The VLDB Journal (2008) 17:1063–1077
DOI 10.1007/s00778-007-0051-4

REGULAR PAPER

Detecting anomalous access patterns in relational databases

Ashish Kamra · Evimaria Terzi · Elisa Bertino

Received: 7 April 2006 / Accepted: 2 April 2007 / Published online: 17 May 2007

Abstract A considerable effort has been recently devoted to the development of Database Management Systems (DBMS) which guarantee high assurance and security. An important component of any strong security solution is represented by Intrusion Detection (ID) techniques, able to detect anomalous behavior of applications and users. To date, however, there have been few ID mechanisms proposed which are specifically tailored to function within the DBMS. In this paper, we propose such a mechanism. Our approach is based on mining SQL queries stored in database audit log files. The result of the mining process is used to form profiles that can model normal database access behavior and identify intruders. We consider two different scenarios while addressing the problem. In the first case, 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 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. In the second scenario, we assume that there are no roles associated with users of the database. In this case, we look directly at the behavior of the users. We employ clustering algorithms to form concise profiles representing normal user behavior. For detection, we either use these clustered profiles as the roles or employ outlier detection techniques to identify behavior that deviates from the profiles. Our preliminary experimental evaluation on both real and synthetic database traces shows that our methods work well in practical situations.

This material is based upon work supported by the National Science Foundation under Grant No. 0430274 and the sponsors of CERIAS.

A. Kamra (✉) · E. Bertino
Purdue University and CERIAS, West Lafayette, IN, USA
e-mail: akamra@ecn.purdue.edu

E. Bertino
e-mail: bertino@cs.purdue.edu

E. Terzi
University of Helsinki and HIIT, Helsinki, Finland
e-mail: evimaria.terzi@cs.helsinki.fi

Keywords Anomaly detection · Intrusion detection · User profiles · DBMS · RBAC

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 controlling access to these data, with respect to both internal users, within the organization, and external users, outside the organization. Data security is also crucial when addressing issues related to privacy of data pertaining to individuals; companies and organizations managing such data need to provide strong guarantees about the confidentiality of these data in order to comply with legal regulations and policies [2]. Overall, data security has a central role in the larger context of information systems security. Therefore, the development of Database Management Systems (DBMS) with high-assurance security (in all its flavors) is a central research issue. The development of such DBMS requires a revision of architectures and techniques adopted by traditional DBMS [1]. An important component of this new generation security-aware DBMS is an Intrusion Detection (ID) mechanism. Even though DBMS provide access control mechanisms, these mechanisms alone are not enough to guarantee data security. They need to be complemented by suitable ID mechanisms; the use of such mechanisms is crucial for protecting against impersonation attacks and against malicious code embedded in application programs. Also ID mechanisms may help in addressing the problem of insider threats, an increasingly important problem in today's organizations for which not many solutions have been devised. 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 DBMS.

The goal of this study is to address such needs by investigating the development of a DBMS-specific ID system. There are two main reasons that motivate the necessity of such ID systems. The first 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. The second, and more relevant motivation, is that ID systems designed for networks and operating systems are not adequate to protect databases against insider threats, which is an important issue when dealing with privacy. These threats are much more difficult to defend against, because they are from subjects that are legitimate users of the system, and thus may have access rights to data and resources.

An additional point that needs to be clarified is that we do not provide a formal definition of the security that a DBMS-specific ID mechanism should guarantee. However, we do not think that such a definition is necessary. More specifically, an ID system that operates at the DBMS level is not responsible for making sure that certain security requirements are guaranteed. This is a responsibility of other components in the security infrastructure. The goal of the ID system is to identify unexpected access patterns by authorized users (and applications) and report them to the interested parties, such as the DataBase Administrator (DBA) or the Site Security Officer (SSO). Such suspicious behaviors may be indicative of organized attacks by authorized users (insider threats), or in some cases may be useful for further refining the initial security requirements of the system.

1.1 Methodology

The key idea underlying our approach is to build profiles of normal user behavior interacting with a database. We then use these profiles to detect *anomalous* behavior. In this context, our approach considers two different application scenarios. In the first case, the approach we follow is similar to the one followed by Bertino et al.[4].[1] We assume that the database has a Role Based Access Control (RBAC) model in place. Authorizations are specified with respect to roles and not with respect to individual users. One or more roles are assigned to each user and privileges are assigned to roles. Our ID system builds a profile for each role and is able to determine role intruders, that is, individuals while holding a specific role deviate from the normal behavior of that role. The use of roles makes our approach usable even for databases with a large user population. Managing a few roles is much more efficient than managing many individual users. With respect to ID, using roles means that the number of profiles to build and maintain is much smaller than those one would need when considering individual users. Note that RBAC has been standardized (see the NIST model [23]) and has been adopted in various commercial DBMS products. This implies that an ID solution, based on RBAC, could be easily deployed in practice.

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, where as the second case would result in high number of missed alarms, that is, alarms that should have been raised. We overcame 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 contribution of this paper is an ID solution specifically tailored for database systems and in this context, the two problems that we address are as follows: how to build and maintain profiles representing accurate and consistent user behavior; how to use these profiles for performing the ID task at hand. The solution to both problems relies on the use of 'intrusion free' database traces, that is, sequences of database audit log records representing normal user behavior.[2] However, the information contained in these traces differ

[1] In this paper, however, we enhance the representation of SQL commands to also include information from the query predicates.

[2] Guarantying the intrusion-free nature of the training data is an issue often raised in the context of anomaly detection systems. The standard technique employed to address this concern is to use outlier detection algorithms to remove potential anomalies from the training data. Although this does not guarantee that all malicious SQL statements are removed from the training data or that every outlying point that is removed is malicious; in practice, this step has often been observed to increase the accuracy of anomaly detection systems. In this work, however, we do not employ such strategy for our experiments.

depending on the application scenario in question. When role information does exist, the problem is transformed into a supervised learning problem. A classifier is trained using a set of intrusion-free training records. This classifier is then used for detecting anomalous behavior. For example, if a user claims to have a specific role while the classifier classifies her behavior as indicative of another role, then an alarm is raised. On the other hand, for the case in which no role information is available, we form our solution based on unsupervised learning techniques. We employ clustering algorithms to construct clusters of users that behave in a similar manner (with respect to their database access patterns). These clusters may also help the DBA in deciding which roles to define. For every user, we maintain the mapping to its representative cluster. For the ID phase, we specify two different approaches. In the first approach, we treat the problem in a manner similar to the supervised case with the clusters as the classifier classes. In the second approach, we treat the detection phase as an outlier detection problem. That is, an alarm is raised for a new query if it is marked as an outlier with respect to its representative cluster.

The main challenge in attacking our problem is to extract the right information from the database traces so that accurate profiles can be built. To address this problem, we propose several representations for the database log records, characterized by different granularity and, correspondingly, by different accuracy levels. By using those representations, we then address the first scenario as a classification problem and the second one as a clustering problem.

1.2 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 Fig. 1. Every time a query is issued, it is analyzed by the ID mechanism before execution. First, the feature selector converts the raw SQL query into one of the quiplet forms supported by our ID mechanism (see Sect. 2). The detection engine then checks the quiplet against the existing profiles and submits its assessment of the query (anomalous vs. not anomalous) 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 and Lehnhardt 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 (Fig. 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.

