Intrusion Detection in Databases

José Fonseca
ESTG/CISUC
University of Coimbra - Portugal
josefonseca@ipg.pt

Abstract

Database management systems (DBMS), which are the ultimate layer in preventing malicious data access or corruption, implement several security mechanisms to protect data. However these mechanisms cannot always stop malicious users from accessing to the data by exploiting system vulnerabilities. This paper presents a study towards the proposal of an intrusion detection mechanism for DBMS. The approach uses a directed graph representing the profile of valid transactions to detect unauthorized ones, which are seen as invalid sequences of SQL[1] commands. This is done by analyzing the commands the users execute and comparing them to the profile of the authorized transactions previously learned.

1. Introduction

A major problem faced by organizations today is the protection of their data against malicious access or corruption. Database security mechanisms offer basic security features such as authentication, authorization, access control, data encryption, and auditing. However, these mechanisms do not assure protection against DBMS vulnerabilities exploits and are very limited in defending data attacks from the inside or from unsecured applications.

The general lack of intrusion detection features in commercial DBMS is an important limitation since malicious actions in a database environment may not be filtered by existing intrusion detection mechanisms at the network or the operating system levels. For example, inside attacks (e.g., a disgruntled employee that may access and damage critical private data) are particularly difficult to detect and isolate, because as these attacks are carried out by legitimate users that may have access rights to data and system resources. In addition, masquerade attacks where people hide their identity by impersonating others are one of the most frequent forms of computer security attacks [1]. Another example is SQL injection in web applications, where an attacker changes the SQL commands sent to the server, and therefore, accessing sensitive data.

Intrusion detection mechanisms for DBMS have not been studied much, in a clear contrast to what has happened in operating systems and networking fields. However, recent works have addressed concurrent intrusion detection and attack isolation in DBMS, and this issue is clearly receiving more and more attention. DEMIDS [2] is a misuse detection system tailored to relational database systems. It uses the audit logs to derive user profiles that describe typical behaviors. In [3] a real-time intrusion detection mechanism based on the profile of user roles obtained by mining database logs is proposed. An intrusion attack and isolation mechanism is proposed in [4]. Vieira and Madeira [5] propose a mechanism for the detection of malicious transactions (sequences of SQL commands creating a unit of interaction in DBMS) based on the manual definition of the transaction profiles.

This paper presents our research study towards the definition of an intrusion detection mechanism for DBMS. The approach is based on the fact that the users usually interact with the database through a well defined application interface. Therefore, the user does not execute ad-hoc transactions but are confined to those programmed in the code of the application. A database is a transactional environment and the set of transactions a given user executes may be viewed as an abstraction of that user's behavior. This represents the user's profile and the assumption is that knowing the profiles of all the users of the database is the key to detect intrusions. If a user's actions do not match a known profile, it is considered an intruder.

Our proposal is to describe each database transaction as a directed graph representing the corresponding sequence of SQL commands. In a first step, this sequence of commands is obtained by analyzing the execution profile of the database applications. After learning the authorized transactions the intrusion detector concurrently monitors the commands the user executes and com-

[1] SQL stands for Structured Query Language, the language used by relational DBMS

pares them with the directed graph.

The structure of the paper is as follows: Section 2 presents our current approach for transactions learning and intrusion detection. Section 3 presents some preliminary results. Section 4 concludes the paper and presents future research directions.

2. Proposed approach

In a typical database environment the transactions are programmed in the database application, which means that the set of transactions remains stable, as long as the database application is not changed. For example, in a banking database application users can only perform the predefined operations available at the application interface (e.g., withdraw money, balance check account, etc). No other operation is available for those users. Normally, end-users cannot execute ad-hoc SQL commands.

