An overview of anomaly based database intrusion detection systems

Mohammed Masoud Javidi[1], Marjan Kuchaki Rafsanjani[1], Sattar Hashemi[2] and Mina Sohrabi[1,3]

[1]*Department of Computer Science, Shahid Bahonar University of Kerman, Kerman, Iran*
[2]*School of Electrical Engineering and Computer Sciences, Shiraz University, Shiraz, Iran*
[3]*Young Researchers Society, Shahid Bahonar University of Kerman, Kerman, Iran*
javidi@uk.ac.ir, kuchaki@uk.ac.ir, s_hashemi@shirazu.ac.ir, mina.sohrabi@gmail.com

Abstract

Database security is a crucial concern today. One mechanism for safeguarding information stored in database systems is to use an Intrusion Detection System (IDS). Recently researchers are working on using machine learning techniques to increase the accuracy of the detection malicious attacks on database systems; Such as mining data dependencies among data items, access patterns of users and learning SQL commands. In this paper, we survey some intrusion detection approaches, which use these techniques. Also, we discuss the advantages and disadvantages of the approaches and compare them with considering their different features.

Keywords: Security, Database systems, Intrusion detection, Machine learning, Data dependency.

1. Introduction

Information is the most valuable asset of organizations and companies in our modern networked world. Some data stored in databases is worth millions of dollars, but existing security models are insufficient to prevent misuse, especially insider abuse by legitimate users. Hence, a growing number of researches have concentrated on handling the vast variety of malicious attacks to theses data. Malicious attacks to database systems may cause corrupted data, which can spread very fast to other parts of the database system through valid users. Organizations need to be able to detect the intrusions into their network and database systems as soon as possible so that they can prevent further damage to sensitive data. Deploying an intrusion detection system (IDS) makes data less vulnerable to these attacks and enables early detection of attacks.

Most researchers in intrusion detection field focus on detecting intrusions at network or operating system level (Javitz & Valdes, 1991; Frank, 1994; Noel *et al.,* 2002; Ertoz *et al.,* 2004; Qin & Hwang, 2004; Kuchaki Rafsanjani, 2010; Renjit & Shunmuganathan, 2011; Kuchaki Rafsanjani *et al.,* 2012) but IDSs at network or operating system level are not appropriate for detecting malicious attacks in database systems. There are two main reasons for database IDSs. The first is that actions considered malicious for a database application are not necessarily malicious for the network or the operating system and therefore IDSs specifically designed for the network or the operating system would not be effective for database protection. The second is that IDSs designed for networks and operating systems do not provide adequate protection to databases against insider threats (Kamra & Bertino, 2008).

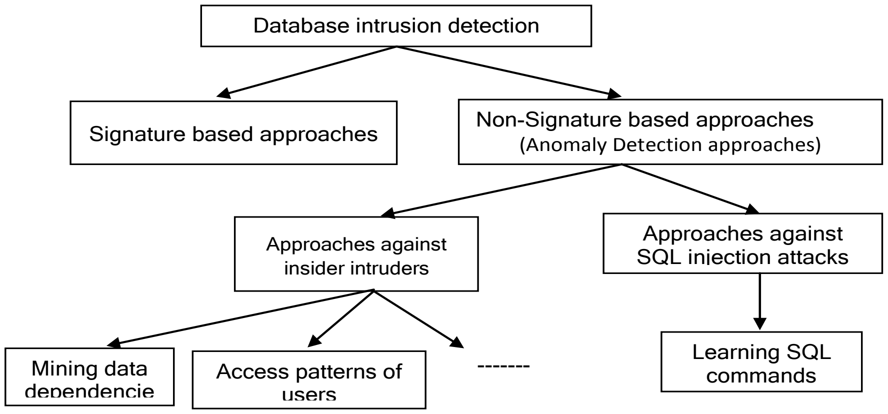
Database intrusion detection approaches can be classified into two groups based on their intrusion classification mechanisms. An intrusion classification mechanism can use signature based or non-signature based intrusion detection methods. To detect attacks, the signature based IDS is configured with a number of signatures of online attacks and identifies future attacks that match these signatures. Unfortunately, it is not a trivial task to keep intrusion detection signatures up to date because a large number of new intrusions take place daily. To overcome this problem, the IDS should be coupled with anomaly detection schemes, which support detection of new attacks. The non-signature based database IDS profiles patterns of normal user transactions and uses these patterns for identifying intrusive behavior. Transactions not compliant with the patterns are identified as malicious transactions. The patterns can also be used for defending against insider threats, too.

Recently researchers are applying artificial intelligence and data mining in database intrusion detection to increase the accuracy of detection. Kamra and Bertino (2008) have surveyed machine learning methods for database security. They have discussed three anomaly detection techniques for detecting insider threat behavior in context of a DBMS and have also presented an overview of two approaches that use machine learning techniques to detect SQL injection attacks. Javidi *et al.* (2010) have discussed some database intrusion detection systems. In this paper, we discuss other existing approaches that detect anomalies for detecting insider abuses. Some of these techniques are as follows: Mining data dependencies among data items, access patterns of users and learning SQL commands. We discuss the pros and cons of these techniques. Also we have an overview on an approach that detects SQL injection attacks and finally we present a web-based approach that detects SQL attacks.

1.1 Anomaly Detection against insider intruders (Non-Signature based approaches)

Insider threats are much more difficult to defend against because they come from subjects that are legitimate users of the system, and thus may have access rights to data and resources (Karma & Bertino, 2008). Many methodologies such as mining data dependencies among data items, access patterns of users have been proposed for defending against insider abuse attacks. In following we discuss some of these methodologies in detail (Fig.1).

