Available online at www.sciencedirect.com

ScienceDirect

Journal of Electrical Systems and Information Technology 3 (2016) 261–269

www.elsevier.com/locate/jesit

Machine learning approach to detect intruders in database based on hexplet data structure

Saad M. Darwish

Department of Information Technology, Institute of Graduate Studies and Research, Alexandria University, 163 Horreya Avenue, El-Shatby 21526, P.O. Box 832, Alexandria, Egypt

Received 15 July 2015; received in revised form 30 October 2015; accepted 15 December 2015
Available online 11 August 2016

Abstract

Most of valuable information resources for any organization are stored in the database; it is a serious subject to protect this information against intruders. However, conventional security mechanisms are not designed to detect anomalous actions of database users. An intrusion detection system (IDS), delivers an extra layer of security that cannot be guaranteed by built-in security tools, is the ideal solution to defend databases from intruders. This paper suggests an anomaly detection approach that summarizes the raw transactional SQL queries into a compact data structure called hexplet, which can model normal database access behavior (abstract the user's profile) and recognize impostors specifically tailored for role-based access control (RBAC) database system. This hexplet lets us to preserve the correlation among SQL statements in the same transaction by exploiting the information in the transaction-log entry with the aim to improve detection accuracy specially those inside the organization and behave strange behavior. The model utilizes naive Bayes classifier (NBC) as the simplest supervised learning technique for creating profiles and evaluating the legitimacy of a transaction. Experimental results show the performance of the proposed model in the term of detection rate.

Keywords: Database security; Anomaly detection; Database intrusion detection; Role-based profiling

1. Introduction

In today's business realm, the most valuable asset of organizations is its information and thus needs efficient management and protection. Over the last few years, database systems constitute the central of the information systems infrastructure because they permit the efficient administration and retrieval of huge amounts of data in addition to offer mechanisms that can be employed to certify the integrity of the stored data. Data found in these databases vary between private information, banking transactions, personal medical data, and commercial contracts, etc. Any violation

E-mail address: saad.darwish@alex-igsr.edu.eg

Peer review under the responsibility of Electronics Research Institute (ERI).

ELSEVIER Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jesit.2015.12.001

of security to these databases will lead to passing reputation of the organization, loss of customers' sureness and might even lead to lawsuits.

There are many ways to secure databases like user authentication, transaction data encryption, data watermarking, and intrusion detection. Each of these ways has its benefits and they should work together to reach the maximum security level of databases. User authentication is a prevention technique that prevents unauthorized users from gaining access to the database. Transaction data encryption is a prevention technique also; which thwarts attackers from understanding the data in case of sniffing on the session. Watermarking the data is a detection technique that used to pledge data integrity. Intrusion Detection is a detection technique that is used to identify the malicious activities as early as possible if the system prevention mechanisms were bypassed to minimize the harm caused by intruders; for more information regarding these techniques readers can refer to Panda and Giordano (1999) and Agrawal and Kiernan (2002).

Actually, available database security mechanisms are not basically designed to detect intrusions; they are intended to avoid the intruders. So, intrusion detection system is considered to be the second defense line. Database Intrusion is commonly defined as a set of actions that try to violate data integrity, data confidentiality or data availability; while database intrusion detection is the process of tracking transactions submitted to database and analyzing them to detect the possible presence of intruders. In general, there are two types of database intrusion attacks (I) insider and (II) outsider (Panda and Giordano, 1999). The insider's attacks are the ones when an intruder has all the privileges to access the database but he performs malicious actions. The outsider's attacks are the ones when the intruder does not have proper rights to access the database. He attempts to first rush into and then performs malicious actions. Detecting insider attacks are often more difficult than detecting outsider's attacks.

In the literature, database intrusion detection has two main models (Agnew, 2003; Chandola et al., 2009), anomaly detection and misuse detection. The anomaly detection model is based on the profile of a user's normal behavior. It analyzes a user's current session and compares it against the profile representing his normal behavior. If a major deviation is found during the comparison, an alarm will be raised. This includes monitoring the system states over a period of time; with assume that this monitored profile can mark the "normal" profile of the system. In general, it is not easy to implement an anomaly detection solution because profiling "normal" usage is a challenging task and anomaly detection methods are also subjected to high false-positive rates, particularly with over-fitted "normal" profiles. On the other hand, relaxing the margins on these profiles might result in missed attacks. In this case, fine-tuning anomaly detection systems to find the optimum thresholds is a major issue. In contrast, the misuse detection model is based on comparing user's session or commands with the signature of attacks previously used by attackers. So, the signature of attacks should be well known to be detected in this model inside regarding data. Misuse detection is clever to detect only known attacks. However, misuse detection can sometimes detect new attacks which share characteristics with previously known attacks.