1.3 Related work

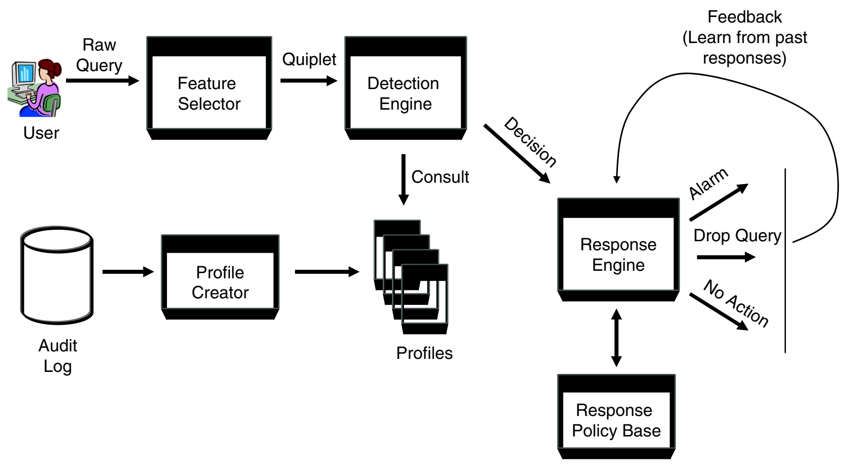
Several approaches dealing with ID for operating systems and networks have been developed [3,12,16,21,25]. However, as we have already argued, they are not adequate for protecting databases.

An abstract and high-level architecture of a DBMS incorporating an ID component has been recently proposed [20]. However, this work mainly focuses on discussing generic solutions rather than proposing concrete algorithmic approaches. Similar in spirit is the work of Wenhui et al. [27] who have developed an architecture for securing web-based database systems without proposing any specific ID mechanisms. Finally, in [19] 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 Kruegel et al. [15]. A learning-based approach to the detection of SQL attacks is proposed by Valeur et al. [26]. 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 [24]. 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 focusses on the semantic aspects

Fig. 1 Overview of the ID process



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 and Pomda. [13]. 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 a consequence of it; and the *post-write set*, the set of items that have been written after the update and as a 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 [6]. 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. [18] 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 paper.

In addition to the above approaches, our previous work on query floods [5] can also be characterized as a DBMS-specific ID mechanism. However, in that work we have focused 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 DBMS's resources making it incapable of serving legitimate users.

Finally, this paper extends our earlier work [4] in two new directions. We enhance the representation of SQL queries by extracting information from the query predicates. This is useful in detecting anomalies where attributes in the query predicate are modified without touching the projected attributes. Moreover, our earlier work only considered the case in which role information is available in the database audit logs. In that setting the problem of ID was reduced to a supervised learning problem. Here, we also consider the case in which role information is not available in the database logs. This is a significant extension because it makes our techniques applicable to settings which do not use a RBAC model for access control.

1.4 Paper roadmap

The paper is organized as follows. Next section describes the audit log record formats and the three different representation levels supported by our approach. Section 3 describes in detail the role-based anomaly detection approach and reports the related experimental results. Section 4 explains the unsupervised anomaly detection setting and reports the related experimental results. We conclude the paper by discussing future work.

2 Data representation

In order to identify user 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 assume that users interact with the database through commands, where each command is a different entry in the log file, structured according to the SQL language. For example, in the case of *select* queries such commands have the format:

```
SELECT [DISTINCT] {TARGET-LIST}
FROM              {RELATION-LIST}
WHERE             {QUALIFICATION}
```

In order to build profiles, we need to pre-process the log-file entries and convert them into a format that can be analyzed by our algorithms. Therefore, we represent each entry by a basic data unit that contains five fields, and thus it is called a *quiplet*.

Quiplets are our basic unit for viewing the log files and are the basic components for forming profiles. User actions are characterized using sets of such quiplets. Each quiplet contains the following information: the SQL command issued by the user, the set of relations accessed, and for each such relation, the set of referenced attributes. This information is available in the three basic components of the SQL query, namely, the `SQL COMMAND`, the `TARGET-LIST` and the `RELATION-LIST`. We also process the `QUALIFICATION` component of the query to extract information on relations and their corresponding attributes, that are used in the query predicate.[3] Therefore, the abstract form of such a quiplet consists of five fields (*SQL Command, Projection Relation Information, Projection Attribute Information, Selection Relation Information and Selection Attribute Information*).[4] For the sake of simplicity we represent a generic quiplet using a 5-ary relation $Q(c, \mathcal{P}_{\mathcal{R}}, \mathcal{P}_{\mathcal{A}}, \mathcal{S}_{\mathcal{R}}, \mathcal{S}_{\mathcal{A}})$, where c corresponds to the command, $\mathcal{P}_{\mathcal{R}}$ to the projection relation information, $\mathcal{P}_{\mathcal{A}}$ to the projection attribute information, $\mathcal{S}_{\mathcal{R}}$ to the selection relation information, and $\mathcal{S}_{\mathcal{A}}$ to the selection attribute information. Depending on the type of quiplet the two arguments $\mathcal{P}_{\mathcal{R}}$(or $\mathcal{S}_{\mathcal{R}}$) and $\mathcal{P}_{\mathcal{A}}$(or $\mathcal{S}_{\mathcal{A}}$) can be of different types, but for simplicity and clarity we allow the symbols to be overloaded. Whenever the type of quiplet is vital, we will explicitly specify it. However, when it is not specified our claims hold for all types of quiplets.

Depending on the level of detail required in the profile construction phase and in the ID phase, we represent the quiplets from the log file entries using three different representation levels. Each level is characterized by a different amount of recorded information.

We call the most naive representation of an audit log-file record, *coarse quiplet* or *c-quiplet*. A c-quiplet records only the number of distinct relations and attributes projected and selected by the SQL query. Therefore, c-quiplets essentially model how many relations and how many attributes are accessed in total, rather than the specific elements that are accessed by the query. The c-quiplets are defined as follows:

Definition 1 A **coarse quiplet** or **c-quiplet** is a representation of a log record of the database audit log file. Each c-quiplet consists of five fields (`SQL-CMD, PROJ-REL-COUNTER, PROJ-ATTR-COUNTER, SEL-REL-COUNTER, SEL-ATTR-COUNTER`). The first field is symbolic and corresponds to the issued SQL command. The next two fields are numeric, and correspond to the number of relations and attributes involved in the projection clause of the SQL query. The last two fields are also numeric, and correspond to the number of relations and attributes involved in the selection clause of the SQL query.

In terms of the quiplet notation $Q()$, here both $\mathcal{P}_{\mathcal{R}}$(or $\mathcal{S}_{\mathcal{R}}$) and $\mathcal{P}_{\mathcal{A}}$(or $\mathcal{S}_{\mathcal{A}}$) correspond to the number of relations and attributes involved in the query, respectively. Apparently, a large quantity of valuable information in the database log is ignored by c-quiplets. It is however useful to consider such a primitive data representation because it is sufficient in the case of a small number of well-separated roles. Moreover, more sophisticated representations of log-file entries are based on the definition of c-quiplets.

The second representation scheme captures more information from the log file records. We call this representation,

[3] The relation and attribute information is assumed to be present in the join conditions of the predicate. We do not consider the cases of complex sub-queries that cannot be reduced to join conditions.

[4] For clarity, we only show the representation for the syntax of a *select* command. The representation is general enough to capture information from other SQL commands such as *insert, delete and update*. For example, for the *insert* command, the inserted into relation and columns are encoded as the projection relation and projection attributes.

medium-grain quiplet or *m-quiplet*. These quiplets extend the coarse quiplets by further exploiting the information present in the log entries. Like a c-quiplet, a m-quiplet represents a single log entry of the database log file. Although in this case, each relation is represented separately by the number of its attributes projected (or selected) by the SQL query. Thus, in terms of the quiplet notation $Q()$, $\mathcal{P}_{\mathcal{R}}$, $\mathcal{P}_{\mathcal{A}}$, $\mathcal{S}_{\mathcal{R}}$ and $\mathcal{S}_{\mathcal{A}}$ are vectors of the same size which is equal to the number of relations in the database.

