of *x* will be refreshed after this write operation. If any data items in Read Set are modified, the value of *x* will be modified.

Case3: $x \in$ Read Set. If the previous value of *x* is updated maliciously, then *x* remains contaminated. Otherwise, if any other item in Read Set is contaminated, *x* is thus affected.

These actually show the data dependency between the data items in related Read Set and the Write Set. The dependency of different relations of Read and Write sets, defined as Normal Activity States, are identified and denoted like this: $\{(R\_Set_1{:}W\_Set_1), (R\_Set_2{:}\ W\_Set_2), \ldots, (R\_Set_n{:}\ W\_Set_n)\}$. For instance, consider a transaction *T* with query statements in the given sequence as shown below:

UPDATE Table1 set x = a + b + x;
UPDATE Table1 set y = x + u;
*UPDATE Table1 set z = v *0.9;*

After analyzing the transaction, the Normal Activity States of transaction *T* is as following:

$T{:}\ \{(\{a, b, x\}{:}\ \{x\}), (\{x, u\}{:}\ \{y\}), (\{v\}{:}\ \{z\})\}$

The transaction *T* consists of three related pairs. Each pair represents the data dependency that once any data in the Read Set is modified, all or parts of the data items in the Write Set of the same pair are affected. Consider that once data item *x* in the Read Set is about to be modified by a previous transaction *T'*. Both the data items *y* and *x* in the related Write Sets of *T* are considered to be potentially affected when the transaction *T'* is in the process of being executed. This is because both data items *x* and y are partially dependent on data item *x*. So if any one or more data item sets in Read Set of *T* is/are updated by a previous transaction *T'*, the value of at least one data item in the Write Set will be presumably influenced. This kind of situation is considered as potential damage to the critical data items in the database system.

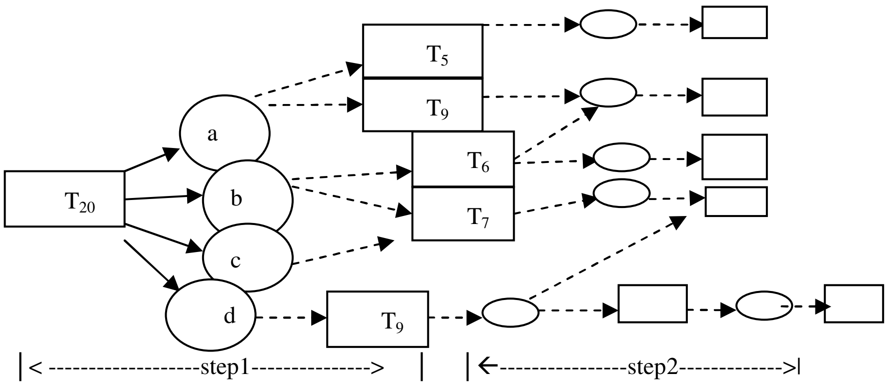
Definition 4. The *Predictive Dependency Graph PDG* is a graph adopted to demonstrate the prediction of a particular transaction's impact or pathway in the database.

Figure 1 shows an example of a PDG. A rectangular node in the graph contains information such as transaction's ID and its operation sequence; an oval node in the graph denotes the data items in the Write Set of the transaction from which the arrows come. A firm edge from transaction to data item represents that transaction updates the data item to which the arrow points. The predictable transactions which might probably be affected by data items being updated are denoted by a dash edge from data item to transaction. It is necessary to mention that the subscript for each transaction only means the transaction id in the Normal Activity States not the time sequence of transactions. As seen in Figure 1, the procedure starts from the rectangular node of transaction, and ends at rectangular nodes which represent all the affected transactions presumably determined by data dependencies.

Definition 5. *Average Time, $time_{avg}$,* is defined as the average time needed for the execution of a particular transaction, which is measured using the following formula:

$$time_{avg} = \frac{\sum_{i=1}^{N}(t_{i-exe} - t_{i-sch})}{N} \tag{1}$$

Where $t_{i\text{-}exe}$ denotes the time when it was executed and $t_{i\text{-}sch}$ represents the time when it was scheduled, N is the number of total transactions executed.



