Mechanisms for Database Intrusion Detection and Response

Ashish Kamra
Electrical and Computer Engineering
Purdue University
akamra@purdue.edu

Co-Advisor: Elisa Bertino
Computer Science
Purdue University
bertino@cs.purdue.edu

Co-Advisor: Guy Lebanon
Electrical and Computer Engineering
Purdue University
lebanon@ece.purdue.edu

ABSTRACT

Data represent today a valuable asset for companies and organizations and must be protected. Most of an organization's sensitive and proprietary data resides in a Database Management System (DBMS). The focus of this thesis is to develop advanced security solutions for protecting the data residing in a DBMS. Our strategy is to develop an Intrusion Detection (ID) mechanism, implemented within the database server, that is capable of detecting anomalous user requests to a DBMS. The key idea is to learn profiles of users and applications interacting with a database. A database request that deviates from these profiles is then termed as anomalous. A major component of this work involves prototype implementation of this ID mechanism in the PostgreSQL database server. We also propose to augment the ID mechanism with an Intrusion Response engine that is capable of issuing an appropriate response to an anomalous database request.

1. INTRODUCTION

Data represent today an important asset for companies and organizations. Some of these data are worth millions of dollars and organizations take great care at controlling access to these data, with respect to both internal users, within the organization, and external users, outside the organization. Recently, we have seen an interest in database activity monitoring solutions that continuously monitor a Database Management System (DBMS) and report any relevant suspicious activity [8]. This step-up in data vigilance on part of the organizations is partly driven by various government regulations concerning data management such as SOX, PCI, GLBA, HIPAA and so forth [20]. Organizations have also come to realize that current attack techniques are more sophisticated, organized and targeted than the broad-based hacking days of past. Also, with greater data integration, aggregation and disclosure, preventing data theft has become a big challenge, both from outside and within. Standard database security controls such as access control mechanisms, authentication, encryption technologies and so forth are not of much help when it comes to preventing data theft from insiders. The above mentioned threat scenarios have forced organizations to re-evaluate security strategies for their internal DBMSs [20]. So what new strategies can be adopted to develop and deploy a DBMS with high-assurance security (in all its flavors)? We see two distinct yet complimentary ways of achieving this goal. The first strategy is to monitor database activity, using third-party products, to detect any potential abnormal behavior. Gartner research has identified Database Activity Monitoring (DAM) as one of the top five strategies to dramatically limit data loss information leaks for organizations [19, 21]. Moreover, there are already several products on the market today that try to address these concerns of organizations [8]. The second strategy is to revise architectures and techniques adopted by a traditional DBMS [3]. An important component of this new generation security-aware DBMS is an Intrusion Detection (ID) mechanism built as an integral part of a DBMS [3]. However, despite the fact that building ID systems for networks and operating systems has been an active area of research, few ID systems exist that are specifically tailored to a DBMS. The goal of this work is to develop efficient and effective algorithms for detecting intrusive user behavior in context of a DBMS, and augment that with an intrusion response engine. We next present the problem statement for this thesis.

1.1 Problem Statement

The goal of this thesis is to develop architectures, mechanisms and algorithms for a DBMS equipped with activity monitoring, intrusion detection and response capabilities. Within this broad context, the research issues that we address are as follows:

1. Creating profiles that succinctly represent user/application-behavior interacting with a DBMS.
2. Developing efficient algorithms for online detection of anomalous database user/application behavior.
3. Developing strategies for responding to intrusions in context of a DBMS.
4. Creating a system architecture for database intrusion detection and intrusion response as an integral component of a DBMS, and a prototype implementation of the same in PostgreSQL relational database [1].

Proceedings of the Second SIGMOD PhD Workshop on Innovative Database Research (IDAR 2008), June 13, 2008, Vancouver, Canada.

1.2 Paper Roadmap

The rest of the paper is organized as follows. The next section presents some of our existing work in detecting anomalous database requests. We then discuss in detail the related work in this area. We end with some preliminary ideas for future work.