The m-quiplets are defined as follows:

Definition 2 A **medium-grain quiplet** or **m-quiplet** is a data object which corresponds to a single entry of the database log file and consists of five fields (`SQL-CMD`, `PROJ-REL-BIN[]`, `PROJ-ATTR-COUNTER[]`, `SEL-REL-BIN[]`, `SEL-ATTR-COUNTER[]`). The first field is symbolic and corresponds to the issued SQL command, the second is a binary (bit) vector of size equal to the number of relations in the database. The bit at position i is set to 1 if the ith relation is projected in the SQL query. The third field of the quiplet is a vector of size equal to the number of relations in the database. The ith element of the `PROJ-ATTR-COUNTER[]` vector corresponds to the number of attributes of the ith relation that are projected in the SQL query. The semantics of `SEL-REL-BIN[]` and `SEL-ATTR-COUNTER[]` vectors are equivalent to those of `PROJ-REL-BIN[]` and `PROJ-ATTR-COUNTER[]` vectors, but the information kept in the former corresponds to the selections rather than to the projections of the SQL query.

Finally, we introduce a third representation level of log-file records which extracts the maximum information from the log files. We call this representation *fine quiplet* or *f-quiplet*. The structure of a f-quiplet is similar to that of a m-quiplet. In particular, the first, the second and the fourth fields of a f-quiplet are the same as the corresponding fields of the m-quiplets. The f-quiplets and m-quiplets differ only for the third and fifth fields. In the case of f-quiplets, these fields are vector of vectors and are called `PROJ-BIN-ATTR[][]` and `SEL-BIN-ATTR[][]`, respectively, The ith element of `PROJ-BIN-ATTR[][]` is a vector corresponding to the ith relation of the database and having size equal to the number of attributes of relation i. The ith element of `PROJ-BIN-ATTR[][]` has binary values indicating which specific attributes of relation i are projected in the SQL query. The semantics of `SEL-BIN-ATTR[][]` are analogous. For f-triplets, $\mathcal{P}_{\mathcal{R}}$ and $\mathcal{S}_{\mathcal{R}}$ are vectors of size equal to the number of relations in the database, where as $\mathcal{P}_{\mathcal{A}}$ and $\mathcal{S}_{\mathcal{A}}$ are vectors of the same size, but with each element i being a vector of size equal to the number of attributes in relation i. The formal definition of the f-quiplets is as follows:

Definition 3 A **fine quiplet** or **f-quiplet** is a detailed representation of a log entry. It consists of five fields (`SQL-CMD`, `PROJ-REL-BIN[]`, `PROJ-ATTR-BIN[][]`, `SEL-REL-BIN[]`, `SEL-ATTR-BIN[][]`). The first field is symbolic and corresponds to the SQL command, the second is a binary vector that contains 1 in its ith position if the ith relation is projected in the SQL query. The third field is a vector of n vectors, where n is the number of relations in the database. Element `PROJ-ATTR-BIN[i][j]` is equal to 1 if the SQL query projects the jth attribute of the ith relation; it is equal to 0 otherwise. Similarly, the fourth field is a binary vector that contains 1 in its ith position if the ith relation is used in the SQL query predicate. The fifth field is a vector of n vectors, where n is the number of relations in the database. Element `SEL-ATTR-BIN[i][j]` is equal to 1 if the SQL query references the jth attribute of the ith relation in the query predicate; it is equal to 0 otherwise.

Table 1 shows a SQL command corresponding to select statement and its representation according to the three different types of quiplets. In the example, we consider a database schema consisting of two relations $R_1 = \{A_1, B_1, C_1, D_1\}$ and $R_2 = \{A_2, B_2, C_2, D_2\}$.

3 Role-based anomaly detection

In this section, we describe our methodology when information related to the roles of users is available in the database traces. This role information allows us to address the problem at hand as a standard classification problem.

3.1 Classifier

We use the Naive Bayes Classifier (NBC) for the ID task in RBAC-administered databases. Despite some modeling assumptions regarding attribute independence inherent to this classifier, our experiments demonstrate that it is surprisingly useful in practice. Moreover, NBC has proven to be effective in many practical applications such as text classification and medical diagnosis [8,10,22], and often competes with much more sophisticated learning techniques [9,17]. The reason for the popularity of NBC is its low computational requirements for both the training and classification task. The small running time is mainly due to the attribute independence assumption. Moreover, like all probabilistic classifiers under the Maximum Aposteriori Probability (MAP) decision rule, NBC arrives at the correct classification as long as the correct class is more probable than any other class. In other words, the overall classifier is robust to deficiencies of its underlying naive probability model. We refer the reader to the paper by Domingos and pazzani [8] that explains the optimality region for the NBC and discusses the reasons for its effective performance even when the attribute independence assumption does not hold.

Table 1 Quiplet construction example

SQL command	c-quiplet	m-quiplet	f-quiplet
SELECT $R_1.A_1$, $R_1.C_1$, $R_2.B_2$, $R_2.D_2$	select$< 2 >< 4 >$ $< 2 >< 2 >$	select $< 1, 1 >< 2, 2 >$	select $< 1, 1 >$
FROM R_1, R_2		$< 1, 1 >< 1, 1 >$	$< [1, 0, 1, 0], [0, 1, 0, 1] >$
WHERE $R_1.B_1 = R_2.B_2$			$< 1, 1 > [0, 1, 0, 0], [0, 1, 0, 0]$

We first describe the general principles of the NBC (for details see [22]) and then show how it can be applied to our setting. In supervised learning, each instance x of the data is described as a conjunction of attribute values, and the target function $f(x)$ can only take values from some finite set V. The attributes correspond to the set of observations and the elements of V are the distinct classes associated with those observations. In the classification problem, a set of training examples D_T is provided, and a new instance with attribute values $(a_1, \dots, a_n)$ is given. The goal is to predict the target value, or the class, of this new instance.

The approach we describe here is to assign to this new instance the most probable class value v_{MAP}, given the attributes $(a_1, \dots, a_n)$ that describe it. That is

$$v_{\mathrm{MAP}} = \arg\max_{v_j \in V} P(v_j | a_1, a_2, \dots, a_n).$$

Using *Bayes Theorem* we can rewrite the expression as

$$\begin{aligned} v_{\mathrm{MAP}} &= \arg\max_{v_j \in V} P(v_j | a_1, a_2, \dots, a_n) \\ &= \arg\max_{v_j \in V} \frac{P(a_1, a_2, \dots, a_n | v_j) P(v_j)}{P(a_1, a_2, \dots, a_n)} \\ &\propto \arg\max_{v_j \in V} P(a_1, a_2, \dots, a_n | v_j) P(v_j). \end{aligned}$$

The last derivation is feasible because the denominator does not depend on the choice of v_j and thus, it can be omitted from the arg max argument. Estimating $p(v_j)$ is simple because it requires just counting the frequency of v_j in the training data. However, calculating $P(a_1, a_2, \dots, a_n | v_j)$ is hard when considering a large dataset and a reasonably large number of attributes [7]. The NBC, however, is based on the simplifying assumption that the attribute values are *conditionally independent*, and thus

$$v_{\mathrm{MAP}} \propto \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j). \tag{1}$$

This reduces significantly the computational cost because calculating each one of the $P(a_i | v_j)$ requires only a frequency count over the tuples in the training data with class value equal to v_j.