The possibility of enhancing existing anomaly detection systems by presenting an effective database IDS constitutes the objective of this work. The contribution of this paper is an anomaly detection system explicitly adapted to the RBAC database system. This system builds and maintains a role profile representing precise and consistent user behavior by means of special data structure that is able to determine role intruders. The recommended data structure is transaction-based which extracts relationship among queries in the same transaction instead of the state-of-the-art approaches that is query-based approach that can detect the attributes which are to be referred together but cannot detect the queries which are to be executed together.

The paper is organized as follows: next section presents some of state-of-the art work in detecting anomalous database requests. Then Section 3 discusses in detail the proposed work in this field. Section 4 reports the related experimental results and then Section 5 end with concluding the paper and demonstrate some preliminary ideas for future work.

2. Related work

In recent years, researchers have proposed a variety of approaches for increasing the intrusion detection efficiency and accuracy. But most of these efforts concentrated on detecting intrusion at the network or operating system level (Kang et al., 2005; Ghosh et al., 1999). For instance, the technique presented in Kang et al. (2005) attempts to profile normal behavior for a program. It tries to conclude the normal system call sequences and save these sequences in the database as normal program access patterns. These efforts, however, are not adequate for protecting the database and not able of detecting malicious data corruption (i.e., what particular data in the database are manipulated by which specific malicious database transaction).

In the last few years, IDS for the database have been starting to take place. Very limited research deliberated the field of database intrusion detection. For example, the approaches in Lane and Brodley (1997) and Ming and Feng (2003) proposed architectures for intrusion tolerant database systems. However, these approaches are more focused on architectures for intrusion detection, and database recovery in case of an attack instead of proposing specific algorithms for performing the intrusion detection task on a DBMS. The technique developed in Liu (2002) has used time signatures in finding out database intrusions. It reviews the database behavior at the level of sensor transactions. Sensor transactions are normally small in size and have predefined semantics such as write only operations and well defined data access patterns. If a transaction tries to update a temporal data which has already been updated during that period, an alarm is raised. But, this algorithm is suitable only to real-time database systems.

In addition to the previous approaches, the work in Lee et al. (2000) can be regarded as a DBMS specific ID mechanism where the authors fingerprint access patterns of genuine database transactions and using them to identify possible intrusions. They summarize SQL queries into regular expression fingerprints. The given query is reported as malicious if it does not match any of the existing fingerprints. However, generating and maintaining the complete set of fingerprints for all transactions is a hard activity. Moreover, if any of the legal transaction fingerprints are missing, it may cause many false alarms.

Among the most noticeable approaches in database intrusion detection are those in Lee et al. (2002), Hu and Panda (2003), and Hu and Panda (2004). The techniques in Lee et al. (2002) and Hu and Panda (2003) are provided for discovering relationships among transactions and use this information to check hidden anomalies in the database log file. They are based on the following idea: if a data item is updated, this update does not happen alone but is accompanied by a set of other activities that are also logged in the database log files. So, they find out the dependency among data items where data dependency refers to the access correlations among data items. Transactions that do not follow any of these mined data dependency rules are marked as malicious transactions. These approaches find out malicious transactions by comparing those sets for different item updates (the read set, the pre-write set, and the post-write set); however these techniques do not focus on role profiles.

The method mentioned in Hu and Panda (2004) is a misuse detection system tailored to relational database systems. It uses audit-log data to construct profiles that describe typical users' activities. It identifies data items frequently accessed together and saves this information for later comparison. If the system determines substantial incidents of data items that are accessed together, but not in normal patterns as identified before, an anomaly is marked. The disadvantage of such an approach is that the number of users for a database system can be quite large and maintaining/updating profiles for such large number of users are not easy tasks. Architectures for Hippocratic databases (Chung et al., 1999) have been planned as a mechanism to protect the privacy of data they manage. But, even though the architecture includes intrusion detection as a core component, it does not specify any methods for performing the intrusion detection task.

