Monitoring Database Application Behavior for Intrusion Detection

José Fonseca
CISUC - Polithecnic Institute of Guarda
6300 Guarda
Portugal
josefonseca@ipg.pt

Marco Vieira, Henrique Madeira
DEI/CISUC - University of Coimbra
3030 Coimbra
Portugal
{mvieira, henrique}@dei.uc.pt

Abstract

Database management systems (DBMS) represent the ultimate layer in preventing malicious data access or corruption and implement several security mechanisms to protect data. However these mechanisms cannot always stop malicious users from accessing data by exploiting system vulnerabilities. The aim of this paper is to propose an intrusion detection mechanism for DBMS to fill this gap. Our approach consists of a comprehensive representation of user database utilization profiles to perform concurrent intrusion detection. Prior to the detection it is necessary to define and learn these utilization profiles. Profiles are defined using a three level abstraction and learned directly from monitoring the database utilization in real conditions. The proposed mechanism is generic and can be easily implemented in commercial and open-source DBMS.

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 the organization 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.

This paper proposes a new intrusion detection mechanism that adds concurrent intrusion detection to DBMS using a comprehensive set of behavior abstractions representing database activity.

2. Proposed approach

The proposed mechanism is based on anomaly detection and includes a learning phase and a detection phase. Very briefly, the database utilization profile is gathered as a first step to feed the learning phase. Once the database utilization profile is established, the information collected is used to concurrently detect database intrusions. The mechanism is able to detect and stop database attacks while calling the database administrator (DBA) attention (e.g., by sending an email or an SMS message, etc.).

We use three abstraction levels to define the user profile representing his/her database activity: command level, transaction level, and session level. The intrusion detection is based on a set of security constraints defined at each of these three levels. A database access that violates any security constraints defined at any level is considered a potential intrusion:

- Command level: checks if the structure of each executed command belongs to the set of command structures previously learned.
- Transaction level: checks if the command is in the right place inside the transaction profile (a transaction is a unit formed by a set of SQL commands always executed in the same sequence).
- Session level: checks if the transaction fits in a known transaction sequence. It represents the sequence of operations that the user executes in a session.

Intrusion detection activity starts at the lowest level

(command level). If no intrusion is detected at this level, the detection continues at the next level (transaction level), and so on. If no restriction is violated after having passed all levels, the command is considered valid.

Different implementation alternatives for data capturing can be followed, including external database approaches, such as using a proxy or a sniffer, or taking advantage of database auditing features available in most of DBMS (the audit approach was used by Vieira et al [3], with quite good results). The detection mechanism can also be implemented inside the DBMS (although this requires the DBMS modification).

3. Preliminary results

In this section we present the experiments conducted to evaluate our intrusion detection mechanism. The sniffer approach has been used as it has no impact on the server performance (it is located in a different computer). It is also a non intrusive technique which turns out to be the best approach to use with real database scenarios. For the DBMS we have chosen the Oracle 10g since it is one of the most representative in the market. 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-W, which provides a controlled database environment quite adequate for initial evaluation of the learning algorithm and for the evaluation of coverage and latency. After a learning period the mechanism has been evaluated considering correct commands executed randomly and correct commands slightly changed.
- 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.

In the first step we have run TPC-W for 51 minutes to learn the transactions. We have then executed TPC-W intensively in order to assess if the all the transactions were successfully. We have observed that only one valid TPC-W transaction was detected as malicious. The injection of slightly changed leaded to a coverage of 100%, while de injection of correct SQL commands in a random order (i.e., invalid transactions) leaded to 4.2% of false negatives.

In another experiment we have manually injected 50 malicious commands while the TPC-W was running to produce the load. All these 50 commands were detected as intrusions. The largest latency time obtained was 10 milliseconds and the average latency time was 1.6 milliseconds. In all the commands the detection was done before the server reply to the malicious command, which makes the detection, was made before the next command after the malicious command. The query was very simple, which means it was quickly executed by the server.

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

In this paper we present a new mechanism for the detection of malicious transactions in DBMS. It uses a representation of the profile of the user utilization of the database using three levels of detail (command, transaction and session) to detect the intrusions. The database intrusion detection mechanism consists of two main phases: transaction learning and concurrent intrusion detection. This mechanism is generic as it can be used in any typical DBMS, including stat-of-the-art commercial DBMS. A set of experiments were performed using one implementation of the mechanism based on the sniffer approach (one of the proposed alternatives). The experimental results show that the proposed mechanism is quite effective and can be easily implemented. For future work we intend to develop other architectures presented and compare their results. Another objective is to include a damage containment module to the intrusion detection mechanism.

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] E. Bertino, et. al., “Intrusion detection in RBAC-administered databases”, *ACSAC,* 2005.

[3] M. Vieira, H. Madeira, "Detection of malicious transactions in DBMS", *PRDC*, China, 2005.