T5
T9
a
T6
T7
T20
b
c
d
T9
|< ------------------step1--------------> |
|←---------------step2------------>|

Fig. 1. Predictive Dependency Graph

Definition 6. *Assurance time t_A* is preset by the database experts and is defined to be used with the information of the average time of execution of particular transaction, which is measured using the following formula.

$$t_A = \frac{\sum_{i=1}^{N}(t_{i-exe} - t_{i-sch})}{N} * Avg_{Ts} * n_{STEP} \tag{2}$$

Where N is the number of steps used for generating the PDG graph and Avg_{Ts} represents the average number of potential affected transactions in each step.

Notice that in the proposed PDG graph, the one complete procedure of a step starts with the rectangular node of transaction and passes through the circular nodes of data items in its Write Set. Hence, in order to predict any further potential damage from an incoming transaction, the Predictive Dependency Graph is produced based on the combined information of real execution time $time_{avg}$ and threshold number n. Hence, for each incoming transaction T, the possibility of potential damage (as mentioned on the previous page) is predicted and measured within the time period of t_A:

Definition 7. *Normal Activity Rules* is a predefined profile that contains a group of relation rules, each relation is of the form:

Rule, Transaction, Read Set, Write Set, Potential Damage, Assessment, Action,

where a transaction is categorized by different user's security levels, and Read and Write Sets are the constrained accessible data item sets that correspond to that security level.

Based on definition 1 and 4, there is a number of Preformed Transaction and Read/Write Set rules in the Normal Activity Rules Table in Figure 2. Predefined Trusted Transactions and Predefined Normal Transactions are denoted by PTTs and PNTs respectively. RDI and CDI represent Regular Data Item and Critical Data Item. We should mention here that *Normal Activity Rules* are constrained by application program or database administrator and is based on security issues and users' privileges.

For the Predefined Normal Transaction, it only has authorization over access to and update of the RDI Set. Since PNT is needed to be judged whether it is from an innocent insider or a masqueraded insider, PNT should be examined before scheduling it to commit. In the real world, however, thousands of PNT from different users can be executed per second in the database application program. In order to avoid slowing down the system, transactions are scheduled to be executed and examined simultaneously. Meanwhile, since the Predefined Trusted Transactions are considered from trusted user by default, any access or update merely within the range of CDI Set can be executed without any supervising action. However, any PTT update, involving both the Critical Data Item Set and at least one other data item from Regular Data Item Set, is needed to be checked carefully and thoroughly. It is the upmost

Rules	Transaction	Read Set	Write Set	Potential Damage	Action
1	PNT	RDI	RDI	indirectly access CDI	verify R Set, then predict damage
2	PNT	RDI	CDI	directly access CDI	not comply to rules/states, reject
3	PNT	RDI /CDI	CDI,RDI	directly access CDI	not comply to rules/states, reject
4	PNT	RDI, CDI	CDI/ RDI	directly access CDI	not comply to rules/states, reject
5	PTT	RDI	RDI	indirectly access CDI	verify R Set, then predict damage
6	PTT	RDI	CDI	directly access CDI	verify R Set
7	PTT	CDI	CDI,RDI	No	allowed
8	PTT	CDI	CDI/RDI	No	allowed
9	PTT	RDI,CDI	RDI	indirectly access CDI	verify R Set, then predict damage
10	PTT	RDI,CDI	CDI	directly access CDI	verify R Set

Fig. 2. Normal Activity Rules Table

requirement since before allowing any update, all data items to be read are verified to make sure that there are no damaged data from previous transactions. In addition, no more potential damage in future can result from the execution.

3.1 The Proposed Model

The goal of this model is to develop a Predictive Insider Threat Detection Mechanism that can be implemented within the database server and is capable of detecting any anomalous users' requests to a DBMS. Suppose that the Normal Activities States are as shown in Figure 3 (for each T_x, x is the transaction id, not the sequence of events). For instance, PNT Transaction T_5 has 2 pairs of Read Sets and Write Sets. Once a data item *m* is updated in the previous PNT Transaction T', data items *l* and *a* are all considered to be potentially affected. On the other hand, once data item *l* in pair 2 is updated beforehand, only data item *l* in the related Write Set will be presumed to be the affected one. In PTT Transaction T_{200}, the data item *z* in bold is defined as a critical data item, and this PTT Transaction T_{200} follows the Rule 10 in the Table in Figure 2, all the other Transactions follow the Rule 1.