Approaches which are conceptually most similar to this work are Agrawal et al. (2002), Kamra et al. (2008a,b), and Bertino et al. (2005). The technique proposed by the authors in Agrawal et al. (2002) is detecting user/role anomalous access to the DBMS; it profiles normal user/role behavior and uses it for detecting anomalous access; it also uses a profiling technique to detect SQL injections. Furthermore, the methods declared in Kamra et al. (2008a) and Bertino et al. (2005) are talking about creating profiles for each role; not for each user. The method suggested in Kamra et al. (2008b) creates its own data structure (Triplet), which stores three basic data about the executed SQL statement (the command, tables accessed, columns accessed) and then uses it to create role profiles. This method is an extended version to work in Agrawal et al. (2002), in which the authors upgraded their data structure to contain information about the where clause to increase the efficiency of intrusion detection system. The new data structure (Quiplet) stores five basic information about the SQL statement (the command, tables accessed, columns accessed, tables accessed in where clause, columns accessed in where clause).

The approach stated by the researchers in Kamra et al. (2008a) uses its own data structure to create a profile for each role in the database. This type of data structure preserves the dependency between SQL statements in the transaction and uses it in further comparisons to detect anomaly access behavior in the database. In their approach, database log file is read to extract the list of tables accessed by the transaction and list of attributes read and written by transactions. The main disadvantages of this approach are that; it does not keep any info about where clause in the data structure and doesn't extract the correlation among queries in the same transaction. Most of the IDSs that are discussed above show a lot of variance in their accuracy and efficiency. The main difficulty encountered by most of them is that any endeavor to improve the rate of correct detection of intrusion, generally causes an increase in the false alarms as well.

In this work, the attention is to build an intrusion detection scheme that employs the concept of creating profiles based on transaction sequence for each role in the database by means of utilizing the usage of database log file to extract profile features with the aim to improve intrusion detection performance. The main challenge in attacking the intrusion detection problem is to extract the right information from the database traces so that accurate profiles can be built. To address this problem, the paper proposes a new representation for the database log records, which could hold information about a whole transaction not only a single query. Algorithms in Agrawal et al. (2002), Kamra et al. (2008b) and Bertino et al. (2005) are very similar to the suggested approach, but this approach surpasses them in keeping dependency between SQL statements in the same transaction.

3. The proposed system

The approach this study follows is similar to the one suggested by Kamra et al. (2008a). However, the study improves the representation of SQL commands to also include information about the sequence of SQL commands in the transaction. The suggested ID system builds a profile for each role and is able to conclude role intruders, that is, individuals that while holding a definite role diverge from the normal activities of that role. With respect to ID, using roles means that the number of profiles to form and maintain is much smaller than those one would need when considering individual users (Rao et al., 2010). This implies that an ID solution, based on RBAC, could be easily deployed in practice.

The two difficulties reported in this work are as follows: how to construct and conserve profiles signifying precise and stable user behavior; how to employ these profiles for carrying out the ID task at hand. The main challenge in attacking these difficulties is to extract the right information from database traces (a set of intrusion-free training records representing normal user behavior) so that accurate profiles can be built. When role information exists, the problem is transformed into a supervised learning problem. Compared to the work in Kamra et al. (2008b), the proposed system employs a new representation of the database log records that holds information about the whole transaction's commands not only the list of attributes read and written by transactions as in the comparable work. By using this representation, the system may possibly enhance intrusion detection performance.

3.1. System design

The system's design consists of three central constituents: the traditional DBMS tool that grips the query execution process, the database audit log files and the ID mechanism. These modules form the new stretched DBMS that is enriched with an independent ID system operating at the application level. The flow of interactions for the ID process is shown in Fig. 1. Every time a transaction is issued, it is examined by the ID mechanism before execution. First, the system transforms the new transaction into a new data structure supported by the ID mechanism (hexplet). Then, the system checks the hexplet contrary to the existing profiles and submits the assessment of the transaction (anomalous vs. not anomalous) to the response engine. The response engine refers a policy base of existing response mechanisms to issue a reply depending on the assessment of the transaction submitted by the comparison process. The most common action is to send an alert to the security administrator. However, other actions are possible such as disable the role and disconnect the user making the access or drop the query. If by the assessment, the transaction is not anomalous, the response engine simply updates the database audit log and the profiles with the transaction information. Before the detection phase, the learning phase should be performed to create the initial profiles from a set of intrusion free records from the database audit log. The readers can refer to the work of Kamra et al. (2008a) for a detailed overview of algorithms for the production of intrusion free records.