2. OUR APPROACH

We discuss the various pieces of our approach in this section. We start off with a discussion of the system architecture.

2.1 System Architecture

The system's architecture consists of three main components: the conventional DBMS mechanism that handles the query execution process, the database audit log files and the ID mechanism. These components form the new extended DBMS that is enhanced with an independent ID system operating at the database level. The flow of interactions for the ID process is shown in Figure 1. Every time a query is issued, it is analyzed by the ID mechanism before execution. First, the feature selector extracts features from the raw user query and converts it into one of the forms supported by our ID mechanism [11]. The detection engine then checks the features against the existing profiles and submits its assessment of the query to the response engine. The response engine consults a policy base of existing response mechanisms to issue a response depending on the assessment of the query submitted by the detection engine. Notice that the fact that a query is anomalous may not necessarily imply an intrusion. Other information and security policies must also be taken into account. For example, if the user logged under the role is performing some special activities to manage an emergency, the ID mechanism may be instructed not to raise alarms in such circumstances. If the response engine decides to raise an alarm, certain actions for handling the alarm can be taken. The most common action is to send an alert to the security administrator. However other actions are possible (Figure 1), such as disable the role and disconnect the user making the access or drop the query. If by assessment, the query is not anomalous, the response engine simply updates the database audit log and the profiles with the query information. Before the detection phase, the profile creator module creates the initial profiles from a set of intrusion free records from the database audit log.

We next describe some of our current work in detecting database anomalies.

2.2 Detecting Anomalous Database Requests

In [11], we have described algorithms for detecting anomalous user/role access to a DBMS. In order to identify normal behavior, we use the database audit files for extracting information regarding users' actions. The audit records, after being processed, are used to form initial profiles representing acceptable actions. Each entry in the audit file is represented as a separate data unit; these units are then combined to form the desired profiles. We consider two different scenarios while addressing this problem. In the first scenario, we assume that the database has a Role Based Access Control (RBAC) model in place. Under a RBAC system, permissions are associated with roles, grouping several users, rather than with single users. Our ID system is able to determine role intruders, that is, individuals that while holding a specific role, behave differently than expected. An important advantage of providing an ID technique specifically tailored to RBAC databases is that it can help in protecting against insider threats. Furthermore, the existence of roles makes our approach usable even for databases with large user population. When role information does exist, the profile learning problem is transformed into a supervised learning problem. The classification engine used by us is the naive bayes classifier. A profile for a naive bayes classifier consists of the table and column access probabilities for a specific role. The roles are treated as classes for the classification purpose. The ID task for this setting is as follows: For every new user request, its role (or class) is predicted by the trained classifier. If the predicted role (or class) is different from the original role associated with the query, an anomaly is detected. For benign queries, the classifier can be updated in a straightforward manner by adjusting the probabilities of the relevant features in the user request.

In the second case, we address the same problem in the context of DBMS without any role definitions. This is a necessary case to consider because not all organizations are expected to follow a RBAC model for authorizing users of their databases. In such a setting, every transaction is associated with the user that issued it. A naive approach for ID in this setting would be to build a different profile for every user. For systems with large user bases such an approach would be extremely inefficient. Moreover, many of the users in those systems are not particularly active and they only occasionally submit queries to the database. In the case of highly active users, profiles would suffer from over-fitting, and in the case of inactive users, they would be too general. In the first case we would observe a high number of false alarms, while the second case would result in high number of missed alarms, that is, alarms that should have been raised. We overcome these difficulties by building user-group profiles (clusters of similar behaviors) based solely on the transactions users submit to the database. Given such profiles, we define an *anomaly* as an access pattern that deviates from the profiles. The specific methodology that we use for the ID task is as follows: we partition the training data into clusters[1] using standard clustering techniques. We maintain a mapping for every 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, we outline two approaches. In the first approach, we apply the naive bayes classifier 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.

We have performed experiments on both synthetic and real-world data sets. For generating synthetic SQL queries, the database object access pattern of such queries was modeled using a zipf probability distribution. Our experiments

[1]In this setting, the clusters obtained after the clustering process represent the profiles.