Fig.1. *Categorization of existing database IDSs*


Database intrusion detection
Signature based approaches
Non-Signature based approaches
(Anomaly Detection approaches)
Approaches against insider intruders
Approaches against SQL iniection attacks
Mining data dependencie
Access patterns of users

Learning SQL commands

1.2 Mining data dependencies among data items

Hu and Panda (2004) observed that in real-world database applications, although the transaction program changes often, the whole database structure and essential data correlations rarely change. So they proposed using dependency among data items to detect intrusion transactions in database systems; transactions not compliant with the mined data dependencies are identified as malicious ones. Since, the other researches (Srivastava *et al.,* 2006; Hashemi *et al.,* 2008; Hu & Panda, 2010) have used the idea of using data dependencies among data items for detecting intrusion transactions in database systems. In following we discuss approaches that use data dependencies for detecting intrusion transactions and present their advantages and limitations.

1.2.1 A data mining approach for database intrusion detection

The approach proposed by Hu and Panda (2004) obtains patterns of normal transactions by mining data dependencies among data items in database systems. Data dependencies discovered by the *data dependency miner* component are employed as classification rules for identifying anomalies. This approach identifies a new user transaction as anomalous if it does not conform to the patterns of normal transactions, mined by data dependency miner component. They use the association rule mining paradigm for extracting dependencies among data items. We illustrate the procedure used by their proposed database IDS in Fig. 2 using an artificial example. We should mention that in Fig. 2, integers are representative of data items in the database. Before explaining different steps of the procedure, we introduce some concepts, which are used for extracting dependencies among data items.
The input log file of transactions consists of the sequences of operations performed by transactions. For instance consider the following update statement in a transaction.
Update Table 1 set u = x + y where z = 20;

In this statement, before updating u, values of x, y and z must be read and the new value of u is calculated. So the operational transaction would be: <r(x), r(y), r(z), w(u)>. For instance, after each normal deposit or withdrawal in a banking database, i.e. changing the balance of a customer, the data item time is expected to be updated to record commitment time of the transaction. In this scenario, the account balance and time are two data items, and for the transaction to be committed normally the following operations might be executed in sequence: read the data item account balance, update it and then update the data item time, say r(account balance), w(account balance), w(time). So <w(account balance), w(time)> is the write sequence of data item account balance. Also <r(account balance), w(account balance)> is the write sequence of the data item account balance and <r(account balance), w(time)> is the write sequence of data item time.

As is shown in Fig. 2, the first phase for extracting data dependencies among data items is concerned with mining the frequent sequential patterns. Given a database D of transactions, the problem of mining frequent sequential patterns

(Agrawal & Srikant, 1995) is to find the maximal sequences among all sequences that have a certain user-specified minimum support. Here frequent sequential patterns indicate data items that are accessed together with a fixed order and more than a user specified minimum support. Then, mined frequent sequential patterns that have any write operations are selected and their read and write sequences which are suitable for generating the dependency rules are extracted.

Table 1. *Comparison of some datab*ase IDSs

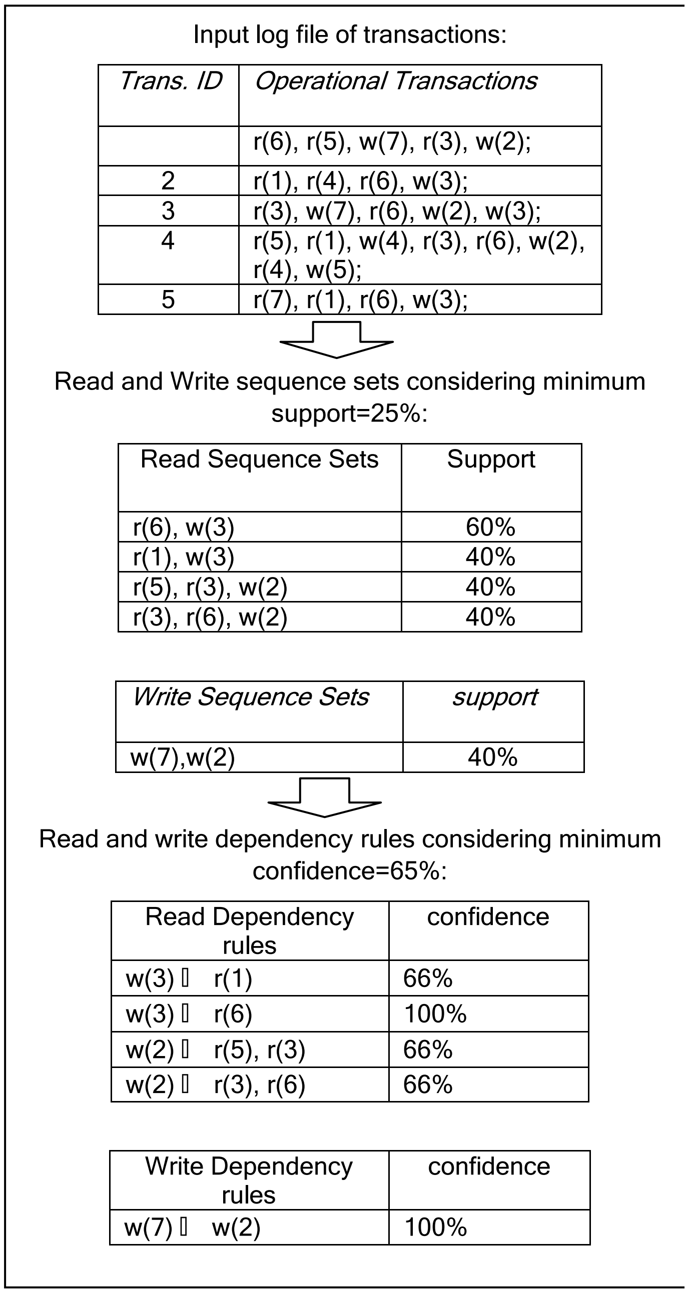
Methods	Technique	Features
Chung et al. (2000)	Access patterns of users	Generates profiles of users by using audit logs and detects insider abuse with the derived profiles.
Lee at al. (2000)	Tagging time signatures to data items	Suitable for Real-time databases.
Barbara et al. (2002)	Building database behavioral models by HMM	Recognizes malicious patterns by using HMM.
Hu & Panda (2004)	Mining data dependencies among data items	It detects malicious transactions targeted at corrupting data. It's less sensitive to the change of user behaviors and database transactions.
Bertino et al. (2005)	Access patterns of users	Under an RBAC system: Suitable for large databases. Can be deployed very easily in practice.
Srivastava et al. (2006)	Weighted data dependency rule miner (WDDRM)	It improves the approach offered by Hu & Panda (2004). It generates rules for possibly less frequent but more important attributes by considering the sensitivity of the attributes.
Kamra et al. (2008)	Access patterns of users	They have developed three models, of different granularity, to represent the SQL queries appearing in the database log files.
Hashemi et al. (2008)	Dependencies among data items and time-series anomaly analysis	It improves the approach offered by Hu & Panda (2004). The concept of malicious transactions is extended from those that corrupt data to those that either read data or write data or both without permission.
Hu & Panda (2010)	Mining Inter-transaction Data Dependencies	It improves the approach offered by Hu & Panda (2004). They cluster legitimate user transactions into user tasks for discovery of inter-transaction data dependencies.

