MINIMIZE THE FALSE POSITIVE RATE IN A DATABASE INTRUSION DETECTION SYSTEM

A. Rezk[1], H. Ali[2], M. El-Mikkawy[3] and S. Barakat[1]

[1]Faculty of Computers and Information Sciences, Mansoura University, Egypt
amira_rezk@mans.edu.eg, sheiib@mans.edu.eg
[2]Faculty of Engineering, Mansoura University, Egypt
h_arafat_ali@mans.edu.eg
[3]Faculty of Science, Mansoura University, Egypt
mikkawy@mans.edu.eg

ABSTRACT

A database intrusion detection system (DIDS) is used to detect potential violations in database security. DIDS follows other traditional database security mechanisms and network security mechanisms such as firewall and network intrusion detection. Therefore, it faces the intrusion from internal users or the intrusion that can be passed through other security layers. This means that the number of intrusion event is rare compared to the number of the normal event. Therefore, it is not efficient to raise a large number of false alarms to achieve a high detection rate. This paper introduces an enhancement for the data dependency model and integrates it with access control to override the high rate of false alarm and increase the detection rate.

KEYWORDS

Database security, intrusion detection, association roles, data dependency

1. INTRODUCTION

In recent years, researchers have become interested in database intrusion detection system as an additional layer of defense behind firewalls, network- and host-based intrusion detection, and other traditional security mechanisms. Designing an intrusion detection system (IDS) to detect attacks directed at the database management system (DBMS) is based on the fact that actions deemed malicious for a DBMS are not necessarily malicious for the underlying operating system or the network. Thus designing an IDS for the latter may not be effective against database attacks [1]. In addition, host- or network-based IDSs are mainly focused on detecting external intrusions as opposed to internal intruders, while legitimate users who abuse their privileges are the primary security threat in database systems. Therefore, SQL injection [2] and other SQL-based attack targeted at databases cannot be effectively detected by network- or host-based IDSs. The distinctive characteristics of DBMSs, together with their widespread use and the invaluable data they hold, make it vital to detect any intrusion attempts made at the database level [3].

Database intrusion refers to an unauthorized access and misuse of database systems. DIDSs identify suspicious, abnormal or downright malicious accesses to the database system from both of internal and external users. These systems aim to detect intrusions as early as possible, so that any damage caused by the intrusions is minimized. Unfortunately, malicious transactions can seriously corrupt a database through a vulnerability denoted as damage spreading [4].

The purpose of IDS is to classify the input correctly as normal or intrusive. That is, the IDS output should faithfully reflect the “truth" about the input (i.e., whether an intrusion occurs or

DOI : 10.5121/ijcsit.2011.3503

not) [5]. Generally, an IDS works in two phases: the training phase and the detection phase. In the training phase, the data that describes the normal pattern (in the case of anomaly-based IDS) or the attack pattern (in the case of misuse-based IDS) are collected, pre-processed, and used to create a profile which is stored in a data repository. In the detection phase, the current event is compared with the profile in the data repository to detect if it is normal or malicious, if it is a malicious, an alarm is raised and a response is taken according to the response strategy [6, 7].

In any intrusion detection system, maintaining a profile, which represented accurate and consistent subject behavior, is the main goal. While the main challenge is generating an optimal set of roles that maximize the detection rate and minimize the false rate.

There are different scenarios for user's access to database. The user may connect to the database through application and act a specific operation, which allowable by the application. In this scenario, the number of submitted queries is limited and its syntax is known. However, many systems, which have a huge number of users who access the application, checks the authentication and the authorization of the user to access the application and the application in this manner is a database user and the DBMS must check his authorization before any transaction execution. Neglecting this action may violate the database security by SQL-injection attacks.

The second scenario allows the user to submit an adhoc queries to database. The DBMS perform authentication and authorization before any transaction execution; Access control checks if the transaction attempts to access data to which user has authority to access [8].