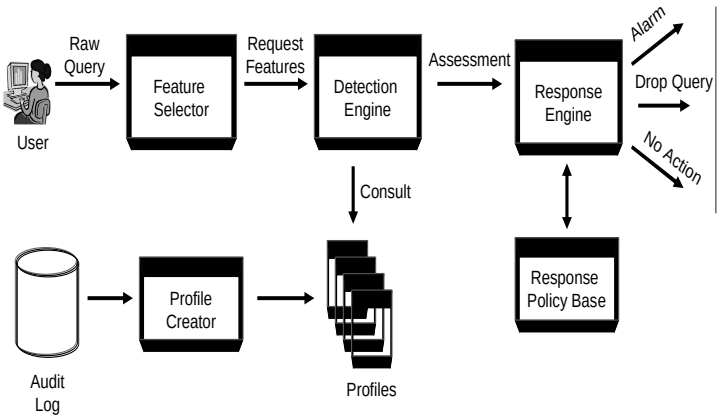

Figure 1: System Architecture

showed that the detection accuracy of our methods improved as the access pattern of the users/roles became more differentiated. For the real data set, the experiments for the supervised case demonstrated a false negative rate of less than 3% and a false positive rate of approx. 17%. More details on these techniques and experiments can be found in [11].

2.3 Detecting SQL Injection Attacks

SQL Injection is an attack exploiting applications that construct SQL statements from user-supplied input. When an application fails to properly validate user-supplied input, it is possible for an attacker to alter the construction of backend SQL statements. Several threat scenarios may arise because of these 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. An attacker may thus abuse the privileges of the program in a manner unintended by the application program designers.

In [5], we propose a framework based on anomaly detection techniques to detect malicious behavior of database application programs. Our approach is as follows: We create profiles of application programs that can succinctly represent their normal behavior in terms of SQL queries submitted by these programs to the database. Query traces from database logs can be used for this purpose. We then use an anomaly detection model based on data mining techniques to detect behavior deviating from normal. The anomalous behavior that we focus on in this paper is behavior related to SQL Injection attacks. SQL Injection, traditionally, has been considered as an application level vulnerability and solutions have been proposed at that level [9]. However, even though the mechanism of a SQL Injection attack is through an application, the resource the security of which is directly threatened is the database. Therefore, we propose a solution to enhance the traditional security mechanisms of a DBMS to protect against such attacks. The essential idea motivating our approach is that SQL Injection can be modeled as an anomaly detection problem. In a typical database application, the input supplied by the users is used to construct the *where* clause of queries at run time. An SQL Injection attack typically involves malicious modifications to this input either by adding additional clauses or by changing the structure of an existing clause. The *projection* clause of the query, however, is not modified and remains static because it is not constructed at run time. Driven by this observation, we use association rule mining techniques to derive rules that represent the associative relationships among the various query attributes. We derive two sets of rules specific to our task. The first set consists of rules binding the *projection* attributes of the query to the attributes used in the *where* clause. The second set of rules represent the relationship among the attributes in the *where* clause of the query. These two sets of rules together form the profile of an application. To detect an attack query, we check if the relationship among its query attributes can be inferred by the set of rules in the application profile. If not, our system raises an alarm. In summary, our contributions in this work are as follows:

1. We propose novel encoding schemes for SQL queries to extract useful information from them.
2. We present an approach for fingerprinting database applications based on the SQL queries submitted by them to a database.
3. We then present an anomaly detection model to detect anomalous behavior of database applications.
4. We further demonstrate how this model can be used to detect SQL Injection at the database level.

For this work, we have performed experiments on synthetically generated SQL queries. The experiments were performed to show the performance of our methods by varying the minimum support (for the association rules) and detection theshold parameters. As value of minimum support was increased, we observed an increase in the false positive rate. More details on this technique and experimental results can be found in [5].

3. RELATED WORK

We first present an categorization of the database activity monitoring products available in the market today. We follow that up by describing existing work in database intrusion detection. Finally, we give an overview of related work in detecting SQL Injection attacks.