3.2. Hexplet data structure

In order to identify user behavior, the recommended system uses the database log file for querying information regarding users' actions. The log records, after being processed, are used to form preliminary profiles representing acceptable actions. Each one or more entry (each single transaction) in the log file is represented as a separate data

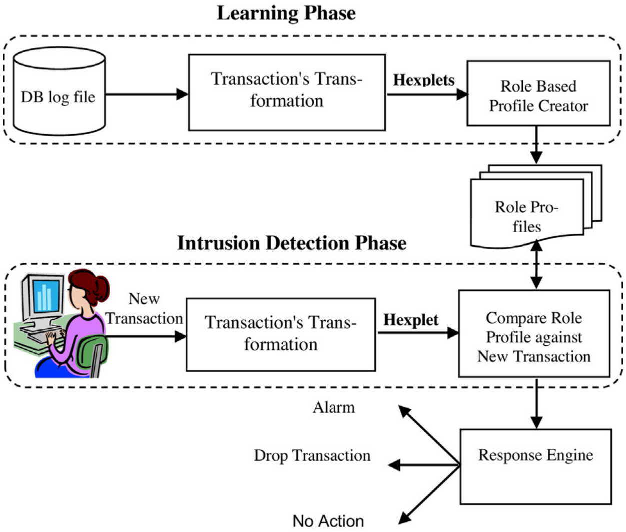
Learning Phase
DB log file
Transaction's Trans-formation
Hexplets
Role Based Profile Creator
Role Pro-files
Intrusion Detection Phase
New Transaction
Transaction's Trans-formation
Hexplet
Compare Role Profile against New Transaction
Alarm
Drop Transaction
No Action
Response Engine

Fig. 1. The proposed intrusion detection system.

unit; these units are then combined to form the desired profiles. Here, the assumption is that SQL queries related to the same transaction are marked together in the log file.

In order to construct profiles, the system needs to preprocess the log file items and translate them into a format that can be analyzed. Therefore, it symbolizes each transaction by a basic data unit that holds six fields, and thus it is called a hexplet. User actions are characterized using sets of such hexplets. Each hexplet represents a single transaction and contains the following information: the first five elements represent the first SQL command in the transaction; the SQL command issued by the user, the set of relations accessed, and for each such relation, the set of referenced attributes. the data unit also processes the qualification component of the query to extract information on relations and their corresponding attributes, which are used in the query predicate; the sixth element holds information about the rest of SQL commands in the transaction (if any).

For the sake of simplicity, the system characterizes a generic hexplet using a 6-ary relation H(c, P_R, P_A, S_R, S_A, R_{SQL}), where c corresponds to the first SQL command, P_R to the projection relation information, P_A to the projection attribute information, S_R to the selection relation information, S_A to the selection attribute information, and R_{SQL} to the linked SQL commands (the rest SQL commands in the transaction). Table 1 shows two different transactions and their

Table 1
Examples of hexplet construction.

Transaction	Hexplet
Start Transaction	<select>
SELECT	< 1; 1 >
R1:A1;R1:C1;R2:B2;R2:D2	< [1; 0; 1; 0]; [0; 1; 0; 1] >
FROM R1;R2	< 1; 1 >
WHERE R1:A1 = R2:B2	< [1;0; 0; 0]; [0; 1; 0; 0]>
End Transaction	<*Null*>
Start Transaction	<select>
SELECT	< 1; 1 >
R1:A1;R1:C1;R2:B2;R2:D2	< [1; 0; 1; 0]; [0; 1; 0; 1] >
FROM R1;R2	< 1; 1 >
WHERE R1:A1 = 5 and R2:B2 = 5	< [1;0; 0; 0]; [0; 1; 0; 0]>
Delete From R2	<[<delete>
Where R2:A2 = 17 and R2:B2 = 5	<0; 1>
End Transaction	<[0; 0; 0; 0]; [0; 0; 0; 0]>
	<0; 1>
	<[0;0; 0; 0]; [1; 1; 0; 0]>]>