Thus, the conditional independence assumption seems to solve the computational cost. However, there is another issue that needs to be discussed. Assume an event e occurring n_{e_j} number of times in the training dataset for a particular class v_j with size $|D_{v_j}|$. While the observed fraction $(\frac{n_{n_j}}{|D_{v_j}|})$ provides a good estimate of the probability in many cases, it provides poor estimates when n_{e_j} is very small. An obvious example is the case where $n_{e_j} = 0$. The corresponding zero probability will bias the classifier in an irreversible way, because according to Eq. 1, the zero probability when multiplied with the other probability terms will give zero as its result. To avoid this difficulty we adopt a standard Bayesian approach in estimating this probability, using the m-estimate [22]. The formal definition of m-estimate is as follows:

Definition 4 Given a dataset D_T with size $|D_T|$ and an event e that appears n_{e_j} times in the dataset for a class v_j with size $|D_{v_j}|$ and n_e times in the entire dataset, then the *m-estimate of the probability* $p_{e_j} = \frac{n_{e_j}}{|D_{v_j}|}$ is defined to be

$$p^m_{e_j} = \frac{n_{e_j} + m \cdot \frac{n_e}{|D_T|}}{|D_{v_j}| + m}. \tag{2}$$

The parameter m is a constant and is called *equivalent sample size*, which determines how heavily to weight p_{e_j} relative to the observed data. If n_E is 0, then we assume that $p^m_E = \frac{1}{|D_{v_j}|}$.

The NBC directly applies to our anomaly detection framework by considering the set of roles in the system as classes and the log-file quiplets as the observations. In what follows, we show how Eq. 1 can be applied for the three different types of quiplets.

For the case of c-quiplets the application is simple because there are five attributes $(c, \mathcal{P}_\mathcal{R}, \mathcal{P}_\mathcal{A}, \mathcal{S}_\mathcal{R}, \mathcal{S}_\mathcal{A})$ to consider, namely the command, the projection relation count, the projection attribute count, the selection relation count and the selection attribute count. If $\mathcal{R}$ denotes the set of roles, the predicted role of a given observation $(c_i, \mathcal{P}_{\mathcal{R}i}, \mathcal{P}_{\mathcal{A}i}, \mathcal{S}_{\mathcal{R}i}, \mathcal{S}_{\mathcal{A}i})$ is

$$\begin{aligned} r_{\mathrm{MAP}} = \arg\max_{r_j \in \mathcal{R}} \; & \{ p(r_j) p(c_i | r_j) p(\mathcal{P}_{\mathcal{R}i} | r_j) p(\mathcal{P}_{\mathcal{A}i} | r_j) \\ & p(\mathcal{S}_{\mathcal{R}i} | r_j) p(\mathcal{S}_{\mathcal{A}i} | r_j) \}. \end{aligned}$$

For m-quiplets, we again have five fields $(c, \mathcal{P}_\mathcal{R}, \mathcal{P}_\mathcal{A}, \mathcal{S}_\mathcal{R}, \mathcal{S}_\mathcal{A})$, where $\mathcal{P}_\mathcal{R}, \mathcal{P}_\mathcal{A}, \mathcal{S}_\mathcal{R}$ and $\mathcal{S}_\mathcal{A}$ are vectors of the same cardinality. Except for the command attribute c, the rest of the attributes considered in this case are from the product $\mathcal{P}_\mathcal{R}\mathcal{P}_\mathcal{A}^\mathcal{T}$ and $\mathcal{S}_\mathcal{R}\mathcal{S}_\mathcal{A}^\mathcal{T}$. Therefore there are $|\mathcal{P}_\mathcal{R} \cdot \mathcal{P}_\mathcal{A}^\mathcal{T}| + |\mathcal{S}_\mathcal{R} \cdot \mathcal{S}_\mathcal{A}^\mathcal{T}| + 1$ attributes, and Eq. 1 can be rewritten as follows

$$r_{\text{MAP}} = \arg\max_{r_j \in \mathcal{R}} \left\{ p(r_j) p(c_i|r_j) \prod_{i=1}^{N} \{ p(\mathcal{P}_\mathcal{R}[i] \cdot \mathcal{P}_\mathcal{A}{}^T[i]|r_j) \; p(\mathcal{S}_\mathcal{R}[i] \cdot \mathcal{S}_\mathcal{A}{}^T[i]|r_j) \} \right\},$$

where N is the number of relations in the DBMS.

Finally, for f-quiplets, where fields $\mathcal{P}_\mathcal{R}$,$\mathcal{S}_\mathcal{R}$ are vectors and $\mathcal{P}_\mathcal{A}$,$\mathcal{S}_\mathcal{A}$ are vectors of vectors, the corresponding equation is

$$r_{\text{MAP}} = \arg\max_{r_j \in \mathcal{R}} \left\{ p(r_j) p(c_i|r_j) \prod_{i=1}^{N} \{ p(\mathcal{P}_\mathcal{R}[i] \cdot \mathcal{P}_\mathcal{A}[i]|r_j) \; p(\mathcal{S}_\mathcal{R}[i] \cdot \mathcal{S}_\mathcal{A}[i]|r_j) \} \right\}.$$

With the above definitions in place, the ID task is quite straightforward. For every new query, its r_{MAP} is predicted by the trained classifier. If this r_{MAP} 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 increasing the frequency count of the relevant attributes.

The procedure for ID can easily be generalized for the case when a user is assigned more than one role at a time. This is because our method detects anomalies on a per query basis rather than per user basis. Hence, as long as the role associated with the query is consistent with the role predicted by the classifier, the system will not detect an anomaly.

3.2 Experimental evaluation

In this section, we report results from an experimental evaluation of the proposed approach and illustrate its performance as an ID mechanism. Our experimental setting consists of experiments with both synthetic and real data sets. In our previous work [4], we had reported the performance of the three quiplet types under different modes of database access patterns. The objective of the current experimental evaluation is to assess the performance of our methods on databases deployed for real-world applications. For modeling the SQL query access patterns in a real-world deployed database, we use the general form of a *zipf* probability distribution function (pdf) that is frequently used to model non-uniform access. The zipf pdf, for a random variable X, is mathematically defined as follows:

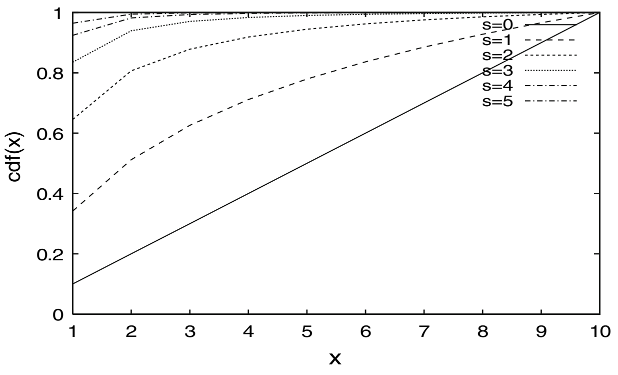


Fig. 2 A sample Zipf distribution for $N = 10$

$$Zipf(X, N, s) = \frac{1/x^s}{\Sigma_{i=1}^{N} 1/i^s},$$

where N is the number of elements and s is the parameter characterizing the distribution. Figure 2 shows the cumulative density function for a zipf distribution for $N = 10$ and different values of s. Suppose N here denotes the number of tables in a database schema ordered according to some criteria such as lexicographic order. Then Fig. 2 shows that, as we increase s, the probability mass accumulates towards the left half of the schema, thereby making the access pattern more and more skewed. For our experiments, we also use a *reverse zipf* distribution which is a mirror image of the corresponding zipf plot with respect to a vertical axis.

Before describing our experimental findings, we give a brief outline of the generation procedure for our test datasets and anomalous queries.