Patterns that do not have any write operations are not considered, because they do not have any information about malicious transactions that are targeted at corrupting data. Then dependency rules are extracted from strong read and write sequences, which have a greater confidence than a pre specified threshold. The mined data dependencies are in the form of classification rules. For instance $w(7) \rightarrow w(2)$ is a write dependency rule that shows that data item 2 is most likely to be updated after data item 7 and $w(3) \rightarrow r(1)$ is a read dependency rule, which says data item 1 probably needs to be read before data item 3 is updated. The transactions not compliant with the data dependencies generated are identified as malicious transactions. Weighted intra-transactional rule mining for database intrusion detection:

Srivastava et al. (2006) found that the IDS proposed by Hu and Panda (2004) does not consider sensitive attributes that are accessed less frequently than minimum support , although they may result in interesting rules. Sensitivity of an attribute shows the importance of the attribute for tracking against malicious modifications. If sensitive attributes are to be tracked for malicious modifications then generating data dependency rules for these attributes is essential because if there is not any rule for an attribute, the attribute cannot be checked. The approach proposed by Hu and Panda does not generate rules for very sensitive attributes which are accessed less frequently because it does not consider the sensitivity of attributes. The motivation of Srivastava et al. (2006) for dividing attributes in to different sensitivity groups and assigning weights to each group is to generate dependency rules for possibly less frequent but more important attributes. They call their algorithm "weighted data dependency rule miner" (WDDRM). After generating weighted data dependency rules, the algorithm marks the transactions which do not follow the extracted data dependencies as malicious.

Also with respect to the size of current databases being increased at the number of attributes, they considered that it is very difficult for administrators to keep track of all attributes whether they are accessed or modified correctly or not so they divided the attributes into different categories based on their relative importance or sensitivity. Therefore, the administrator checks only those alarms, which are generated due to malicious modification of sensitive data instead of checking all the attributes.

Fig.2. *Artificial example for describing the main procedure used by database IDS of Hu and Panda (2004)*



Input log file of transactions:

Trans. ID	*Operational Transactions*
	r(6), r(5), w(7), r(3), w(2);
2	r(1), r(4), r(6), w(3);
3	r(3), w(7), r(6), w(2), w(3);
4	r(5), r(1), w(4), r(3), r(6), w(2), r(4), w(5);
5	r(7), r(1), r(6), w(3);

Read and Write sequence sets considering minimum support=25%:

Read Sequence Sets	Support
r(6), w(3)	60%
r(1), w(3)	40%
r(5), r(3), w(2)	40%
r(3), r(6), w(2)	40%