Generally, whatever the natural of the database's users (person or application), the database administrator must specify their authorization carefully. This is an important step toward a secure system, which eliminate the Privilege Abuse and Privilege Elevation attacks [9]. It is important to note that, the intruder can submit malicious transactions to the database only by masquerading as a normal user. However, Access control checks authorization without any intelligence. Therefore, it is important to combine it with IDS, which defines the normal behavior for each user [10].

In this paper, the normal behavior of each user will be described using the enhanced data dependency model, which integrated with access control.

The remainder of this paper will be organized as follows. Section 2, the related work. Section 3 introduces the proposed model. Section 4 provides the experimental analysis, and Section 5 conclusion and future work.

2. RELATED WORK

In the last few years, the researchers began to interest in DIDS as an additional layer of defense. The researchers follow different directions to build their systems. One of these directions is analyzing the query expressions, which has many approaches. These different approaches include syntax-based, semantic based and data dependency.

Syntax-based: analyzing the SQL-expression syntax of queries. In [11], SQL statements are summarized as regular expressions, which are then considered "fingerprints" for legitimate transactions. [12] introduce an anomaly based intrusion detection model based on the types of queries an authorized user fires and classify them as malicious or legitimate based on the analysis of the query parameters. In [13, 14, 15], database transactions are represented by directed graphs describing execution paths (select, insert, delete etc.) and these are used for malicious data access detection. In [16] a grammar based approach is used to represent SQL

queries. They use tree-kernels for analyzing SQL statement, which brings together the results of natural language processing with a highly structured query language. This approach made assumptions such as restricting the number of distinct queries possible. However, syntactically similar queries may produce vastly different results, leading the syntax-based engine to generate false negatives.

Semantic-based: interested with what the user is trying to access – the result of the query itself – rather than how he expresses it. [17] Introduces a data centric approach in which a user profile is built on what he accesses (i.e. the semantic of the query). The feature vector is extracted from the result set of a query and used to build the user profile.

Data-dependency: This approach mines dependencies among attributes in a database. The transactions that do not follow these dependencies are marked as malicious transactions [18, 19, 20]. In [18], researchers pay attention to the sequences that include write operation and delete the sequences that contain read only operations because a write operation is more critical for the database. In [19] the work in [18] is enhanced by taking the sensitivity of the attributes into consideration in the form of weights. Sensitivity of an attribute signifies how important the attribute is, for tracking against malicious modifications. Researches in [20] also modify the work in [18] by extending the concept of malicious transactions from those that corrupt data to those that either read data or write data or do both without permission. In [21] Hu & Panda extended their work in [18] to include the inter-transaction data dependencies in order to detect attacks carried out by a set of seemingly harmless database transaction. [22] also uses inter-transactional as well as intra-transactional features for intrusion detection

SQL injection and other SQL-based attack targeted at databases become a series threats which violate the database security. Therefore, the researcher interest in the last few years to establish an intrusion detection system that detect this type of attacks [23, 24]

The data-dependency models in [18, 19, 20], find the association roles between the data items without consideration as to who accessed this data. This may generate a role that conflict with the access control mechanism for some users, and this will cause false positives and violate the availability of the database for these users. In [10] we proposed a model that enhances the data-dependency model via integrating it with access control and finds association rules of data dependency for each individual user.

In this paper, we will enhance the data dependency model that was presented in [10], in trial to minimize the false positive rate and increase the detection rate.

3. THE PROPOSED MODEL

It is important to highlight that, the DBMS perform authentication and authorization before any transaction execution; this means that, the intruder can submit malicious transactions to the database only by masquerading as a normal user. Access control checks if the transaction attempts to access data that user has authority to access without any intelligence. On the other hand, the IDS check if the behavior of a user while accessing the data is normal or abnormal. However, current database intrusion detection systems work in isolation from access control. The lack of coordination and inter-operation between these two components prevents detecting and responding to ongoing attacks in real time, in addition, it causes high false alarm rate.

The proposed model integrates the IDS with the DBMS. It uses the access table to determine the user authorization and specify which items he has privilege to read and/or modify. The model mines the dependency among user's data items from his transaction log and generates specific roles, which determine the context in which the user access the data items.