3.2.1 Data sets

Synthetic data sets: The synthetic data are generated according to the following model: Each role r has a probability, $p(r)$, of appearing in the log file. Additionally, for each role r the generator specifies the following five probabilities: (i) the probability of using a command c given the role, $p(c|r)$, (ii) the probability of projecting on a table t given the role and the command, $p(P_t|r, c)$, (iii) the probability of projecting an attribute within a table $a \in T$ given the role, the table and the command, $p(P_a|r, t, c)$, (iv) the probability of using a table t in the selection clause given the role and the command, $p(S_t|r, c)$ and finally, (v) the probability of using an attribute $a \in t$ in the query predicate given the role, the table and the command, $p(S_a|r, t, c)$. We use four different kinds of probability distribution functions for generating these probabilities namely, uniform, zipf, reverse zipf and multinomial.

Real data set: The real dataset used for evaluating our approach consists of 8, 368 SQL traces from eight different applications submitting queries to a MS SQL server database. The database schema consists of 130 tables and $1,201$ columns in all. The queries in this dataset consist of a mix of *select, insert and update* commands with precisely $7,583$ *select* commands, 213 *insert* commands and 572 *update* commands. There are no sub-queries present in any of the query predicates. Furthermore, because role information is not available, we consider the applications themselves as our roles. For a more detailed description of the dataset we refer the reader to [28].

Anomalous query generation: We generate the anomalous query set keeping in mind the insider threat scenario. For this, we generate the anomalous queries from the same probability distribution as that of normal queries, but with role information negated. For example, if the role information associated with a normal query is 0, then we simply change the role to any role other than 0 to make the query anomalous.

3.3 Results

We now describe the first synthetic dataset that we use for our experiments. The database schema consists of 100 tables and 20 columns in each tables. The number of roles for the database is 4. The SQL query submission pattern for the roles is governed by the pdf, $Zipf(N = 4, s = 1)$. The first two roles are read-only, such that they use the *select* command with probability 1. The first role accesses the tables with a pdf, $Zipf(100, s)$, and the columns with a pdf, $Zipf(20, s)$. We vary the parameter s for our experiments. Similarly, the second role accesses the tables and columns with a pdf governed by $R\_Zipf(100, s)$ and $R\_Zipf(20, s)$, respectively. The third and the fourth roles are read–write such that they issue the *select*, *insert*, *delete* and *update* commands with probabilities 0.1, 0.1, 0.1 and 0.7, respectively. For the *select*, *delete* and *insert* commands, these two role access all the tables and columns within each table with a uniform probability. The third role executes the *update* command with a pdf, $Zipf(100, s)$, and the fourth with a pdf, $R\_Zipf(100, s)$. We use a training data size of cardinality $5,000$ and set the m parameter (in Eq. 2) to 100. Figure 3 shows the False Positive (FP) and False Negative (FN) rates for increasing values of s. As expected, the FP and the FN rate for f-quiplet is the lowest among the three quiplet types. Also, as we make the database access becomes more skewed by increasing s, FP rate for the f-quiplet goes down.

We generate the second dataset as follows. The database schema is same as in the first dataset with 100 tables and 20 columns in each table. However, there are now 9 roles that access the database as shown in Fig. 4. Roles 1 to 6 are read-only and roles 7, 8 and 9 are read-write. Fig. 5 shows the FP and FN rates for this dataset. An interesting observation is that the performance of an m-quiplet is actually better than that of an f-quiplet for lower values of s and comparable to f-quiplet for higher values of s. This suggests that m-quiplet may prove to be an effective replacement for f-quiplet for a DBMS with an access pattern similar to that of the second dataset.

Finally, we present experimental results for the real data set. The results are averaged over a ten-fold cross validation of the dataset. Anomalous queries are generated as described earlier. The parameter m in Eq. 2 is again set to be 100. Table 2 shows the performance of the three quiplet types. The FN rate for all three quiplet types is quite low. One matter of concern is the high FP rate for this dataset. This result could be due to the specific nature of the real dataset; or for m and f-quiplet the large number of attributes may trigger such behavior.

Overall, the experimental evaluation reveals that in most cases f-quiplet capture the access pattern of the users much better than either c or m-quiplet.

4 Unsupervised anomaly detection

We now turn our attention to the case where the role information is not available in the audit log files. In this case, the problem of forming user profiles is clearly unsupervised and thus it is treated as a clustering problem. The specific methodology that we use for the ID task is as follows: we partition the training data into clusters[5] 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. The methods we use for clustering include some standard techniques. The next section explains in detail the distance measures used for clustering. After that we briefly explain the clustering algorithms and the statistical test for detecting intruders and finally report experimental results on them.

[5] In the unsupervised setting, the clusters obtained after the clustering process represent the profiles.

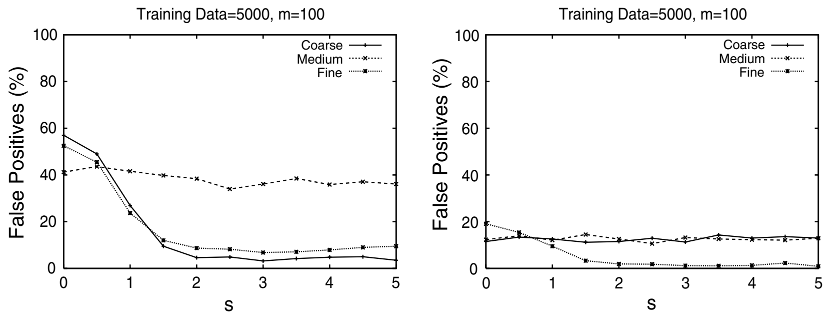


Fig. 3 Dataset 1: False Positive and False Negative rate

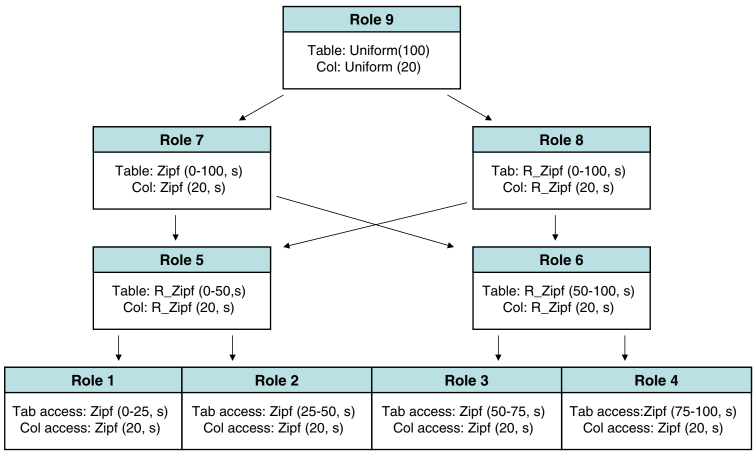


Fig. 4 Dataset 2: description of roles

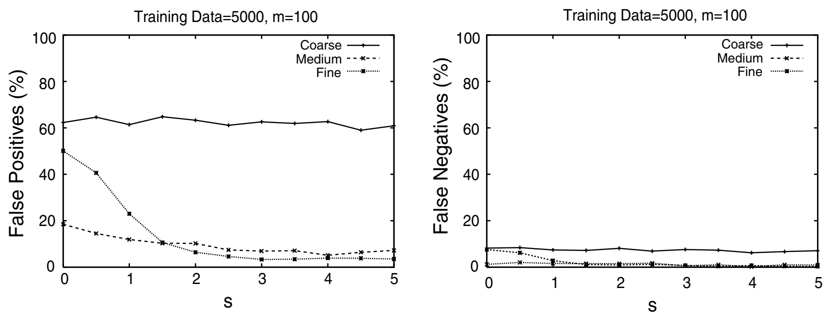


Fig. 5 Dataset 2: False Positive and False Negative rate

4.1 Distance functions

For clustering the quiplets into groups such that quiplets in the same group are close to each other, we need a measure to establish the "closeness" between the quiplets. For this we provide definitions of the necessary distance functions.