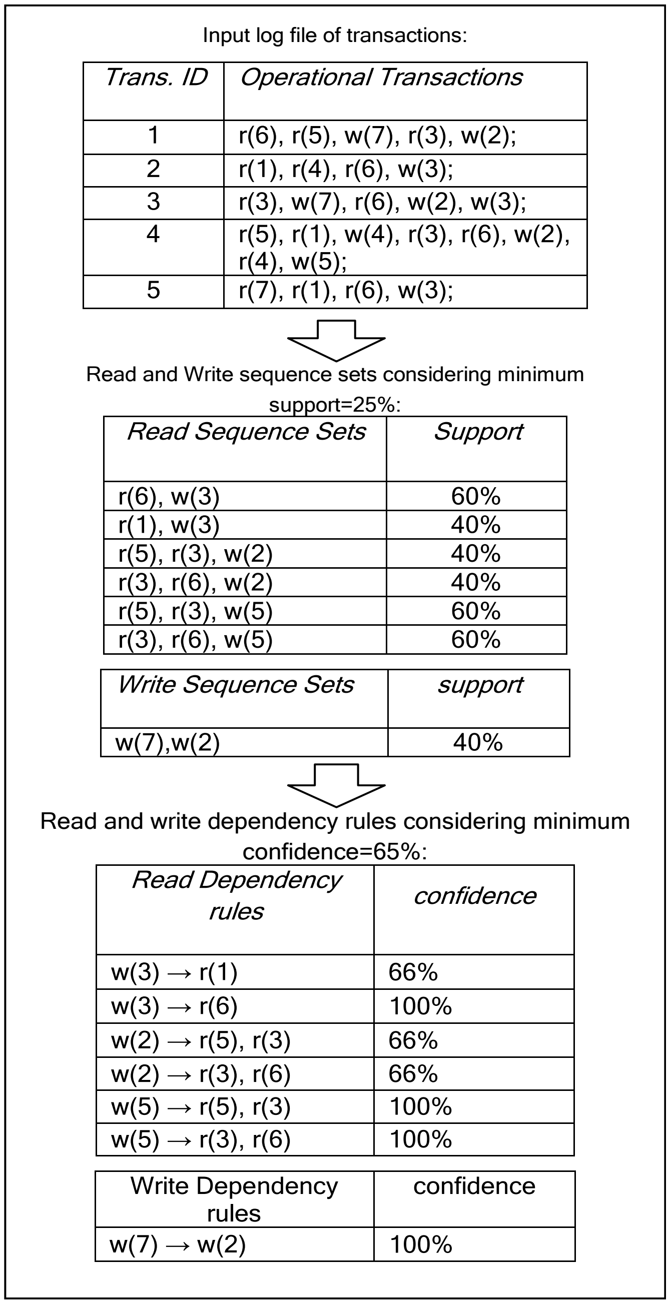
Write Sequence Sets	*support*
w(7),w(2)	40%

Read and write dependency rules considering minimum confidence=65%:

Read Dependency rules	confidence
w(3) ▯ r(1)	66%
w(3) ▯ r(6)	100%
w(2) ▯ r(5), r(3)	66%
w(2) ▯ r(3), r(6)	66%

Write Dependency rules	confidence
w(7) ▯ w(2)	100%

Fig.3. *Artificial example for describing the main procedure used by database IDS of Srivastava et al. (2006)*



Input log file of transactions:

Trans. ID	*Operational Transactions*
1	r(6), r(5), w(7), r(3), w(2);
2	r(1), r(4), r(6), w(3);
3	r(3), w(7), r(6), w(2), w(3);
4	r(5), r(1), w(4), r(3), r(6), w(2), r(4), w(5);
5	r(7), r(1), r(6), w(3);

Read and Write sequence sets considering minimum support=25%:

Read Sequence Sets	*Support*
r(6), w(3)	60%
r(1), w(3)	40%
r(5), r(3), w(2)	40%
r(3), r(6), w(2)	40%
r(5), r(3), w(5)	60%
r(3), r(6), w(5)	60%

Write Sequence Sets	*support*
w(7),w(2)	40%

Read and write dependency rules considering minimum confidence=65%:

Read Dependency rules	*confidence*
w(3) → r(1)	66%
w(3) → r(6)	100%
w(2) → r(5), r(3)	66%
w(2) → r(3), r(6)	66%
w(5) → r(5), r(3)	100%
w(5) → r(3), r(6)	100%

Write Dependency rules	confidence
w(7) → w(2)	100%

We illustrate their weighting algorithm by an illustrative example using the input log file of transactions in Fig. 3. Suppose that data item 5 is a data item with high sensitivity compared with other data items in Fig. 3 and the weight of w(5) is 3. As is obvious the update of data item 5 has just occurred in transaction ID number 4 and therefore the support of w(5) is 20%, which is less than 25% the minimum support. As can be seen, the procedure used by Hu and Panda (2004) does not generate any frequent Read/Write sequences containing w(5). So there is no chance that a rule will be created for it. In what follows, we discuss how Srivastava's et al. (2006) proposed algorithm, WDDRM, solves this problem. WDDRM calculates support of sequence s with weight of w_s as follows. If s is present in n transactions out of N transactions, then the support of sequence s is:

Support(s) = $(n * ws) / N$ (1)

The support of w(5) is calculated considering weight of the data item 5, i.e., 3. By equation 1, the support of w(5) is as follows:

Support w(5) = $(1*3) / 5 = 60\%$ (2)

The effect of this weighted approach on the sequence mining algorithm is significant. Sequences containing very sensitive attributes (data item 5) but accessed less in the transactions (20%) can become frequent sequences because each such sequence count is enhanced by multiplying by its weight. The weighted support can now exceed the minimum support, i.e., 25%. In Fig. 3, we have illustrated the procedure of extracting dependency rules considering data item 5 as a high sensitive attribute. Compared with the procedure proposed by Hu and Panda (2004) more sequence sets and more dependency rules are extracted. As is obvious two read sequence sets, i.e., “r(5), r(3), w(5)” and “r(3), r(6), w(5)” are included in Read sequence sets and two dependency rules, i.e., “w(5) → r(5), r(3)” and “w(5) → r(3), r(6)” are included in Read dependency rules.