3.1 Database Activity Monitoring

As we mentioned earlier in section 1, there are several database activity monitoring products on the market today [8]. We categorize them into two broad categories : *network-appliance-based* and *agent-based*. Network-appliance-based solutions consist of a dedicated hardware appliance that taps into an organization's network, and monitors network traffic to and from the data center. These appliances can either be operated in a *passive* mode in which they observe the traffic and report suspicious behavior, and an *inline* mode where all traffic to the data center passes through them. Agent-based solutions, on the other hand, have a software component installed on the database server that interacts with the DBMS in order to monitor them. Each method has its own advantages and drawbacks. A passive network-appliance does not increase network latency but its ability to block an actual attack is limited [8]. Moreover, network appliances in general, are handicapped by their inability to monitor privileged users who can log into the database server directly. Agent-based solutions, on the other hand, impose extra overhead of an additional software running on the database server and using CPU cycles. But communicating directly with the database server also gives them a better view of the database structure thus enabling them to monitor the server usage in a more effective manner [8].

3.2 Database Intrusion Detection

Several approaches dealing with ID for operating systems and networks have been developed [4, 2, 13, 18, 23]. However, they are not adequate for protecting databases. The reason is that actions deemed malicious for a database application are not necessarily malicious for the network or the operating system; thus ID systems specifically designed for the latter would not be effective for database protection.

An abstract and high-level architecture of a DBMS incorporating an ID component has been recently proposed [17]. However, this work mainly focuses on discussing generic solutions rather than proposing concrete algorithmic approaches. Similar in spirit is the work of Shu et al. [26] who have developed an architecture for securing web-based database systems without proposing any specific ID mechanisms. Finally, in [16] a method for ID is described which is applicable only to real-time applications, such as a programmed stock trading that interacts with a database. The key idea pursued in this work is to exploit the real-time properties of data for performing the ID task.

Anomaly detection techniques for detecting attacks on web applications have been discussed by Vigna et al. [12]. A learning based approach to the detection of SQL attacks is proposed by Valeur et al. [24]. The motivation of this work is similar to ours as in the use of machine learning techniques to detect SQL based attacks on databases. Their methodologies, however, focus on detection of attacks against back-end databases used by web-based applications. Thus, their ID architecture and algorithms are tailored for that context. We, on the other hand, propose a general purpose approach towards detection of anomalous access patterns in a database as represented by SQL queries submitted to the database.

An anomaly detection system for relational databases is proposed by Spalka et al. [22]. This work focuses on detecting anomalies in a particular database state that is represented by the data in the relations. Their first technique uses basic statistical functions to compare reference values for relation attributes being monitored for anomaly detection. The second technique introduces the concept of Δ relations that record the history of changes of data values of monitored attributes between two runs of the anomaly detection system. This work complements our work as it focuses on the semantic aspects of the SQL queries by detecting anomalous database states as represented by the data in the relations, while we focus on the syntactic aspects by detecting anomalous access patterns in a DBMS.

Another relevant approach towards a database-specific ID mechanism is by Hu et al. [10]. They propose mechanisms for finding data dependency relationships among transactions and use this information to find hidden anomalies in the database log. The rationale of their approach is the following: if a data item is updated, this update does not happen alone but is accompanied by a set of other events that are also logged in the database log files. For example, due to an update of a given data item, other data items may also be read or written. Therefore, each item update is characterized by three sets: the *read set*, the set of items that have been read because of the update; the *pre-write set*, the set of items that have been written before the update but as consequence of it; and the *post-write set*, the set of items that have been written after the update and as consequence of it. Such approach identifies malicious transactions by comparing those sets for different item updates.