representation according in hexplets. The example considers a database schema consisting of two relations R1 = {A1; B1; C1; D1} and R2 = {A2; B2; C2; D2}. The complete representation of the hexplet is (SQL-CMD, PROJ-REL-BIN [], PROJ-ATTR-BIN [][], SEL-REL-BIN[], SEL-ATTR-BIN[][], RELAT-SQL-CMD[]). The first field is symbolic and corresponds to the first SQL command in the transaction, the second is a binary vector that contains 1 in its i-th position if the i-th relation is projected in the SQL query. The third field is a vector of n vectors, where n is the number of relations in the database. Element PROJ-ATTR-BIN[i][j] is equal to 1 if the SQL query projects the j-th attribute of the i-th relation; it is equal to 0 otherwise. Likewise, the fourth field is a binary vector that contains 1 in its i-th position if the i-th relation is used in the SQL query predicate. The fifth field is a vector of n vectors, where n is the number of relations in the database. Element SEL-ATTR-BIN[i][j] is equal to 1 if the SQL query references the j-th attribute of the i-th relation in the query predicate; it is equal to 0 otherwise. The sixth field is a vector with length equal to the number of remaining SQL commands in the transaction; each element in the vector holds the same preceding five elements.

For example if a transaction consists of three SQL Commands; then the first five fields in the hexplet will hold information about the first SQL command in the transaction and the sixth field will be a vector of two elements; each element consists of the same previous structure (SQL-CMD, PROJ-REL-BIN[], PROJ-ATTR-BIN[][], SEL-REL-BIN[], SEL-ATTR-BIN[][]) and R_{SQL} holds the information about the rest of about the rest of SQL commands comes after in the transaction.

To summarize, in query based approach, both queries of the same transaction are recorded in different data structure (i.e. different actions); whereas in the proposed approach complete transaction is recorded in one data structure (i.e. queries form one action). Consider for example that the user issued the following transaction:

```
Start Transaction
  Delete From R2   Where R2:A2 = 17 and R2:B2 = 5
  Select * From R2
End Transaction
```

In query based approach, this action is considered as normal. It may result in false positive. But, as the proposed approach binds all the queries of the same transaction in one behavior, it will certainly reduce the false positive rate.

3.3. Classifier

This work employs the Naive Bayes Classifier (NBC) for the ID task in RBAC-administered databases. The motivation for utilizing NBC is its low computational requirements for both the training and classification task. The small running time is mainly due to the attribute independence assumption. For a better understanding of the concepts underlying the NBC, the reader is encouraged to refer to the works in Kamra et al. (2008b) and Bertino et al. (2005) that explain the optimality region for the NBC and discuss the reasons for its effective performance even when the attribute independence assumption does not hold.

In the classification problem, a set of training examples is provided, and a new instance with attribute values is given (correspond to the set of observations). The goal is to predict the target value, or the class, of this new instance. The technique that is defined here is to assign to this new instance the most probable class value $v_{class} \varepsilon V$, given the attributes $(a_1, \ldots, a_n)$ that describe it. That is

$$v_{class} = \arg\max_{v_j \varepsilon V} P(v_j | a_1, a_2, \ldots, a_n) \tag{1}$$

Using Bayes Theorem anyone can rewrite the expression as (Bertino et al., 2005)

$$\begin{aligned} &= \underset{v_j \varepsilon V}{\operatorname{argmax}} \frac{P(a_1, a_2, \ldots, a_n | v_j) P(v_j)}{P(a_1, a_2, \ldots, a_n)}, \\ &\propto \underset{v_j \varepsilon V}{\operatorname{argmax}} P(a_1, a_2, \ldots, a_n | v_j) P(v_j), \\ &\propto \underset{v_j \varepsilon V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j). \end{aligned} \tag{2}$$

In this case, estimating $P(v_j)$ is simple since it requires just counting the frequency of v_j in the training data. $P(a_i|v_j)$ requires only a frequency count over the tuples in the training data with class value equal to v_j. To tackle the problem

Table 2
Evaluation results of the proposed anomaly detection approach.

	All connections	New attacks excluded
Detection rate (%)	92.6	96.4
False positive rate(%)	1.68	1.68

of zero probability (number of observations is very small in large training sets or zero), the system adopts a standard Bayesian approach in estimating this probability as discussed in Kamra et al. (2008a).