Srivastava et al. (2006) compared their work with the non-weighted dependency rule mining approach, i.e., Hu and Panda (2004). They carried out several experiments and showed that WDDRM performs better than the non-weighted dependency rule mining approach. However WDDRM has the limitation that the process of weighting to data items is carried out by hand.

Detecting intrusion transactions in databases using data item dependencies and anomaly analysis in time series: The approach proposed by Hashemi et al. (2008) also has fewer limitations than that of Hu and Panda’s approach (Hu & Panda, 2004). Their approach identifies malicious transactions using a data mining technique, which detects malicious transactions more effectively. The approach proposed by Hu and Panda can only detects malicious transactions that corrupt data items and cannot identify transactions that read data without permission. This results in a significant reduction in the detection rate. Also, when a dependency rule is violated by a transaction, Hu and Panda, without considering the confidence of the rule, always identify the transaction as an intrusion. This leads to a high false positive rate. In addition, sometimes consistency rules of a database do not allow users to execute any arbitrary transaction and therefore dependency between data items is no longer violated though there may be the some abnormal variations in the update pattern of each data item.

The types of intrusions that are not detected by the previous models are studied in Hashemi’s et al. (2008) paper. By detecting these intrusions, they are able to significantly improve the intrusion detection rate. Their approach has three advantages. First, the dependency rules among data items are extended not only to detect transactions that write data without permission, but also to detect transactions that read data without permission. Second, a novel behavior similarity criterion is introduced to reduce the false positive rate of the detection. Third, time-series anomaly analysis is conducted to identify intrusion transactions, which update data items with unexpected patterns.

They detected intrusion transactions in databases using two components. (1) The first component extracts dependency rules among data items, which are represented by the order in which these items are accessed (read or write). It is similar to the approach Hu and Panda, but the concept of malicious transactions is extended from those that corrupt data to those that either read data or write data or both without permission. In addition, when a transaction t violates a dependency rule, it is not immediately identified as malicious. Their approach examines whether there exists any normal transaction, which violates t and has similar behavior to it. If such a transaction exists, t is not considered as an intrusion. This is desirable because, in reality, dependency rules are not always 100% correct. (2) The second component uses anomaly analysis of the time series corresponding to each data item. This anomaly analysis approach can detect intrusion transactions, which cannot be identified by first component. This component extracts the time series from the normal transaction log for each data item and discretizes it using clustering techniques. This component uses a suffix tree structure to efficiently discover normal update patterns and their frequencies for the corresponding data item based on the discretized representation of each time series. The anomaly weight of each pattern is calculated by the difference between its occurrence frequency in the transaction in question and in the normal time series. Finally, using weighted output integration, the final decision is made by combining the outputs of the above two components.

Hashemi et al. (2008) conducted several experiments for evaluating the performance of their proposed method comparing with the performance of the approach presented by Hu and Panda (2004), in terms of false positive and true positive rates and they showed that their approach performs better in all experiments.

Mining inter-transaction data dependencies for database intrusion detection: Hu *et al.* (2010) considered that an attacker may launch a group of malicious transactions so well crafted that each of them appears normal. They proposed a data mining method for clustering legitimate user transactions into user tasks for facilitating the discovery of inter-transaction data dependencies. Then they illustrated their methodology for mining inter-transaction write sequences and dependencies. They claimed that simply applying the sequential pattern mining algorithm across database transactions for profiling legitimate access patterns is not useful for facilitating discovery of inter-transaction data dependencies and that it is critical to develop an algorithm that clusters user transactions into user tasks. They define a user task as a group of transactions that are semantically related to perform a user task. For clustering user transactions into user tasks they first clustered the database log by user ID i.e. all transactions submitted by the same user are clustered into one group inside which the relative execution order of transactions is kept. Then the transactions submitted by the same user are clustered again so that transactions belonging to the same user task are grouped into a cluster. This is done using data mining method based on the observation that normally the time gap between two adjacent transactions inside a user task is significantly smaller than that between two adjacent user tasks.

Then they illustrated the methodology for mining inter-transaction write sequences and inter-transaction write dependencies. First, the transactions in each user task need to be preprocessed. This is because the transactions in the database log not only contain write operations but may also have read operations and the methodology proposed only considers inter-transaction data dependencies based on the updating operations. Also information about aborted transactions should be filtered. Any transactions submitted by system administrators are also removed since the system maintenance activities are not considered as a part of "normal" system operations. After this data preprocessing phase, a filtered user task only contains write operations of each transaction in the task. The algorithm for generating inter-transaction write dependencies and inter-transaction write sequences have been presented in Hu *et al.* (2010). Since inter-transaction write dependencies specify data dependencies across user transactions regardless of data updating sequence, discovering this kind of data correlation is intrinsically similar to the procedure of association rule mining. Although in this case, we try to find association rules across different transactions, the clustered user tasks are considered as atomic units that are analyzed for association rule discovery. The inter-transaction write sequences illustrate data update sequences. Therefore mining this category of data relationships is similar to the sequential pattern mining process. The experiments confirm the advantage of mining inter-transaction data dependencies for detecting malicious database transactions.

1.2.2 Access patterns of users

Some database IDSs profile patterns of normal user behavior and use these patterns for identifying intrusive behavior, so that the transactions not compliant with the patterns are identified as malicious.