In Figure 4, suppose the incoming PNT Transaction T is the one that is to be scheduled to update data items *a*, *b*, *d*, and data item *z* is a critical data item. There are N steps after any updates on them. Similarly there are also countless transactions and data items that need to be checked. However, in this model, only 2 steps are analyzed for that particular Transaction T which might change data item *a*, *b*, *d*. From this graph, once the PNT Transaction T on data items *a*, *b*, and *d* is executed, there is a possibility that the critical data item *z* might be manipulated by the effect of previous events through T_7 to T_{200}.

T_2: {({a}:{m,o})}
T_3: {({a,f}:{c,f})}
T_4: {({a,b}:{c,d})}
T_5:{({m,k,l}:{l}), ({m,k,l}:{l})}
T_7: {({m,j}:{t})}
T_8: {({p,o}:{s,n})}
T_{12}: {({b,d}:{p,q})}
T_{15}: {({a,d}:{d,q})}
T_{20}: {({q,l}:{g,l}),({q,l}:{m})}
……
……
T_{200}: {({t,**z**}:{**z**})}

Fig. 3. Example of Normal Activities States

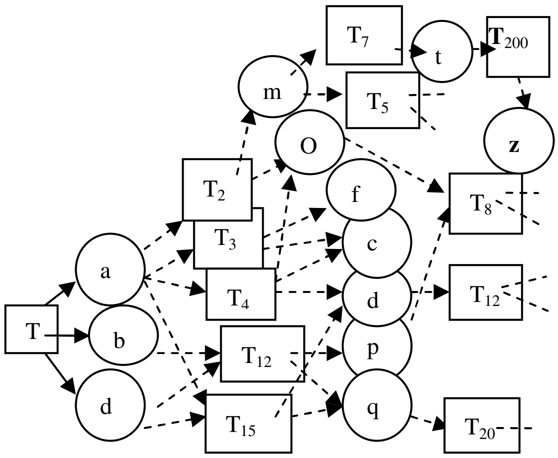


Fig. 4. Predictive Dependency Graph Model

However, since only the first 2 steps are checked in this graph, the PNT Transaction is considered to be safe in the time range we preset. If we further analyze it to 3 steps, T_{200} might be detected since it has indirect potential damage on critical data item *z*. In light of the situation of Transaction T_3 and T_{15}, for which the same data item appears in both the Read Set and Write Set, in order to keep the limited account of the considered affected transactions, the predictive dependency between the data item and the transaction which updates the same data item is not considered in the model.

3.2 The Labeling Mechanism

The data item is labeled and stored into the unverified data items set in two situations. The first situation is that the PNT Transaction whose Read Set is all verified and is still considered to be a threat. This is because it gets too close to CDI Set and all the following transactions needs to be alerted by labeling process before any direct damage to CDI. Accordingly, all of the data items in the Write Set of that PNT Transaction are initially labeled as unverified "dirty" data items and stored once it gets
executed. The second situation is when a damage spread might happen. In order to keep the both efficiency and availability of database and store the further impact of the to-be-verified data item, some of the data items in the Write Set whose corresponding Read Set are not verified are labeled as unverified "dirty" data items.

3.3 The Damage Predictor

The Damage Predictor is used to predict any potential impact on data items in the Critical Data Item Set within a certain time of period t_A. For all the transactions, PNTs or PTTs, which are submitted to the database application program, before scheduling them to be executed, the prediction of affected transactions and its data items can be made by using the Damage Predictor. First, the data items in the Read Set of that particular Transaction are checked and verified, and then, data dependency relation among other transactions in the Normal Activity States list is found. Second, a PDG graph can be produced with all the possible paths of the dataflow initialed from the incoming transaction. With enough information and in cooperation with the threshold time t_A, if the PDG graph of incoming transactions show indirect access to the CDI, or as also termed "potential damage", the transaction might be considered to be from a user with malicious motivation, and thus, a more strict and harsh action will respond to it. For the PTT transaction that has more than one related pair, there might be more complicated situations that may hinder the Damage Predictor process. For example, since PTT has privilege over the CDI Set, the Read Set of PTT can either be the combination of CDI and RDI, or only contain the RDI. In this case, in order to simplify the process, the graph is checked using the same strategy instead of checking the pairs separately.