Figure 1 presents the cooperation between the intrusion detection system and access control. When user submit query to execute, the DBMS checks the user identification and his authorization to execute the query using the access table, if he has authority to access the data item in query, the IDS check the context in which the user access the data item to determine if he follow his normal behavior or not. If there is any deviation from his normal behavior, an alarm is raised, else the query is executed.

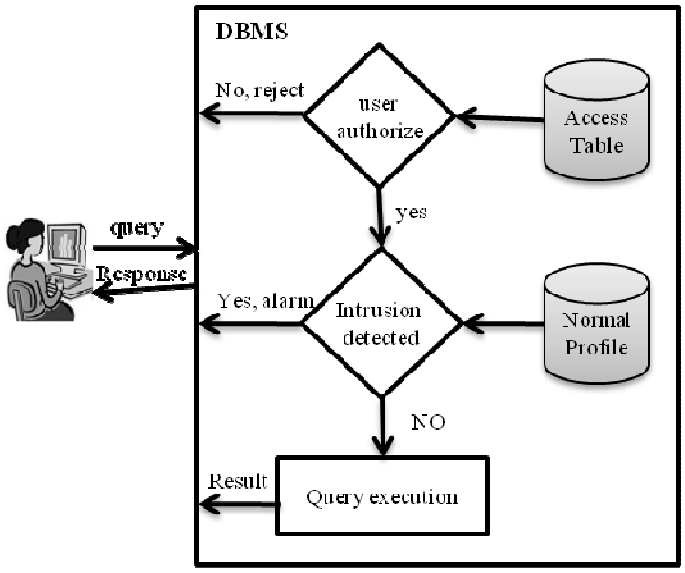

Figure 1. The cooperation between the IDS and access control.

3.1 The Data Dependency Model

In [10], we introduced a data-dependency model and combined it with access control. Data-dependency model, which depends on the association role algorithms, uses the concept of support and confidence to generate the dependency roles. The process of extracting dependency rules is done in three phases: (1) mining the frequent sequential patterns, (2) generating the read, pre-write, and post-write sequence sets, and (3) extracting the dependency rules among data items [18]. First step finds all frequent sequential patterns from a given data set with a specific support threshold. Then the read, pre-write, and post-write sequences, which are suitable for generating the dependency rules are extracted.

Definition 1: The read sequence RS(X) of a data item x is the sequence with the format $< r(d_1), r(d_2) \ldots r(d_n), O(X)>$ which represents that the transaction needs to read all data items d_1, d_2, ..., d_n in that order before the transaction reads or writes X.

Definition 2: The pre-write sequence pre-WS(X) of a data item x is the sequence with the format $< w(d_1), w(d_2), \ldots, w(d_n), O(x) >$, which represents that the transaction needs to write all data items $d_1, d_2, \ldots, d_n$ in that order before the transaction reads or writes X.

Definition 3: The post-write sequence pre-WS(X) of a data item x is the sequence with the format $< O(X), w(d_1), w(d_2), \ldots, w(d_n)>$, which represents that the transaction needs to write all data items $d_1, d_2, \ldots, d_n$ in that order after the transaction reads or writes X.

Finally Data dependency rules (read rules, pre-write and post-write rules) are generated. If the confidence of the rule is larger than the minimum confidence, then it is added in the answer set. For more detail, refer [10].

For example, Table 1 shows a typical log file of transactions. We use the same example, which used in [10, 18, 20] to introduce clear comparison. Table 2 introduce Data Dependency Rules Generated [10].

Table 1. Example Transactions log

Trans. ID	User ID	Transaction Operations
1	U1	r(7), r(1), r(6), w(5), r(1), w(4)
2	U1	r(1), r(5), w(1), r(4), r(5), w(4)
3	U1	r(7), r(6), r(2), w(4), r(7), r(3), w(6), r(1), r(6), w(2), r(3), r(5), r(2), w(5)
4	U1	r(2), w(2), r(4), r(7), w(3), r(6), w(5), r(1), w(4)
5	U1	r(6), r(1), w(3), r(1), w(6), r(2), r(7), r(4), w(2)
6	U2	r(5), r(3), r(6), w(7)
7	U2	r(2), r(5), w(6)
8	U2	r(4), w(6)
9	U2	r(6), r(5), w(5), r(3), r(4), w(4), r(3), w(7)
10	U2	r(5), r(6), w(5), r(5), w(4)