In our work we use the profile of the transactions implemented by the database applications to identify user attempts to execute malicious SQL commands. A database transaction is represented by a directed graph describing the different SQL execution paths from the beginning to the end of each transaction. The nodes in the graph represent commands and the arcs represent the valid command sequences. Depending on the data being processed, several execution paths may exist for the same transaction and an execution path may include cycles representing the repetitive execution of sets of commands (a typical example of cycles in a transaction is the insertion of a variable number of lines in a customer's order). The transaction ends with a commit or rollback command.

Our intrusion detection approach comprises a learning phase and a detection phase. In the first phase the transaction profiles are learned through the identification of the authorized sequences of SQL commands, which result in a directed graph. In the second phase they are used to detect deviating behaviors, which are considered to be intrusions.

One important security mechanism of DBMS is auditing. The audit trail can be used to perform a posteriori analysis of the operations done by each user in order to identify potential intrusion attempts. However, the analysis of the audit trail is a difficult and time consuming task and only serves for diagnosis or investigation purposes of past security attacks. The time between a malicious action and its detection is of major importance and every second of delay may represent loss of privacy, risk of data destruction, and propagation of corrupted data after the attack.

Our first approach to the intrusion detection problem is based on the information collected by the database auditing mechanism. The goal is to add on-line intrusion detection capabilities to typical DBMS auditing system. In the learning phase, the audit trail is used off-line to generate the graphs representing the authorized transactions. The minimum information that needs to be available for each executed command is the name of the user, the identification of the session, the identification of the transaction, the command text and the time stamp of the command. This information must be collected in controlled conditions guaranteeing that the system is free of intrusion attempts (which would potentially lead to the identification of malicious transactions as authorized ones). Of course, the DBA must choose the adequate time window that is both representative of database utilization and free of attacks. Inspection of the database audit trail collected during the learning window can be used to assure that no malicious actions have been learned.

After having concluded the learning phase, the detection mechanism is ready to detect intrusions. During this phase the audit trail is used on-line to obtain the sequence of commands executed by each user, which is concurrently compared to the profile of the authorized transactions to identify potential malicious transactions. The detection is done at the SQL command level. That is, it is not necessary to reach the end of the transaction where the suspicious command was found to detect the potential intrusion. All the transactions that have suspicious commands are considered malicious and that triggers several actions depending on the configuration. It can raise an alarm, send a message to the DBA, kill the session, or even activate a damage confinement procedure [6].

3. Preliminary results

In this section we present the experiments conducted to evaluate our intrusion detection mechanism implemented over the audit trail of the Oracle 10g R2 DBMS. The experimental setup includes three computers connected through an Ethernet broadband router/switch. We used two different database application scenarios in this evaluation:

- A well-known database performance benchmark, the TPC-C [7], which provides a controlled database environment quite adequate for initial evaluation of the learning algorithm and for the evaluation of performance overhead, coverage and latency. The mechanism has been evaluated considering both transactions generated as random sequences of commands and malicious transactions submitted by real database users that tried to break the intrusion detection mechanism and change the database content or access data in database tables.
- A real large hospital database sterilization application to assess mainly the transaction learning curve in a real situation. This allows us to obtain the time window necessary to perform a representative learning of the transactions.

The intrusion detection efficiency can be characterized by the following measures: coverage, latency, false

positives, and impact in the performance.

The detection coverage observed for random transactions was 99.23%. The small percentage of undetected transactions corresponds to very small TPC-C transactions whose commands were occasionally mimicked by the random transaction injector. However, in all the cases the intrusion was immediately detected in the following command. In reality, the intrusion detection coverage was 100%, if we consider the sequence of transactions.

As far as the human users are concerned, the intrusions have been detected in all the cases. Four people volunteered to test the system. Three of them are students with basic skills and one is a professional DBA. In 5 sessions, from a total of 132, the users managed to introduce changes in the database, but they were spotted as an intruder in the subsequent command. The intrusion detection latency is consistently low, ranging from 1 second to 1.6 seconds.

The performance penalty in normal load conditions is 6% or less. In heavy load conditions, however, performance overhead increases up to 25%. It is important to note that this is mainly due to the DBMS audit feature being switched on.