DEMIDS: A misuse detection system for database systems

Chung *et al.* (2000) proposed DEMIDS (DEtection of MIsuse in Database Systems) for relational database systems. This misuse detection system uses audit logs to build profiles, which describe the typical behavior of users working with the database systems by specifying the typical values of features audited in audit logs. These profiles can be used to detect both intrusion and insider abuse, but in particular DEMIDS detects malicious behavior by legitimate users who abuse their privileges. By a database scheme and associated applications, some working scopes comprising certain sets of attributes, which are often referenced together with some values, will be formed by the access patterns of users. The idea of working scopes has been captured well by the concept of frequent item-sets, which are sets of features with certain values. DEMIDS defines a notion of distance measure, which measures the closeness of a set of attributes with respect to the working scopes, by integrity constraints (the data structure and semantics, which are encoded in the data dictionary) and the user behavior reflected in the audit logs. Distance measures are used for finding the frequent item-sets in the audit logs, which describe the working scopes of users by a novel data mining approach. Then misuse can be detected by comparing the derived profiles against the security policies specified or against new information (audit data) gathered about the users.

1.2.3 Real-time database intrusion detection system via time signatures

Lee *et al.* (2000) proposed an intrusion detection system for real-time database systems via time signatures. Transactions often have time constraints in real time database systems (like in Stock Market applications). They exploit the real-time properties of data in intrusion detection. Their approach monitors the behavior at the level of sensor transactions, which are responsible for updating the values of real-time data. Sensor transactions have predefined semantics such as write-only operations and well defined data access patterns where as user transactions have wide varieties of characteristics.

Real time database systems deal with temporal data objects, which should be updated periodically and their values change over time. Therefore a sensor transaction is generated in every period. They Supposed that every sensor transaction consumes time e to complete the processing, where $0 < e < P$, P is the time period to update the temporal data and there is only one transaction in a period P. By applying this signature, if a transaction attempts to update a temporal data, which is already being updated in that period, an alarm will be raised.

1.2.4 Mining Malicious data corruption with hidden markov models

The approach proposed by Barbara *et al.* (2002) for database intrusion detection uses Hidden Markov Model (HMM) and time series to find malicious corruption of data. They used HMM to build database behavioral models, which capture the changing behavior over time and recognized malicious patterns with using them.

1.2.5 Intrusion detection in RBAC-administered databases

Bertino *et al.* (2002) proposed an intrusion detection system for database systems, which is conceptually very similar to DEMIDS. One important fact that they considered is that databases typically have very large number of users and therefore keeping a profile for each single user is not practical. Hence their approach is based on the well-known role-based access control (RBAC) model. It builds a profile for each role and checks the behavior of each role with respect to that profile. Under an RBAC system, permissions are associated with roles rather than with single users. With using roles, the number of profiles to build and maintain is much smaller than when considering individual users. Therefore their approach is usable even for databases with large user population.

"Other important advantages of RBAC are that it has been standardized (see the NIST model (Sandhu *et al.,* 2000)) and has been adopted in various commercial DBMS products as well in security enterprise solutions (Karjoth *et al.,* 2003). This implies that an ID solution, based on RBAC, could be deployed very easily in practice" (Bertino *et al.,* 2005).

Moreover, the approach used by DEMIDS for building user profiles assumes domain knowledge about the data structures and semantics encoded in a given database schema, which can adversely affect the general applicability of their methods, but Bertino *et al.* (2005) built profiles using syntactic information from the SQL queries, which makes their approach more generic than others' ones. They used Naïve Bayes Classifier to predict the role which the observed SQL command most likely belongs to, and compared it with the actual role. If the roles differ from each other, the SQL statement is considered illegal.

1.2.6 Detecting anomalous database access patterns in relational databases

The approach proposed by Kamra *et al.* (2008) is based on mining SQL queries, which are stored in database audit log files. They have developed three models, of different granularity, to represent the SQL queries appearing in the database log files. So their approach can extract useful information from the log files regarding the access patterns of the queries. They have considered two different scenarios. In the first case, they have assumed that the database has a Role Based Access Control (RBAC) model in place. This case is similar to the approach proposed by Bertino *et al.* (2005). Under an RBAC model, authorizations are specified with respect to roles and not with respect to single users and privileges are assigned to roles. This case is transformed into a supervised learning problem, because role information does exist. They have used Naïve Bayes Classifier (NBC) for intrusion detection task in this case. Their proposed system can detect role intruders, who are individuals while holding a specific role, behave differently than expected.

In the second scenario, they have assumed there are no roles associated with users. The specific methodology that is used for the ID task in this case is as follows: "the training data is partitioned into clusters using standard clustering techniques. In this setting, the clusters obtained after the clustering process represent the profiles. A mapping is maintained for every database user to its representative cluster. The representative cluster for a user is the cluster that contains the maximum number of training records for that user after the clustering phase. For every new query under observation, its representative cluster is determined by examining the user-cluster mapping. Note the assumption that every query is associated with a database user. For the detection phase, two different approaches are outlined. In the first approach, the naive bayes classifier is applied in a manner similar to the supervised case, to determine whether the user associated with the query belongs to its representative cluster or not. In the second approach, a statistical test is used to identify if the query is an outlier in its representative cluster. If the result of the statistical test is positive, the query is marked as an anomaly and an alarm is raised" (Kamra & Bertino, 2009)**.**