Table 2. Data Dependency Rules Generated for Each User

User1		User2	
Dependency rules	**Confidence**	**Dependency rules**	**Confidence**
Read rules			
r(1) → r(6)	80%	r(3) → r(5)	100%
r(5) → r(1)	100%	w(4) → r(6),r(5)	100%
r(6) → r(7)	75%	w(5) → r(5)	100%
w(2) → r(2)	100%	w(5) → r(6)	100%
w(4) → r(1)	75%	w(7) → r(6)	100%
w(4) → r(7),r(6)	75%	w(7) → r(5),r(3)	100%
w(5) → r(7),r(6)	100%		
w(6) → r(6)	100%		
Pre-write rules			
		w(4) → w(5)	100%
Post-write rules			
w(4) → r(7)	75%	w(5) → w(4)	100%
w(5) → r(7)	75%	w(7) → r(3)	100%
w(4) → r(6)	75%		
w(5) → r(6)	75%		
w(2) → w(6)	100%		

As shown in Table 2, some items do not have dependency roles because: (1) its appearance in the user log file is rare (less than minimum support). (2) Its correlation with other items is weak (less than minimum confidence); this means that the users can access them in any context. However, these items may be sensitive.

On the other hand, the item, which has a role with confidence less than 100%, appears in another context sometimes equal to (100 - minimum confidence). However, the dependency model rejects any transaction conflict with the dependency roles; this means that, some normal transactions will be detected as malicious (false positive error).

As example, in Table 1, user1 has a T2: r(1), r(5), w(1), r(4), r(5), w(4), although, this is a normal transaction, it conflicts with the roles r(1) → r(6) and w(4) → r(7),r(6).

This means that, it will be detect as a malicious transaction and cause a false positive error. On the other hand, if the user1 submits the transaction T: r(2), w(2), r(4), r(7), w(1), r(6), w(5), r(1), w(4), It will be detected as normal transaction because it does not conflict with any role, However, it is a malicious transaction.

This observation leads us to search for other solutions rather than the association roles, which depend on the principles of support and confidence in their traditional form.

3.2 The Enhanced Data Dependency Model

A transaction is an executing program that forms a logical unit of database processing. It includes one or more database access operations, these can include insertion, deletion, modification, or retrieval operations [8].

Based on this definition, transaction can be described by the number of its operations, read data set, write data set, and dependency roles. Again, we will use the transaction log file as shown in Table 1. Although transactions can be described through different attributes, we will use the same example to get clear comparison. let us look for the transactions in table1 from this point of view. Table 3 describes the transactions in Table 1 as they number of its operations, read data set and write data set. According to the available data in Table 1, suppose that, each write with read immediately before it forms an operation.

Table 3. The Transactions Representation

Trans. ID.	No. of Operations	Read data set	Write data set
1	2	7,1,6,1	5,4
2	2	1,5,4,5	1,4
3	4	7,6,2,7,3,1,6,3,5,2	4,6,2,5
4	4	2,4,7,6,1	2,3,5,4
5	3	6,1,1,2,7,4	3,6,2
6	1	5,3,6	7
7	1	2,5	6
8	1	4	6
9	3	6,5,3,4,3	5,4,7
10	2	5,6,5	5,4

Although, the read data set and write data set determine the correlation between data items; two transactions may have the same sets. So, data dependency roles are required to determine the manner of each one to access these data items. However, the data dependency means as in the previous models can be achieved if the read and write data sets save the order of items as it appear in transaction. Additional data dependency roles are required to specify the transaction well done. For each write operation in transaction, generate the read, pre-write, and post-write sets as shown in Table 4.