An approach which is conceptually most similar to ours is the one underlying the DEMIDS system [7]. DEMIDS is a misuse-detection system, tailored for relational database systems. It uses audit log data to derive profiles describing typical patterns of accesses by database users. Essential to such an approach is the assumption that the access pattern of users typically forms a working scope which comprises sets of attributes that are usually referenced together with some values. DEMIDS assumes domain knowledge about the data structures and semantics encoded in a given database schema. Distance measures are then used to guide the search for frequent item-sets describing the working scope of users. The drawback of such an approach is that the number of users for a database system can be quite large and maintaining (or updating) profiles for such large number of users is not a trivial task. Moreover, the approach used by DEMIDS to build user profiles assumes domain knowledge about a given database schema. This can adversely affect the general applicability of the method. Our approach, on the other hand, builds profiles using syntactic information from SQL queries appearing in the database log, which makes our approach more general than others.

Lee et al. [14] present an approach for detecting illegitimate database accesses by fingerprinting transactions. The main contribution of this work is a technique to summarize SQL statements into compact regular expression fingerprints. The system detects an intrusion by matching new SQL statements against a known set of legitimate database transaction fingerprints. In this respect, this work can be classified as a signature-based ID system which is conceptually different from the learning-based approach that we propose in this thesis.

In addition to the above approaches, our previous work on query floods [6] can also be characterized as a DBMS-specific ID mechanism. However, in that work we have fo-

cused on identifying specific types of intruders, namely those that cause query-flood attacks. A user can engineer such an attack by "flooding" the database with queries that can exhaust a DBMS's resources making it incapable of serving legitimate users.

3.3 SQL Injection Attacks

To date, there have been few approaches that apply anomaly detection techniques to identify anomalous database application behavior. One such technique is by Valeur et al. [25]. It builds a number of different statistical query models using a set of typical application queries, and then intercepts the new queries submitted to the databases to check for anomalous behavior. Our work is similar in spirit to their work but uses association rule mining techniques to build application query profiles. Another work by Lee et al. [15] considers the use of learning techniques for identifying web-based attacks on databases. However, their methods for learning query fingerprints are fundamentally different from ours. Apart from the above two learning based techniques, there have been other approaches proposed in the literature that use application code analysis to detect SQL Injection attacks. We direct the reader to [9] for an in-depth survey of these approaches.

4. FUTURE WORK

In this section we discuss some preliminary ideas for future work in this thesis.

4.1 Intrusion Response

Intrusion detection for a DBMS also invokes another interesting research question. Once a database request is detected as malicious by an ID mechanism, how can a response be issued for it? Standard security violations in the context of a DBMS can be dealt with simple and intuitive responses. For example, if a user tries to exercise an unassigned privilege, the access control mechanism denies the access request. However, in order to respond to anomalous database requests, the response mechanisms are not trivial to define. The response can either be *conservative* and only raise an alarm but let the anomalous request proceed. A more *aggressive* approach would be to drop the request along with raising an alarm. In most situations, a hybrid approach may be necessary. The key observation is that the ID mechanism should have the capability to decide which response action to take under a given situation. We envisage future ID systems to be augmented with an intrusion response mechanism that is responsible for providing the end-user with a framework for defining appropriate response actions to intrusive behavior. As part of our future work, we propose to design and implement an intrusion response mechanism within the context of a DBMS.

4.2 Prototype Implementation

A major component of this thesis is to illustrate the feasibility and efficiency of our methods. For this purpose, we intend to implement a prototype intrusion detection mechanism integrated with the PostgreSQL database server. The main objectives of this implementation are as follows :

1. To illustrate efficient implementation strategies for developing an intrusion detection and response mechanism inside a DBMS.
2. To measure the overhead on normal query processing introduced by the intrusion detection and response mechanism.

5. REFERENCES

[1] Postgresql 8.3. http://www.postgresql.org/.

[2] K. H. A. Hoglund and A. Sorvari. A computer host-based user anomaly detection using the self-organizing map. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN)*, 2000.

[3] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB)*, pages 143–154. Morgan-Kaufmann, 2002.

[4] S. Axelsson. Intrusion detection systems: A survey and taxonomy. Technical Report 99-15, Chalmers Univ., mar 2000.