In order to introduce the distance functions, we first need to introduce a (generic and overloaded) function $\beta(\,)$ which will be used for evaluating and comparing quiplets of the same type. Let $Q = (c, \mathcal{P}_\mathcal{R}, \mathcal{P}_\mathcal{A}, \mathcal{S}_\mathcal{R}, \mathcal{S}_\mathcal{A})$ and $Q' = (c', \mathcal{P}'_\mathcal{R}, \mathcal{P}'_\mathcal{A}, \mathcal{S}'_\mathcal{R}, \mathcal{S}'_\mathcal{A})$ be two quiplets in general, and let $T = (\mathcal{P}_\mathcal{R}, \mathcal{P}_\mathcal{A}, \mathcal{S}_\mathcal{R}, \mathcal{S}_\mathcal{A})$ and $T' = (\mathcal{P}'_\mathcal{R}, \mathcal{P}'_\mathcal{A}, \mathcal{S}'_\mathcal{R}, \mathcal{S}'_\mathcal{A})$ denote information contained in Q and Q', respectively, minus the command c. We define, $\beta : T \times T \to \Re$ as a mapping from pairs of quiplets (minus the command c) to real numbers.

– For *c-quiplet*, function $\beta(\,)$ is calculated as follows:

$$\beta(T, T') = \sqrt{(P_R - P'_R)^2 + (P_A - P'_A)^2 + (S_R - S'_R)^2 + (S_A - S'_A)^2}$$

– For *m-quiplet*, we have:

$$\beta(T, T') = ||P_R P_A - P'_R P'_A||_2 + ||S_R S_A - S'_R S'_A||_2$$

Note that given two vectors $v_i = \{v_{i1}, v_{i2}, \ldots, v_{in}\}$ and $v_j = \{v_{j1}, v_{j2}, \ldots, v_{jn}\}$, their L_2 distance $||v_i - v_j||_2$ is defined to be $||v_i - v_j||_2 = \sqrt{\sum_{\ell=1}^{n}(v_{i\ell} - v_{j\ell})^2}$.

Table 2 Real data: False Positive and False Negative rate

Quiplet type	False Negative (%)	False Positive (%)
c	2.6	19.2
m	2.4	17.1
f	2.4	17.9

– For f-quiplet, function $\beta(\,)$ is calculated as follows:

$$\beta(T, T') = \sum_{i=1}^{N} \{||P_R[i]P_A[i] - P'_R[i]P'_A[i]||_2 + ||S_R[i]S_A[i] - S'_R[i]S'_A[i]||_2\}$$

Observation 1 *All the above definitions of $\beta(\,)$ satisfy the triangle inequality.*

Definition 5 The distance between two quiplets Q and Q' is defined as follows:

$$d_Q(Q, Q') = \begin{cases} 1 + \beta(T, T') & \text{if } c \neq c' \\ \beta(T, T') & \text{otherwise} \end{cases} \tag{3}$$

The following lemma states an important property of function $d_{\mathcal{T}}$.

Lemma 1 *If $\beta(\,)$ satisfies the triangle inequality, then $d_Q(\,)$ satisfies the triangle inequality.*

Proof Consider three quiplets T_1, T_2 and T_3, minus the command c. If $\beta(\,)$ satisfies the triangle inequality then the following inequality holds:

$$\beta(T_1, T_2) + \beta(T_2, T_3) \geq \beta(T_1, T_3).$$

This means that d_Q satisfies the triangle inequality as well for all the cases when $c_1 = c_2 = c_3$. Therefore we only have to re-examine the case when $c_1 \neq c_3$. Assume then that $c_1 \neq c_3$. If $c_1 = c_2$, then it should be that $c_3 \neq c_2$ and therefore the triangular inequality for d_Q is also preserved.

4.2 Clustering algorithms

This section describes the algorithms that are used for forming the profiles in the unsupervised setting.

4.2.1 k-centers

The k-centers algorithm takes as input the set of data points and their distances and a parameter k, which is the desired number of clusters. The output is a flat k-clustering, that is, a single clustering consisting of k clusters $\mathcal{C} = \{C_1, \ldots, C_k\}$. The clusters form a partitioning for the input data points.

If we denote by x a data point, that is a quiplet in the log files, and by μ_j the point representing the jth cluster, in the k-centers clustering algorithm we try to find the partition that optimizes the following function:

$$\max_j \max_{x \in C_j} d_Q(x, \mu_j)$$

This problem is NP-Hard. For solving it we use the following approximate algorithm [11], also known as the *furthest-first traversal* technique. The idea is to pick any data point to start with, then choose the point furthest from it, then the point furthest from the first two[6] and so on until k points are obtained. These points are taken as cluster centers and each remaining point is then assigned to the closest center. This algorithm provides a 2-approximation guarantee for any distance function that is a metric. Given Lemma 1 that proves that d_Q is a metric, the above algorithm provides a 2-approximate solution in our setting as well.

This algorithm minimizes the largest radius of the clusters that are returned as output and uses as cluster centers, or representatives, points that are already in the data set. The advantages of this algorithm are expected to be revealed in cases in which the data set does not contain large number of outliers. That is, if the data we use for creating user profiles are free from intruders, this algorithm is expected to create profiles reasonably close to the real ones.

4.2.2 k-means

In order to address the case where some outliers may exist in the data used to build the profiles, we also consider an alternative clustering heuristic. This is the widely used k-means algorithm. The k-means algorithm is also a member of the flat k-clustering algorithms, that output a single clustering consisting of k-clusters that partition the input points. Although, there is no proof of how good approximations we obtain using k-means, the algorithm has been widely adopted due to its low computational requirements, ease of implementation and mainly due to the fact that it works well in practice. The algorithm consists of a simple re-estimation procedure and works as follows. First, k points are chosen randomly, representing the initial cluster representatives. In this case, the representatives of the clusters correspond to the means of the data points in the cluster, given the metric space. Then, the remaining data points are assigned to the closest cluster. The new representatives, subject to the last assignment, are re-computed for each cluster. The last two steps are alternated until a stopping criterion is met, that is, when there is no further change in the assignment of data points to clusters. The algorithm minimizes the following cost function:

$$\sum_j \sum_{x \in C_j} d_Q(x, \mu_j)$$

[6] The distance of a point x from a set S is the usual $\min\{d_Q(x, y) : y \in S\}$.

where x again corresponds to a data point and μ_j is the representative of the jth cluster; and in this case, it is the mean of the points in the cluster.

A significant advantage of the k-means algorithm when compared to the other clustering algorithms discussed in this section is that updates of the clusters can be executed in constant time. Consider the case in which we have already formed the k clusters on some initial set of normal input points. Now assume that new normal points arrive and we want to incorporate them into the existing clusters. Assume that x is a new point and we want to incorporate it in cluster C_i that has cardinality $|C_i|$ and is described by the mean μ_i. Then of course finding the new mean μ_i' of the cluster after the addition of point x is a trivial task, since $\mu_i' = \frac{x+\mu_i \cdot |C_i|}{|C_i|+1}$. Now our additional claim is that the error in the new cluster that contains the points $C_i \cup \{x\}$ can also be computed in constant time. This can be executed by computing the error of each cluster by using the following formula:

$$\sum_{i=1}^{|C_i|}(x_i - \mu_i) = \frac{1}{|C_i|}\sum_{i=1}^{|C_i|} x_i^2 - \left(\frac{1}{|C_i|}\sum_{i=a}^{b} x_i\right)^2$$

Now, the error when the additional point x is added can be computed in constant time by keeping two pre-computed arrays for the cluster points: the sum of the values and the sum of squares of the values of the points appearing in the cluster.