3.4 Real Events Analyzer

The Real Event Analyzer is used separately during the transaction activity-monitor process of the proposed Insider Detection System. In order to detect an attack, the

Real Event Analyzer will examine the real events that correspond to each query and judge whether the transaction is from a legitimate user based on the query information. If the presumed malicious transaction turns out to be innocent, all the data items in the Write Set of that transaction will be cleared and all the labels will be removed from the unverified data item set. Another function for the Real Events Analyzer is its ability to calculate the recovery time for the cleaning process of each data item *x*. So once potential damage is found in the PDG within the time period t_A, the average recovery time t_R should be checked:

$$t_R = \sum_{i=1}^{n}(t_{i-max} + t_{i-min}) * Avg_{OP} /2 \qquad (3)$$

where $t_{i\text{-max}}$ and $t_{i\text{-min}}$ are the maximum and minimum recovery time of labeled data items for a single step respectively, and Avg_{OP} is the average number of operations.

The recovery time t_R is calculated as average time length used for recovering data items during each step and t. If the recovery time t_R for the data item is beyond the assurance time t_A, which means that the unverified data items usually cannot be cleaned by the time it has spread the damage to the CDI indirectly, the transaction will be rejected. The advantage of the Real Event Analyzer lies in the fact that the examination process is arranged after the execution of the transaction, thus the speed of the DBMS system can be guaranteed. For those PTT Transactions whose Read_Set are not cleared, the transactions will be on hold unless all the data items are safe to use.

3.5 System Architecture

Figure 5 shows the architecture of the proposed approach. It has two main processing parts: the Damage Predictor and the Real Event Analyzer. Transactions are first compared with Normal Activities States (NAS) and sorted into PNT and PTT Transactions for further detection. The Damage Predictor processes transactions and generates Predictive Dependency Graph to predict potential damage and to report alerts to the Real Event Analyzer and the Scheduler. The Real Event Analyzer (REA) examines the executed transaction and verifies the data items, and then reports back to the scheduler.