Table 4. Read, pre-write and post write sets for transactions

Trans. ID.	Read set	Pre- write set	Post-write set
1	w(5) →r(7),r(1),r(6) w(4) → r(1)		
2	w(1) → r(1),r(5) w(4) → r(4),r(5)		
3	w(4) → r(7),r(6),r(2) w(6) → r(7),r(3) w(2) → r(1), r(6) w(5) → r(3),r(5),r(2)	w(2) → w(6) w(5) →w(2)	w(6) →w(2) w(2) →w(5)
4	w(2) →r(2) w(3) → r(4),r(7) w(5) →r(6) w(4) → r(1)		
5	w(3) → r(6),r(1) w(6) →r(1) w(2) → r(2),r(7),r(4)		
6	w(7) → r(5),r(3),r(6)		
7	w(6) → r(2),r(5)		
8	w(6) → r(4)		
9	w(5) → r(6),r(5) w(4) → r(3),r(4) w(7) → r(3)		
10	w(5) → r(6),r(5) w(4) → r(5)	w(4) → w(5)	w(5) → w(4)

In Table 4, the read set determines the data items that transaction reads before writes an item in the same operation. The pre-write means that update a data item dependent in an updated data item. The post-write set highlight that the transaction updates a data item value after update another data item. Figure 2 simplifies the detection process, which done after authorization and process.

4. EXPERIMENTAL RESULTS

Several experiments were conducted for evaluating the performance of the proposed model in terms of false positive and true positive rates. False positive rate (FP), which is the probability that the IDS outputs an alarm when there is no intrusion, and true positive rate (TP), which is the probability that the IDS outputs an alarm when there is an intrusion [5].

Two different database logs were generated for testing both true positive rate and false positive rate of our model. The first log consisted of normal user transactions. The second log consisted of synthetic database transactions, which were treated as malicious transactions. These

malicious transactions were generated randomly based on the assumption that the attacker might not have the knowledge of data dependencies in user database.

We apply the models on [10, 20] with minimum support 0.20 and minimum confidence 0.85 and our proposed model on these log files and calculate the FP rate and TP rate for each model. Figure 3 shows the FP and TP rates in the three models.