The NBC is directly applied to the proposed anomaly detection framework by considering the set of roles in the system as classes and the log file hexplets as the observations. In what follows, the researcher illustrates how equation 1 can be applied for the suggested data type (hexplet). If R denotes the set of roles, the predicted role of a given observation ($\mathrm{c_i}$, $\mathrm{P_{R_i}}$,$\mathrm{P_{A_i}}$, $\mathrm{S_{R_i}}$,$\mathrm{S_{A_i}}$, $\mathrm{R_{SQL_i}}$) is

$$r_{class} = \arg\ \max_{r_j \varepsilon R} \prod_{c=1}^{N_C} p(r_j)p(c_i|r_j) \times \left\{ \prod_{i=1}^{N} \left\{ p(S_R[i].S_A[i]|r_j)p\left(S_R[i].S_A[i]|r_j\right) \right\} \right\} \tag{3}$$

where N is the number of relations in the DBMS and N_c is the number of SQL Commands in the transaction. With the above equation in place, the ID task is quite straightforward. For every new transaction, its r_{class} is predicted by the trained classifier. If this r_{class} is different from the original role associated with the transaction, an anomaly is detected. For benign transactions, the classifier can be updated in a direct way by increasing the frequency count of the relevant attributes. The procedure for ID can easily be generalized for the case when a user is assigned more than one role at a time. This is because the suggested method detects anomalies on a per transaction basis rather than per user basis. Hence, as long as the role associated with the transaction is consistent with the role predicted by the classifier, the system will not detect an anomaly.

4. Experimental evaluation

In this section, several experiments were performed on the DBMS of Microsoft SQL Server 2000 on Microsoft Windows 7 Home Premium SP1 for testing the performance of the proposed approach. In all experiments, a real dataset consists of 2000 transactions (6500 SQL statements) is used to evaluate the proposed approach. The database itself consists of 55 tables with 665 different attributes in all. In the database, there exist 8 roles that access the database with various read only and read-write roles. The dataset used for training should be intrusion free (trusted transactions) that is extracted from the database log file after revision for this purpose. The intrusion/anomalous queries were generated by reading the database log file and change the role assigned to each transaction randomly (approximately 25% of the transactions). The queries in this dataset consist of a mix of select, insert, update, and delete commands. Furthermore, numerous fresh, never-before-seen attacks have been used in order to access the generalization aptitude of the ID system. For the experimental evaluation, the performance of the suggested approach was evaluated in terms of false positive and detection rates, which are estimated as follows (Tajbakhsh et al., 2009):-

$$\text{False positive rate} = \frac{\text{No. of false positive}}{\text{No. of normal connections}} \tag{4}$$

$$\text{Detection rate} = 1 - \frac{\text{No. of false negative}}{\text{No. of attack connections}} \tag{5}$$

where false positive (negative) rate is the number of normal (attack) connections labeled as an attack (normal). The obtained results are reported in Table 2. It can be concluded that the achieved detection rate is approximately optimum for intrusion detection phase (100% detection rate of the learning phase) while the false positive rate is reserved much lower in most cases. Some classification results on the same dataset using another technique (Kamra et al., 2008b) are shown in Table 3. As compared to query-based approach, it can be easily verified that the proposed approach can reduce the false positive rate.

Although the execution time of the approach presented in Kamra et al. (2008a) is better than the suggested one that takes about 110 s for a total training time over 2000 transactions. This is because the hexplet-based detection

Table 3
Comparative resuts.

	Proposed approach	Query-based approach (Kamra et al., 2008b)
Detection rate (%)	96.4	91.3
False positive rate(%)	1.68	14.5

system is more complex than their system, which deals with a single SQL statement not a transaction (two or more SQL statements). Finally, total execution time of the recommended approach is about 120 s (containing training and testing), which is completely reasonable with regard to the number of training and testing records. The complexity of the detection algorithm is $O\ (R \times A \times N \times N_c)$ where R is the number of roles in the database, A is the number of attributes considered by the classifier, N is the number of relations in the DBMS and N_c is the number of SQL Commands in the transaction. This gives us a chance to discover the opportunity of integrating the approach with other query processing features of a database for an integrated online ID mechanism entrenched inside a database.

5. Conclusions and future work

This paper suggested a transaction-based anomaly detection system in RBAC databases. This machine learning system is capable of extracting valuable information from the log files regarding the access patterns of queries in the same transaction. The use of roles that are employed for training a classifier makes the system practical, even for databases with a large user population. As compared to query-based approaches, the suggested system will certainly diminish the false positive rate as it has deliberated the correlation among queries of a transaction. Future works include elaborating other machine learning techniques to improve the intrusion detection accuracy and study the case when role information is not present in the log records.