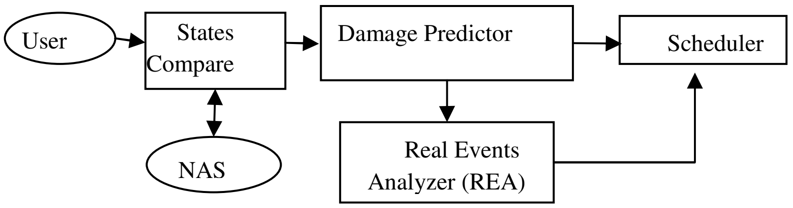


Fig. 5. System Architecture

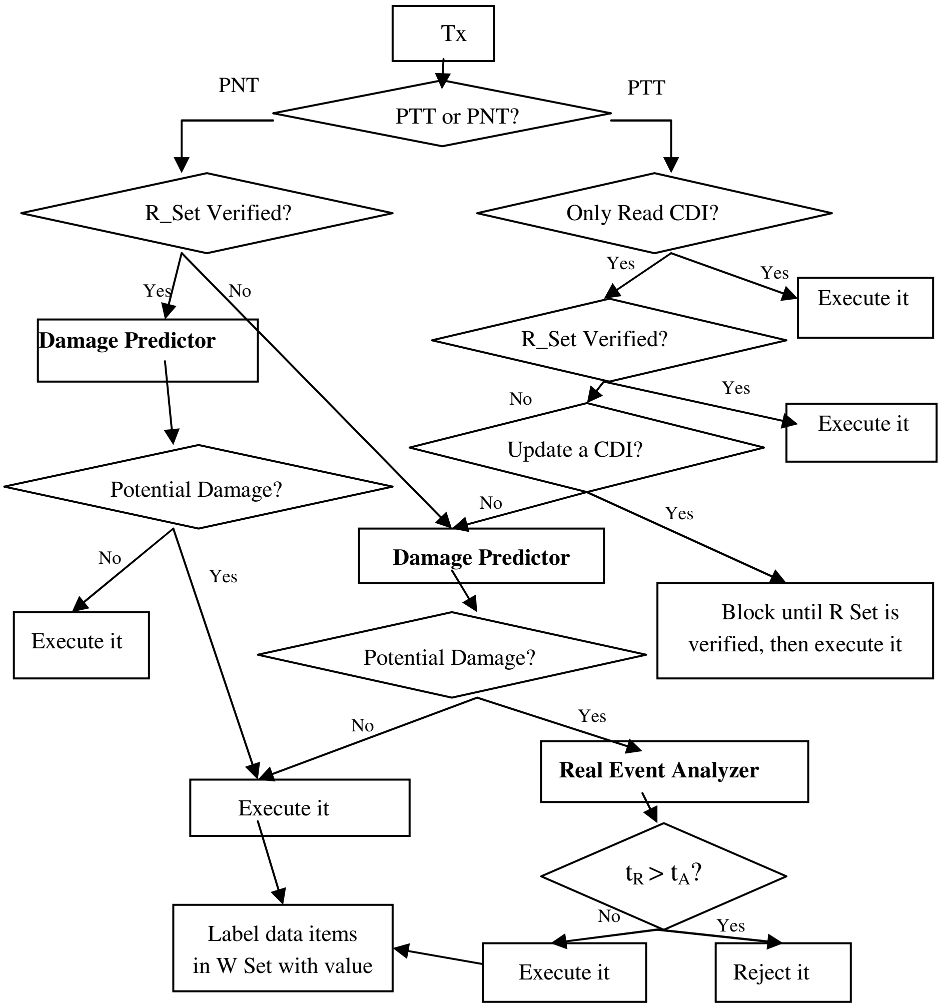


Fig. 6. Dataflow Chart of the Model

Figure 6 shows the flow chart incorporated with Damage Predictor and Real Events Analyzer. The Transaction Scheduler schedules PNT or PTT as to be executed or not. The ID mechanism is adopted to identify if the transaction is from a normal user or a trusted high level user. If the transaction is PNT, all Read Sets in R/W Set pairs are checked to see if there is any "bad" data item x which is damaged in previous executed transaction. If the system is not sure whether the transaction is from an innocent user, the labeling mechanism is initialized after the execution of T_x, so that the data items in the Write Sets will be cleaned during the next verification process. The time t_R and t_A are generated in the Damage Predictor and the Real Events Analyzer respectively. For instance, if threshold t_A is set to 15 seconds when the PDG is generated, and recovery time t_R is 12 seconds by consulting the individual recovery time for data items, such transaction will be rejected since once data flow reaches the Critical Data Item Set, the verification is still in process and the value of the Critical Data Item will be compromised. In the case of the transaction from trusted user (PTT), if the transaction only reads data items from CDI (like the rules 7 and 8 in Figure 2), it will be executed without any check. If the transaction has direct access to CDI (like the rules 6 and 10 in Figure 2), the transaction should wait until all the labeled data items in Read Sets to be verified by REA. The system must be aware of the data flows between trusted users and suspicious users. Thus, the identity of the transactions and data items that T_x depends upon needs to be checked beforehand. Moreover, since Predefined Normal Transactions play an important role numerically in the database system, and any "block" action might significantly reduce the efficiency, labeling mechanism and the Real Events Analyzer are adopted here, instead of simply discarding the normal transaction. Meanwhile, on the other side of the Presumed Trusted Transaction, identification is also a mandatory step in the proposed approach. In addition, since critical data items always have extreme influence on regular data items and it is assumed that all the PTTs are from the trusted users that take more responsibility. CDIs should avoid any damage flow from RDI before any PTT is executed.