Concerning the learning curve using the real database application, we could observe that the rate of transactions learning is very quick at the beginning, reducing over the time. In certain periods of the day or the week we could see that some new transactions were executed. This should be due to some specific procedures that are executed at a very specific moment.

4. Conclusion and future work

This paper presents our research work on intrusion detection in DBMS. The goal is to provide DBMS with on-line intrusion detection capability. Our approach includes two phases: one which is devoted to the learning of transaction profiles and another to detect malicious users. In the learning phase a graph with the sequence of commands that compose each valid transaction is built. In the detection phase the mechanism catches malicious users by detecting the transactions that fall outside the learned profile. When a intruder is detected some corrective actions may be performed (e.g., warning the DBA, killing the malicious session, etc).

An implementation of the proposed mechanism in the Oracle 10g R2 DBMS has been evaluated using the TPC-C benchmark and a production database used by a large hospital. Positive results where obtained, showing that the approach is valuable and can be successfully applied to current DBMS.

We are now studying new algorithms to improve performance, latency and false positives. The detection of ad-hoc queries executed by the DBA, managers and decision making personnel may be addressed by algorithms adapted from those used by system masquerade detectors, like those used by [8,1], where the sequence of commands is not important, but the statistic usage of those commands is. We are also analyzing other forms of collecting the transaction logs from both the outside and the inside the DBMS. The inclusion of the whole system inside an open source DBMS like PostgreSQL is also being considered.

5. References

[1] M. Schonlau, W. DuMouchel, W.-H. Ju, A. F. Karr, M. Theus, Y. Vardi, “Computer intrusion: Detecting masquerades”, Statistical Science, 16(1):58–74, Feb. 2001.

[2] C. Chung, M. Gertz, K. Levitt, “DEMIDS: A Misuse Detection System for Database Systems”, in Proc. of Third Int. IFIP TC-11 WG11.5 Working Conference on Integrity and Internal Control in Information Systems, Kluwer Academic Publishers, 1999.

[3] E. Bertino, A. Kamra, E. Terzi, A. Vakali, “Intrusion detection in RBAC-administered databases”, 21st Annual Computer Security Applications Conference (ACSAC) 2005.

[4] P. Liu, “DAIS: A Real-time Data Attack Isolation System for Commercial Database Applications”, In Proc. of the 17th Annual Comp. Security Applications Conf., 2001.

[5] M. Vieira, H. Madeira, "Detection of malicious transactions in DBMS", The 11th IEEE International Symposium Pacific Rim Dependable Computing, PRDC2005, Changsha, Hunan, China, Dec. 2005.

[6] P. Liu, “DAIS: A Real-time Data Attack Isolation System for Commercial Database Applications”, In Proc. of the 17th Annual Computer Security Applications Conference, 2001.

[7] Transaction Processing Performance Council, “TPC Benchmark C, Standard Specification, Version 5.4”, 2005, available at: http://www.tpc.org/tpcc/.

[8] L. A. Gordon, M. P. Loeb, W. Lucyshyn and R. Richardson, Computer Security Institute, computer crime and security survey, 2005.

[9] R. Maxion, T. Townsend, "Masquerade Detection Using Truncated Command Lines." International Conference on Dependable Systems and Networks, pp. 219-228, Washington, D.C. 23-26 June 2002.

[10] Y. Hu, B. Panda, “Identication of malicious transactions in database systems”, In Proc. of the International Database Engineering and Applications Symposium (IDEAS), 2003.

[11] W. Lup Low, J. Lee, P. Teoh, “DIDAFIT: Detecting Intrusions in Databases Through Fingerprinting Transactions”, International Conference on Enterprise Information Systems, 2002.

[12] T. Lunt, C. McCollum. “Intrusion detection and response research at DARPA”, Technical report, The MITRE Corporation, McLean, VA, 1998.