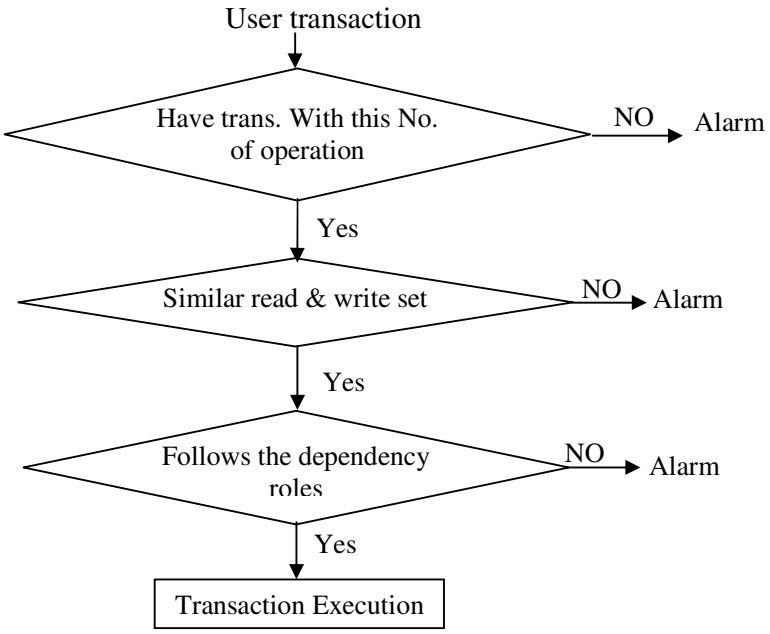

Figure 2. The sequence of the detection process

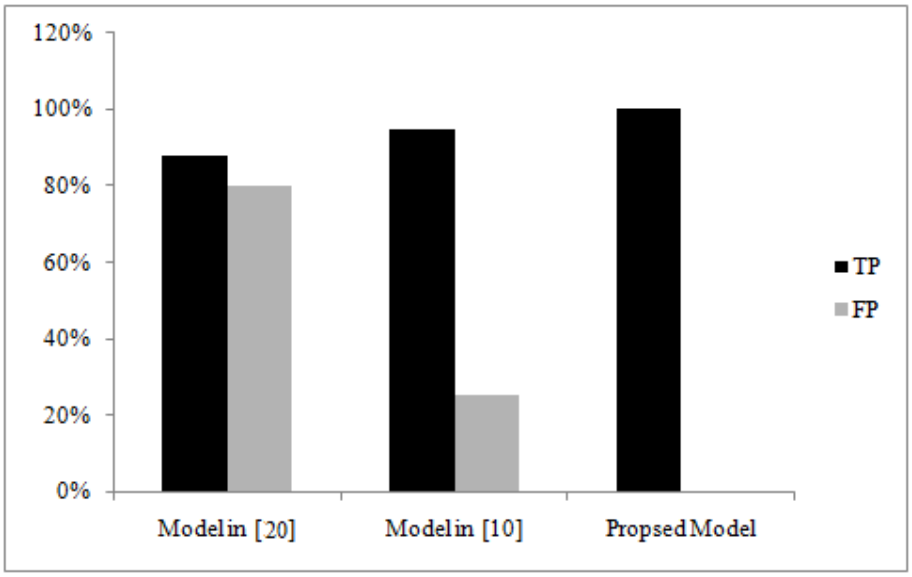

Figure 3. Comparison of the TP and FP rates in the three models

First, we apply the three models on the log file that contains the normal transactions, which used in the design phase and hence any alarm raised at this stage is interpreted as a false positive.

The second part of our experiments is assessing the true positive rates of the models. In this part, we feed the three models with the second log, which comprises intrusion transactions, and count the number of transactions that detect as intrusion to inform us about the true positive rate.

Based on this experiment and our generated log files, and as shown in Figure 3, the proposed model achieves better performance than the rival methods.

Our model output error only in two cases, fist, if the user tries to perform a malicious event using his normal transaction in different context, in this no alarm rise. This situation can be eliminated using additional factor as update latency, execution frequency, time of execution, etc. The second case occurs when the user amends his normal transaction, or defines new ones, and this can be solved by update the user's normal behavior.

5. CONCLUSIONS

In spite of the significant role of databases in information systems, not enough attention has been paid to intrusion detection in database systems. A limited number of techniques have been proposed in the last few years for the detection of intrusion in databases. Therefore, there is still an urgent need to exert more effort to improve the performance of those systems. In this context, this paper introduces an enhanced data dependency model. The proposed model describes the normal transactions of each user using the number of operations in each one, read data set, write data set and dependency rule. The experiment on synthetic database transactions illustrates that the proposed model reduces the false positive rate and achieves better performance than the rival methods. Hence, it overrides the shortage of traditional data dependency model. The future work will enhance the model by add additional feature to define the normal behavior of each user.

REFERENCES

[1] Kamra, A. Terzi, E. & Bertino, E., (2008) "Detecting anomalous access patterns in relational databases", VLDB journal, vol. 17, no. 5, pp 1063-1077

[2] Clarke, J., (2009) *SQL injection attacks and defense*, Syngress, Burligton.

[3] Jin, X. & Osborn, S. L., (2007) "Architecture for data collection in database intrusion detection systems", in Secure Data Management, W. Jonker and M. etkovic, Eds. Springer-Verlag, Berlin, pp. 96–107

[4] Liu, P., (2002) "Architectures for intrusion tolerant database systems", In Proceedings of the Annual Computer Security Applications Conference (ACSAC), pp 311-320

[5] Gu, G. Fogla, P. Dagon, D. Lee, W. & S˘koric, B., (2006) "Measuring Intrusion Detection Capability: An Information Theoretic Approach", ASIACCS '06, Taipei, Taiwan.

[6] Solomon, G. & Chapple, M., (2005) *Information Security Illuminated*, Jones & Bartlett Learning, USA

[7] Rezk, A. Ali, H. Elmikkawy, M. & Barakat, S., "Database intrusion detection system – A short survey", Accepted for publishing in Mansoura Journal of Computer and Information Sciences (MJCIS).