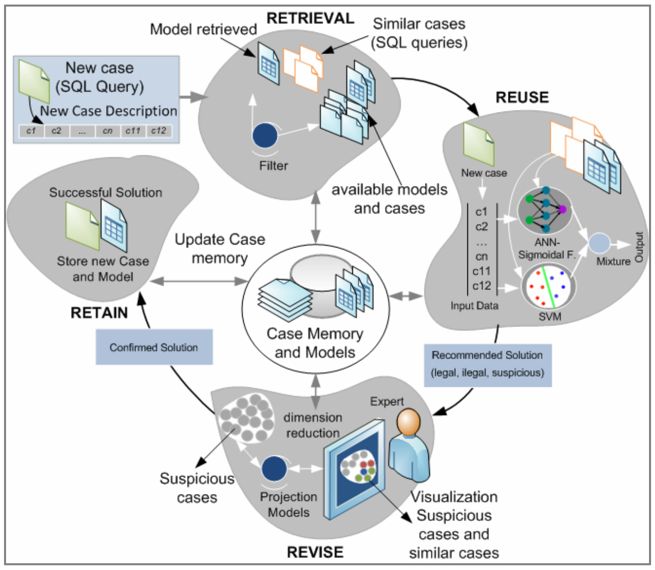
1.2.7 Anomaly detection against SQL injection attacks (Non-signature based approaches)

SQL Injection is an attack exploiting applications that construct SQL statements from user-supplied input. In this class of SQL attack, un-sanitized user input is able to change the structure of an SQL query such that when it is executed it has an unintended effect on the database. Several threat scenarios may arise because of the altered SQL queries. Injecting SQL queries may allow an attacker to get unauthorized access to the database. Moreover, the modified SQL queries are executed with the privileges of the application program. Therefore an attacker may abuse the privileges of the program in a manner unintended by the application program designers. In following we discuss an approach which is against SQL injection attacks.

1.2.8 A CBR Intrusion Detector for SQL Injection Attacks:

Nowadays SQL Injection attack is one of the most serious security threats around databases. Pinzón *et al.* (2010) considered that previous solutions for SQL injection attacks seem insufficient to block this type of attack because they lack the learning and adaptation capabilities for dealing with attacks and their possible variations in the future. In addition, they considered that the vast majority of solutions are based on centralized mechanisms with little capacity to work in distributed and dynamic environments (Fig. 4).

Fig.4. *CBR cycle and classification mechanism of the CBRid4SQL agent (Pinzón et al., 2010)*



So they used the intelligent agent CBRid4SQL (a CBR Intrusion Detector), which incorporates a Case-Based Reasoning mechanism, which provides it with a greater level of adaptation and learning capability for the classification of malicious codes. CBR is a paradigm which is based on the idea that similar problems have similar solutions. Thus, a new problem is resolved by consulting the case memory to find a similar case which has been resolved in the past. In terms of CBR, the case is composed of elements of the SQL Query. Some of the fields that define a case are as follows: User, Length_SQL_String, Start_Time_Execution and End_Time_Execution.

Also an integrated mixture through an Artificial Neural Network (ANN) and a Support Vector Machine (SVM) are used as a mechanism of classification in the CBRid4SQL agent's internal structure. So it is possible to exploit the advantages of both predictive strategies in order to classify the SQL queries in a more reliable way.

As you can see in Fig. 4, the reasoning cycle has four different stages. In the first stage, the retrieval stage, there is a selection of queries sorted by type and by the memory's classification models. In the second stage, the reuse stage, a Multilayer Perceptron (MLP) and an SVM are applied simultaneously to carry out the prediction of the new query. In the third stage, revise stage, in the case of the query resulting as suspicious, more inspection will be carried out by a human expert. At this stage a visualization neural technique is incorporated, which notably eases the revision stage carried out by human experts in the case of suspicious queries. At the last stage, retain stage, memory information regarding the cases and models will be updated.

1.2.9 Database intrusion detection in web databases

Web applications have become very popular nowadays. On the other hand developers of web-based applications often have little or no security skills. As a result, there are many security holes in web applications and existing prevention systems are often insufficient to protect them. Therefore, prevention mechanisms and signature-based detection systems should be complemented by anomaly detection systems, which learn the normal usage profiles associated with web-based applications and identify attacks as anomalous deviations from the established profiles. In following we discuss a learning based approach for the detection of attacks against back-end databases used by web-based applications (Table 1).

1.2.10 A learning-based approach to the detection of SQL attacks

The approach presented by Valeur *et al.* (2005) detects SQL attacks that exploit v ulnerabilities in Web-based applications to compromise a backend database. Their anomaly-based system uses multiple statistical models to build the profiles of benign access to the database. As with most learning-based anomaly detection techniques, during training phase the profiles are learned automatically by analyzing a number of sample database accesses. Then, during the detection phase, the system is able to identify anomalous queries that might be associated with an SQL attack.

Their proposed IDS taps into the communication channel between web-based applications and the database server. SQL queries performed by the applications are intercepted and sent to the IDS for analysis. The IDS parses each input SQL statements and returns a sequence of tokens. Flag of each token shows whether the token is a constant or not. Constants are the only elements that should contain a user supplied input (which may be malicious). Each constant has a data type attached to it. For each data type, a different statistical model is used. For instance for sting data type, the following models can be used: String length, String Character Distribution, String Prefix and Suffix Matcher, String Structure Inference and so on. A feature vector is created by extracting all tokens marked as constants. A profile is a collection of models, which the features are fed to in order to train the set of models or to generate an anomaly score.