4 Conclusions

This paper has addressed the problem of malicious transactions by insiders in database systems. It has established a number of rule sets to constrain the relationship between data items and transactions, which are called Normal Activity Rules. Moreover, it has used a graph model, called Predictive Dependency Graph (PDG), to determine data flow patterns among data items. The PDG has been used to look for the data items that might be affected once an update to a particular set of data items occurs. That is, PDG has been used to detect any indirect damage to particular data items. The Labeling Mechanism and Real Event Analyzer have been used to detect and prevent malicious transactions. The Labeling Mechanism has been employed to flag any unverified data items in order to predict and stop any malicious data flow. The Real Event Analyzer has been used to permit monitoring of suspicious users' activities and potential harm to databases. As a future work, we plan to formally define and develop the Real Event Analyzer and analyze its performance.

Acknowledgments. This work has been supported in part by US AFOSR under grant FA 9550-08-1-0255. We are thankful to Dr. Robert. L. Herklotz for his support, which made this work possible.

References

1. Mills, R.F., Peterson, G.L., Grimaila, M.R.: Insider Threat Prevention, Detection and Mitigation. In: Knapp, K.J. (ed.) Cyber Security and Global Information Assurance: Threat Analysis and Response Solution. U.S. Air Force Academy, Colorado, USA (2009)

2. Clark, D., Wilson, D.: A comparison of Commercial and Military Computer Security Policies. In: IEEE Symposium on Security & Privacy (1987)
3. Chung, C.Y., Gertz, M., Levitt, K.: Demids: A misuse detection system for database systems. In: 14th IFIP WG11.3 Working Conference on Database and Application Security (2000)
4. Lee, S.Y., Low, W.L., Wong, P.Y.: Learning Fingerprints for a Database Intrusion Detection System. In: 7th European Symposium on Research in Computer Security (2002)
5. Kamra, A., Bertino, E., Terzi, E.: Detecting anomalous access patterns in relational databases. The International Journal on Very Large Data Bases 17(5), 1063–1077 (2008)
6. Hu, Y., Panda, B.: Design and Analysis of Techniques for Detection of Malicious Activities in Database System. Journal of Network and System Management 13(3), 111–125 (2005)
7. Lee, W., Stolfo, S.J.: A framework for constructing features and models for intrusion detection system. ACM Transactions on Information and System Security 3(4), 227–261 (2000)
8. Meng, Y., Liu, P., Zang, W.: Multi-Version Attack Recovery for Workflow Systems. In: Proceedings of the 19th Annual Computer Security Applications Conference (2003)
9. Srivastava, A., Surai, S., Majumbar, A.K.: Weighted Intra Transaction Rules Mining for Database Intrusion Detection. In: Proceedings of the Pacific-Asia Knowledge Discovery and Data Mining (2006)
10. Ray, I., Poolsappasit, N.: Using Attack Trees to Identify Malicious Attacks from Authorized Insiders. In: Proceedings of the 10th European Symposium on Research in Computer Security (2005)
11. Martinez-Moyano, I., Rich, E., Conrad, S., Anderson, D.F., Stewart, T.R.: A Behavioral Theory of Insider-Threat Risk: A System Dynamic Approach. ACM Transactions on Modeling and Computer Simulation 18(2) (2008)
12. Yaseen, Q., Panda, B.: Predicting and Preventing Insider Threat in Relational Database Systems. In: Samarati, P., Tunstall, M., Posegga, J., Markantonakis, K., Sauveron, D. (eds.) WISTP 2010. LNCS, vol. 6033, pp. 368–383. Springer, Heidelberg (2010)
13. Yaseen, Q., Panda, B.: Malicious Modification attacks by Insiders in Relational Databases: Prediction and Prevention. In: 2nd IEEE International Conference on Information Privacy, Security, Risk and Trust (2010)
14. Newman, A.: Database Activity Monitoring: Intrusion Detection & Security Auditing. DAM Whitepaper, `http://www.appsecinc.com/presentations/DAM_wp82305.pdf`
15. Mogull, R.: Understanding and Selecting a Database Activity Monitoring Solution, `http://www.securosis.com/reports/DAM-Whitepaper-final.pdf`