4.3 Anomaly detection methodology

So far we have described two alternative ways of building the profiles given unclassified log quiplets. In this section, we describe our methodology in detail for identifying anomalous behavior given the set of constructed profiles.

Let z denote an issued SQL query for which our mechanism has to tell whether it is anomalous or not. The mechanism that decides whether a query is a potential intruder works as follows:

1. Find the *representative cluster* (C_z) for query z issued by user U. The representative cluster is obtained by simply looking up the user-cluster mapping created during the clustering phase.
2. We specify two different approaches for the detection phase. In the first approach, we use the naive bayes classifier in a manner similar to the supervised case by treating the clusters as the classifier classes. In the second approach, we determine if z is an outlier in cluster C_z with the help of a statistical test. If it is an outlier, we declare z as an anomaly.

In the second approach for the detection phase, we use a statistical test for deciding whether a query is an anomaly or not, but we do not specify the statistical test to use. In principle, any test appropriate for identifying outliers from a univariate data set which cannot be mapped to a standard statistical distribution like Normal and Lognormal, is applicable. In our setting we use the *MAD* (Median of Absolute Deviations) test [14], which we describe below in brief.

MAD test: Assume to have n data points (log quiplets). Let d_i denote the distance of data point i from the cluster center it belongs to. Also, let $\overline{d}$ denote the median value of the d_i's for $i = 1, 2, \ldots, n$. Then first, we calculate the MAD as

$$\text{MAD} = \text{median}_i(|d_i - \overline{d}|).$$

Additionally, for each point i we calculate

$$Z_i = \frac{0.6745(d_i - \overline{d})}{\text{MAD}}.$$

Now if $|Z_i| > D$, then d_i is an outlier, meaning that we can infer that point i is an outlier. D is a constant which has to be experimentally evaluated. In our case, it is set to 1.5 because for this value we experience satisfactory performance of our system. We treat differently the special case where $\text{MAD} = 0$. This case is quite likely because many quiplets are expected to collide with the cluster center. In that case, we consider a point i that belongs in profile C_j as an outlier if $d_Q(i, \mu_j) > \bar{d_{\mu_j}} + 2 \cdot \sigma_j$. In the above equation, $\bar{d_{\mu_j}}$ corresponds to the mean of the distances of all the points in cluster C_j from the representative of cluster i, namely μ_j. Likewise, σ_j corresponds to the standard deviation of those distances.

4.4 Experimental evaluation

We now present the experimental results for the unsupervised case. The objective of the unsupervised case, after forming the user-cluster mapping is similar to that of the supervised case. For every new query under observation, we check if the user associated with the query is indeed a member of its mapped cluster. The dataset that we use for this evaluation is similar to the dataset 2 used in the supervised case. However, we reduce the number of tables to 20 and the number of columns per table to 10. The number of training records used for clustering are 1,000. The results are averaged over five iterations of the clustering algorithms.

Figures 6 and 7 show the results for the naive bayes detection methodology for both k-means and k-centers clustering algorithms. The FP rate for both the algorithms is extremely low. However, the corresponding FN rate for k-means is much better than that of k-centers. This makes k-means the algorithm of choice for the kind of dataset considered in this test case. Another noticeable observation is the better performance of m-quiplet over f-quiplet for datasets with smaller values of s.

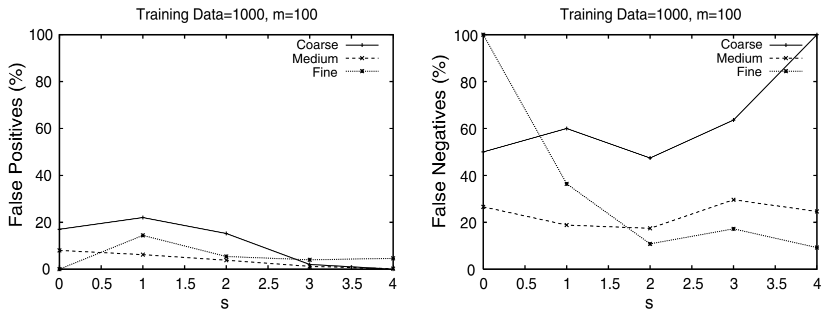


Fig. 6 Unsupervised dataset: *k*-means—False Positive and False Negative rate for the naive bayes detection methodology

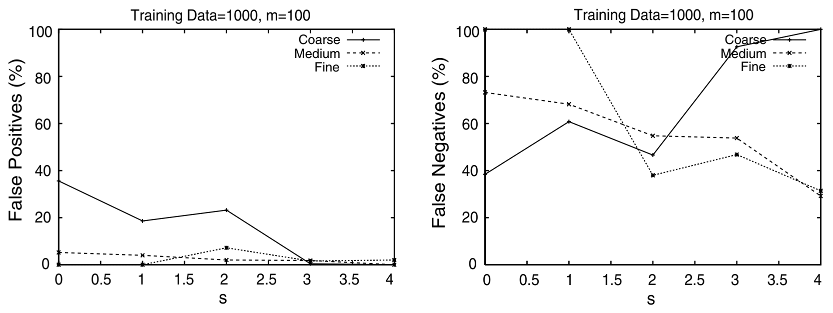


Fig. 7 Unsupervised dataset: *k*-centers—False Positive and False Negative rate for the naive bayes detection methodology

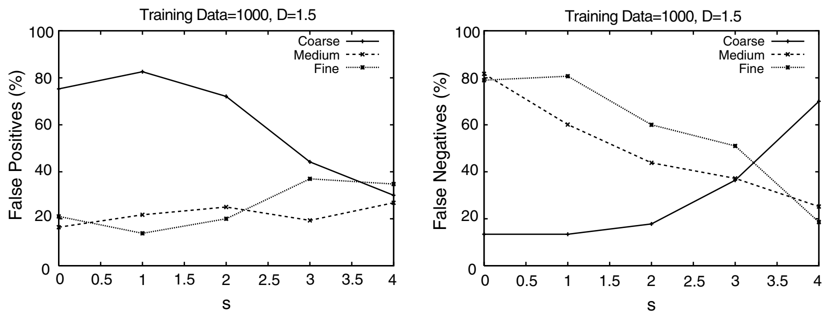


Fig. 8 Unsupervised dataset: *k*-means—False Positive and False Negative rate for the outlier detection methodology

Figures 8 and 9 report the performance of the experiments for the outlier detection methodology. The results for the outlier detection methodology are not very impressive for either of the clustering algorithms. One probable reason for this result is that anomalous queries considered in this test case come from the same probability distribution as that of the normal queries, although with role information inverted. Since they come from the same distribution, they no longer behave as outliers in the metric space and therefore, the outlier detection methodology fails to characterize most of the anomalous queries as outliers. We illustrate this with the help of Fig. 10 which shows the FN rate for *k*-means and *k*-centers when the anomalous queries are generated from a uniform random probability distribution. For such queries, the FN rate decreases as the access pattern becomes more specific. This shows the usefulness of the outlier detection-based methodology when the access pattern of users deviate from the overall distribution of the normal access pattern.

Overall, the clustering-based approach for the unsupervised case shows promising results for both m and f-quiplet. Among the clustering algorithms, the results for *k*-means are better than those for the *k*-centers algorithm. This is because *k*-means better captures the trend of the dataset.