The training phase has two steps. In the first step of the training phase, the data fed to the models is used for building the profiles associated with the models' parameters. The data processed in the training phase is normal. So during this step the models learn what benign queries look like. In the second step of the training phase, instead of updating the model parameters, an anomaly score is calculated based on how well the processed features fit the trained models. For each model, the maximum anomaly score observed during this step is stored and used to set an anomaly threshold. In detection phase, anomaly scores are calculated for each query. If an anomaly score exceeds the maximum anomaly score observes during training by a certain tunable percentage, the query is considered anomalous.

2. Conclusion

Despite the vast amount of work on network intrusion detection, there is a little work on database intrusion detection. Database Intrusion detection approaches can be categorized to signature based and non-signature based approaches. In this paper we focused on non-signature based approaches, which are anomaly based database IDSs. The techniques of the approaches are mining data dependencies, weighted data dependency rule miner, access patterns of users and learning SQL commands.

3. Acknowledgement

This work is supported by the Iran Tele Communication Research Center.

4. References

1• Agrawal R and Srikant R (1995) Mining sequential patterns. Proc. Int. Conf. Data Eng., Taipei, Taiwan. pp: 3-14.

2• Barbara D, Goel R and Jajodia S (2002) Mining malicious data corruption with hidden markov models. *Proc. 16th Annual IFIP WG 11.3 Working Conf. Data & Appl. Sec.,* Cambridge, England.

3• Bertino E, Kamra A, Terzi E and Vakali A (2005) Intrusion detection in RBAC-administered databases. *Proc. 21st Annual Comput. Sec. Appl. Conf.* pp: 170-182.

4• Chung CY, Gertz M and Levitt K (2000) Demids, a misuse detection system for database systems. Integrity & Internal Control Info. Sys., Strategic Views on the Need for Control. Norwell, MA, Kluwer Acad. Publ. 159-178.

5• Ertoz L, Eilertson E, Lazarevic A, Tan P, Srivava J, Kumar V and Dokas P (2004) The MINDS – Minnesota intrusion detection system. In: Next Generation Data Mining. MA. MIT Press, Boston.

6• Frank J (1994) Artificial Intelligence and intrusion detection, current and future directions. Proc. 17th National Comput. Sec. Conf.

7• Hashemi S, Yang Y, Zabihzadeh D and Kangavari M (2008) Detecting intrusion transactions in databases using data item dependencies and anomaly analysis. *Expert Sys.* 25(5), 460-473.

8• Hu Y and Panda B (2004) A Data mining approach for database intrusion detection. Proc. ACM Sym. Appl. Comput. pp: 711-716.

9• Hu Y and Panda B (2010) Mining inter-transaction data dependencies for database intrusion detection. Innovations and Advances in Computer Science and Engineering. Sobh T (Ed.), Springer.

10• Javidi MM, Sohrabi M and Kuchaki Rafsanjani M (2010) Intrusion detection in database systems. Proc. FGCN 2010, Part II, CCIS. 120, 93-101.

11• Javitz HS and Valdes A (1991) The SRI IDES statistical anomaly detector. IEEE Sym. Sec. & Privacy.

12• Kamra A and Bertino E (2009) Survey of machine learning methods for database security. Machine Learn. Cyber Trust: Security, Privacy, and Reliability, Tsai JJP and Yu PS (Eds.) Springer-Verlag.

13• Kamra A, Terzi E and Bertino E (2008) Detecting anomalous access patterns in relational databases. *The VLDB J*. 17(5), 1063-1077.

14• Karjoth G (2003) Access control with IBM tivoli access manager. *ACM Trans. Info. & Sys. Sec.(TISSEC)*. 6(2), 232-257.

15• Kuchaki Rafsanjani M (2010) Generalized intrusion detection in mobile ad hoc networks. *Indian J. Sci. &. Technol*. 3(10), 1098-1101.

16• Kuchaki Rafsanjani M, Aliahmadipour L and Javidi MM (2012) A hybrid Intrusion Detection by game theory approaches in MANET. *Indian J. Sci. &. Technol*. 5(2), 2123-2131.

17• Lee VC, Stankovic J and Son SH (2000) Intrusion detection in real-time database systems via time signatures. Proc. 6th IEEE Real Time Technol. & Appl. Sym.(RTAS'00). pp: 124.

18• Noel S, Wijesekera D and Youman C (2002) Modern intrusion detection, data mining, and degrees of attack guilt. *Appl. Data Mining in Comput. Security*. Dordrecht: Kluwer Academic.

19• Pinzón C, Herrero Á, De Paz JF, Corchado E and Bajo J (2010) CBRid4SQL. A CBR Intrusion detector for SQL injection attacks. Rodriguez ES, *et al.* (Eds) HAIS 2010, Part II, LNAI 6077. pp: 510-519.

20• Qin M and Hwang K (2004) Frequent episode rules for Internet traffic analysis and anomaly detection. Proc. IEEE Conf. Network Comput. & Appl. (NAC '04). IEEE Press, NY.

21• Renjit JA and Shunmuganathan KL (2011) Network based anomaly intrusion detection system using SVM. *Indian J. Sci. & Technol.* 4(9), 1105-1108.

22• Sandhu R, Ferraiolo D and Kuhn R (2000) The NIST model for role based access control: Towards a unified standard. Proc. 5th ACM Workshop on Role Based Access Control.

23• Srivastava A, Sural S and Majumdar AK (2006) Database intrusion detection using weighted sequence mining. *J. Comput.* 1(4), 8-17.

24• Valeur F, Mutz D and Vigna G (2005) A learning-based approach to the detection of SQL attacks. Proc. Int. Conf. Detection of Intrusions & Malware, & Vulnerability Assessment (DIMVA).