References

Agnew, G., 2003. Secure Electronic Transactions: Overview, Capabilities, and Current Status. A Chapter in Payment Technologies for E-commerce Book. Springer-Verlag, ISBN 3-540-44007-0, pp. 211–226.

Agrawal, R., Kiernan, J., 2002. Watermarking relational databases. In: Proceedings of the 28th International Conference on Very Large Databases, Hong Kong, August 20–23, pp. 155–166.

Agrawal, R., Kiernan, J., Srikant, R., Xu, Y., 2002. Hippocratic databases. In: Proceedings of the 28th International Conference on Very Large Databases, Hong Kong, August 20–23, pp. 143–154.

Bertino, E., Kamra, A., Terzi, E., Vakali, A., 2005. Intrusion detection in RBAC-administered databases. In: Proceedings of the 21st Annual Computer Security Applications Conference, USA, December 05–09, pp. 170–182.

Chandola, V., Banerjee, A., Kumar, V., 2009. Anomaly detection: a survey. ACM Comput. Surv. 41 (3), 1–58.

Chung, C., Gertz, M., Levitt, K., 1999. DEMIDS: a misuse detection system for database systems. In: Proceedings of the 3rd International Working Conference on Integrity and Internal Control in Information Systems, Netherlands, November 18–19, pp. 159–178.

Ghosh, A., Schwartzbard, A., Schatz, M., 1999. Learning program behavior profiles for intrusion detection. In: Proceedings of the 1st Workshop on Intrusion Detection and Network Monitoring, USA, April 9–12, pp. 51–62.

Hu, Y., Panda, B., 2003. Identification of malicious transactions in database systems. In: Proceedings of 7th International Database Engineering and Applications Symposium, Hong Kong, July 16–18, pp. 329–335.

Hu, Y., Panda, B., 2004. A data mining approach for database intrusion detection. In: Proceedings of ACM Symposium on Applied Computing, Cyprus, March 14–17, pp. 711–716.

Kamra, A., Bertino, E., Lebanon, G., 2008a. Mechanisms for database intrusion detection and response. In: Proceedings of the 2nd SIGMOD PhD Workshop on Innovative Database Research, Canada, June 13, pp. 31–36.

Kamra, A., Terzi, E., Bertino, E., 2008b. Detecting anomalous access patterns in relational database. VLDB J 17 (5), 1063–1077.

Kang, D., Fuller, D., Honavar, V., 2005. Learning classifiers for misuse and anomaly detection using a bag of system calls representation. In: Proceedings of 3rd IEEE International Workshop on Information Assurance, USA, March 23–24, pp. 118–125.

Lane, T., Brodley, C., 1997. Sequence matching and learning in anomaly detection for computer security. In: Proceedings of AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management, USA, July 27–28, pp. 43–49.

Lee, V., Stankovic, J., Son, S., 2000. Intrusion detection in real-time database systems via time signatures. In: Proceedings of 6th IEEE Real-Time Technology and Applications Symposium, USA, May 31–June 02, pp. 124–133.

Lee, S., Low, W., Wong, P., 2002. Learning fingerprints for a database intrusion detection system. In: Proceedings of the 7th European Symposium on Research in Computer Security, Switzerland, October 14–16, pp. 264–280.

Liu, P., 2002. Architectures for intrusion tolerant database systems. In: Proceedings of the 18th Annual Computer Security Applications Conference, USA, December 09–13, pp. 311–320.

Ming, Z.J., Feng, M.J., 2003. Intrusion-tolerant based architecture for database system security. J. Xidian Univ. 30 (1), 85–89.

Panda, B., Giordano, J., 1999. Defensive information warfare. ACM Commun. 42 (7), 30–32.

Rao, U.P., Sahani, G., Patel, D., 2010. Machine learning proposed approach for detecting database intrusions in RBAC enabled databases. In: Proceedings of the International Conference on Computing Communication and Networking Technologies, India, July 29–31, pp. 1–4.

Tajbakhsh, A., Rahmati, M., Mirzaei, A., 2009. Intrusion detection using fuzzy association rules. Appl. Soft Comput. 9, 462–469.