5 Conclusions and future work

In this paper, we have proposed an approach for detecting anomalous access patterns in DBMS. We have developed three models, of different granularity, to represent the SQL queries appearing in the database log files. We were thus able to extract useful information from the log files regarding the access patterns of the queries. When role information is available in the log records, we use it for training a classifier that is then used as the basic component for our anomaly detection mechanism. For the other case, when no role information is present in the log records, the user profiles are created using standard clustering algorithms. The anomaly detection phase is then addressed in a manner similar to the supervised case or as an outlier detection problem. Experimental results for both real and synthetic data sets showed that our methods perform reasonably well.

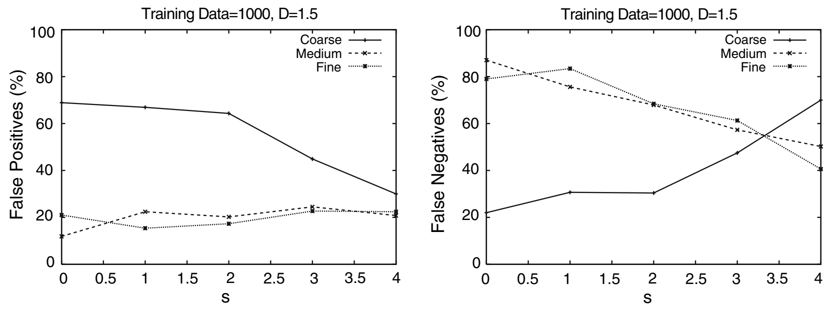


Fig. 9 Unsupervised dataset: *k*-centers—False Positive and False Negative rate for the outlier detection methodology

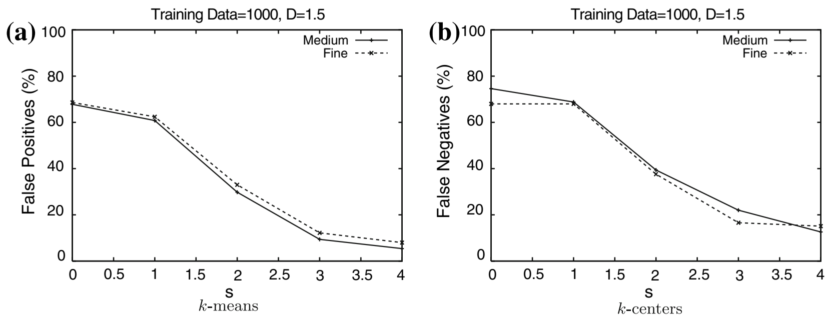


Fig. 10 Unsupervised dataset: False Negative rate for the outlier detection methodology with intrusion queries from a different probability distribution

We plan to extend this work on following lines. The modeling of queries described in this paper extracts information only from the query syntax. We plan to extend this approach by also considering the query semantics in terms of the data values used in the predicates and the quantum of data 'accessed' by the queries because of these data values. This is a non-trivial problem to address because of the dynamic nature of the underlying data in the database. Another complementary area for further research is the response of the system when an intrusion is detected. We envisage an ID system architecture (Fig. 1) with a separate and well-defined intrusion response component. The intrusion response system, depending on the anomaly that is detected, issues an appropriate response type either by looking up a policy base of pre-defined response mechanisms or by learning from its own previous responses. Finally, as part of our efforts to improve the techniques used for database intrusion detection, we would like to experiment with other novel machine learning techniques.

References

1. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Hippocratic databases. In Proceedings of the 28th International Conference on Very Large Data Bases (VLDB), pp.143–154. Morgan-Kaufmann, New York (2002)
2. Anton, A., Bertino, E., Li, N., Yu, T.: A roadmap for comprehensive online privacy policies. In: CERIAS Technical Report (2004)
3. Axelsson, S.: Intrusion detection systems: a survey and taxonomy. Technical Report 99–15, Chalmers Univ., (2000)
4. Bertino, E., Kamra, A., Terzi, E.: Intrusion detection in rbac-administered databases. In: Proceedings of the Applied Computer Security Applications Conference (ACSAC) (2005)
5. Bertino, E., Leggieri, T., Terzi, E.: Securing dbms: characterizing and detecting query floods. In: Proceedings of the International Security Conference (ISC) (2004)
6. Chung, C., Gertz, M., Levitt, K.: 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)
7. Cooper, G.F.: The computational complexity of probabilistic inference using bayesian belief networks. Artif. Intell. **42**(2–3), 393–405 (1990)
8. Domingos, P., Pazzani, M.J.: On the optimality of the simple bayesian classifier under zero-one loss. Mach. Learn. **29**(2–3), 103–130 (1997)
9. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Mach. Learn. **29**(2–3), 131–163 (1997)
10. Hilden, J.: Statistical diagnosis based on conditional independence does not require it. Comput. Biol. Med. **14**(4), 429–435 (1984)
11. Hochbaum, D.S., Shmoys, DB.: A best possible approximation algorithm for the k-center problem. Math. Oper. Res. **10**, 180–184 (1985)
12. Hoglund, K.H.A., Sorvari, A.: 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)
13. Hu, Y., Panda, B.: Identification of malicious transactions in database systems. In: Proceedings of the International Database Engineering and Applications Symposium (IDEAS) (2003)
14. Iglewicz, B., Hoaglin, D.C.: How to Detect and Handle Outliers. ASQC Quality Press, Milwaukee, Wisconsin (1993)
15. Kruegel, C., Vigna, G.: Anomaly detection of web-based attacks. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS) (2003)

16. Lane, T., Brodley, CE.: Temporal sequence learning and data reduction for anomaly detection. ACM Trans. Inf. Syst. Secur. (TISSEC) **2**(3), 295–331 (1999)
17. Langley, P., Iba, W., Thompson, K.: An analysis of bayesian classifiers. In: National Conference on Artificial Intelligence pp.223–228 (1992)
18. Lee, S.Y., Low, W.L., Wong, P.Y. Learning fingerprints for a database intrusion detection system. In: ESORICS '02: Proceedings of the 7th European Symposium on Research in Computer Security London. pp. 264–280, Springer-Heidelburg (2002)
19. Lee, V., Stankovic, J., Son, S.: Intrusion detection in real-time databases via time signatures. In: Proceedings of the IEEE Real-Time Technology and Applications Symposium (RTAS) (2000)
20. Liu, P.: Architectures for intrusion tolerant database systems. In: Proceedings of the Annual Computer Security Applications Conference (ACSAC) (2002)
21. Lunt, T., Tamaru, A., Gilham, F., Jagannathan, R., Neumann, P., Javitz, H., Valdes, A., Garvey, T.: A real-time intrusion detection expert system (ides)—final technical report. Technical Report, Computer Science Laboratory, SRI International (1992)
22. Mitchell, TM.: Machine Learning. McGraw-Hill, Newyork (1997)
23. Sandhu, R., Ferraiolo, D., Kuhn, R.: The nist model for role based access control: Towards a unified standard. In: Proceedings of the 5th ACM Workshop on Role Based Access Control (2000)
24. Spalka, A., Lehnhardt, J.: A comprehensive approach to anomaly detection in relational databases. In: DBSec, pp. 207–221 (2005)
25. Talpade, R., Kim, G., Khurana, S.: Nomad: traffic-based network monitoring framework for anomaly detection. In: Proceedings of the 4th IEEE Symposium on Computers and Communications (ISCC) (1998)
26. Valeur, F., Mutz, D., Vigna, G.: 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)
27. Wenhui, S., Tan, T.: 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)
28. Yao, Q., An, A., Huang, X. Finding and analyzing database user sessions. In: Proceedings of the 10th International Conference on Database Systems for Advanced Applications (DASFAA) (2005)