[5] E. Bertino, A. Kamra, and J. Early. Profiling database application to detect sql injection attacks. *IEEE International Performance, Computing, and Communications Conference (IPCCC) 2007*, pages 449–458, April 2007.

[6] E. Bertino, T. Leggieri, and E. Terzi. Securing dbms: Characterizing and detecting query floods. In *Proceedings of the International Security Conference (ISC)*, 2004.

[7] C. Chung, M. Gertz, and K. Levitt. Demids: a misuse detection system for database systems. In *Integrity and Internal Control in Information Systems: Strategic Views on the Need for Control. IFIP TC11 WG11.5 Third Working Conference*, 2000.

[8] A. Conry-Murray. The threat from within. Network Computing (Aug 2005), http://www.networkcomputing.com/showArticle.jhtml?articleID=166400792.

[9] W. G. Halfond, J. Viegas, and A. Orso. A Classification of SQL-Injection Attacks and Countermeasures. In *Proceedings of the International Symposium on Secure Software Engineering (ISSSE)*, 2006.

[10] Y. Hu and B. Panda. Identification of malicious transactions in database systems. In *Proceedings of the International Database Engineering and Applications Symposium (IDEAS)*, 2003.

[11] A. Kamra, E. Bertino, and E. Terzi. Detecting anomalous access patterns in relational databases. *The International Journal on Very Large Data Bases (VLDB)*, 2008.

[12] C. Kruegel and G. Vigna. Anomaly detection of web-based attacks. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2003.

[13] T. Lane and C. E. Brodley. Temporal sequence learning and data reduction for anomaly detection. *ACM Transactions on Information and System Security (TISSEC)*, 2(3):295–331, 1999.

[14] S. Y. Lee, W. L. Low, and P. Y. Wong. Learning fingerprints for a database intrusion detection system. In *ESORICS '02: Proceedings of the 7th European Symposium on Research in Computer Security*, pages 264–280, London, UK, 2002. Springer-Verlag.

[15] S. Y. Lee, W. L. Low, and P. Y. Wong. Learning fingerprints for a database intrusion detection system. In *Proceedings of the 7th European Symposium on Research in Computer Security (ESORICS)*, 2002.

[16] V. Lee, J. Stankovic, and S. Son. Intrusion detection in real-time databases via time signatures. In *Proceedings of the IEEE Real-Time Technology and Applications Symposium (RTAS)*, 2000.

[17] P. Liu. Architectures for intrusion tolerant database systems. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2002.

[18] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. Neumann, H. Javitz, A. Valdes, and T. Garvey. A real - time intrusion detection expert system (ides) - final technical report. *Technical Report, Computer Science Laboratory, SRI International*, 1992.

[19] R. Mogull. Top five steps to prevent data loss and information leaks. Gartner Research (July 2006), "http://www.gartner.com/".

[20] R. B. Natan. *Implementing Database Security and Auditing*. Digital Press, 2005.

[21] M. Nicolett and J. Wheatman. Dam technology provides monitoring and analytics with less overhead. Gartner Research (Nov 2007), "http://www.gartner.com/".

[22] A. Spalka and J. Lehnhardt. A comprehensive approach to anomaly detection in relational databases. In *DBSec*, pages 207–221, 2005.

[23] R. Talpade, G. Kim, and S. Khurana. Nomad: Traffic-based network monitoring framework for anomaly detection. In *Proceedings of the 4th IEEE Symposium on Computers and Communications (ISCC)*, 1998.

[24] F. Valeur, D. Mutz, and G. Vigna. A learning-based approach to the detection of sql attacks. In *Proceedings of the International Conference on detection of intrusions and malware, and vulnerability assessment (DIMVA)*, 2003.

[25] F. Valeur, D. Mutz, and G. Vigna. A learning-based approach to the detection of sql injection attacks. In *Proceedings of the Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, 2005.

[26] S. Wenhui and T. Tan. A novel intrusion detection system model for securing web-based database systems. In *Proceedings of the 25th Annual International Computer Software and Applications Conference (COMPSAC)*, 2001.