[8] Elmasri, R. & Navathe, S. B., (2007) *Fundamentals of Database System*, 5th ed., Addison Wesley

[9] Shulman, A., (2006) "Top Ten Database Security Threats - How to Mitigate the Most Significant Database Vulnerabilities", white paper, Imperva Inc.

[10] Rezk, A. Ali, H. Elmikkawy, M. & Barakat, S., (2011) "Integrating a Database Intrusion Detection System with Access Control", In Proceedings of the 5th international conference on intelligent computing and information system, Egypt, pp 46-52

[11] Lee, S.Y., Low, W.L. & Wong, P.Y., (2002) "Learning fingerprints for a database intrusion detection system", In: Gollmann, D., Karjoth, G., Waidner, M. (Eds.) *ESORICS 2002*. LNCS, vol. 2502, pp. 264–280. Springer, Heidelberg

[12] Gupta, B. Arora, D. & Jha, V. (2010) "An Anomaly Based Approach for Intrusion Detection by Authorized Users in Database Systems", In: Prasad, S.K. et al. (Eds.) *ICISTM 2010,* CCIS 54, pp. 348–356

[13] Fonseca, J. Vieira, M. & Madeira, H., (2008) "Online detection of malicious data access using DBMS auditing," SAC'08, March 16-20, pp 1013-1020, (Fortaleza, Ceará, Brazil, 2008)

[14] Fonseca, J. Vieira, M. & Madeira, H., (2007) "Integrated intrusion detection in databases", 3rd Latin-American Symposium on Dependable Computing (LADC 2007), Morelia, Mexico, pp 198- 211

[15] Chagarlamudi, M. Panda, B. & Hu, Y., (2009) "Insider threat in database systems: Preventing malicious users' activities in databases", 6th International Conference on Information Technology: New Generations, pp 1616-1620

[16] Bockermann, Christian; Apel, Martin & Meier, Michael (2009) "Learning SQL for Database Intrusion Detection Using Context-Sensitive Modelling (Extended Abstract)", In: Flegel, U. and Bruschi, D. (Eds.) *DIMVA 2009*, LNCS 5587, pp. 196–205

[17] Mathew, S., Petropoulos, M., Ngo, H. Q. & Upadhyaya, S., (2010) "A data-centric approach to insider attack detection in database systems", In Recent Advances in Intrusion Detection (RAID) Symposium, Springer, pp 382- 401.

[18] Hu, Y. & Panda, B., (2004) "A Data Mining Approach for Database Intrusion Detection", ACM Symposium on Applied Computing, pp 711 – 716.

[19] Srivastava, A. Sural, S. & Majumdar, A.K., (2006) "Database intrusion detection using weighted sequence mining", Journal of Computers, VOL. 1, NO. 4, pp 8-17

[20] Hashemi, S. Yang, Y. Zabihzadeh, D. & Kangavari, M., (2008) "Detecting intrusion transactions in databases using data item dependencies and anomaly Analysis", Expert Systems, Vol. 25, No. 5, Blackwell Publishing Ltd, November 2008, pp 460-473

[21] Hu, Y. & Panda, B., (2010) "Mining Inter-transaction Data Dependencies for Database Intrusion Detection", In: Sobh, T. (ed.), *Innovations and Advances in Computer Sciences and Engineering*, Springer, pp 67 -72.

[22] Panigrahi, S. Sural, S. & Majumdar, A. K. (2010) "Two-stage database intrusion detection by combining multiple evidence and belief update ", Information Systems Frontiers, Springer, pp 1-19

[23] Pinzón,C. De Paz, Y. & Cano, R. (2008) "Classification Agent-Based Techniques for Detecting Intrusions in Databases" In: Corchado, E. Abraham, A. & Pedrycz, W. (Eds.), *HAIS 2008*, LNAI 5271, pp. 46–53

[24] Pinzón, C. Herrero, Á. De Paz, J. F. Corchado, E. & Bajo, J. (2010) "CBRid4SQL: A CBR Intrusion Detector for SQL Injection Attacks" In: Rodriguez, E.S. C. et al. (Eds.): *HAIS 2010*, Part II, LNAI 6077, pp. 510–519