Department of Informatics, University of Zürich

BSc In-Depth Study

A Role Intrusion Detection System for Role Based Access Controlled Relational Database Management Systems

Philip Hofmann

Matrikelnummer: 14-710-842

Email: philip.hofmann@uzh.ch

June 24, 2017

supervised by Prof. Dr. Michael Böhlen and Kevin Wellenzohn

University of Zurich^UZH

Department of Informatics

Abstract

Most companies maintain a Relational Database Management System (RDBMS) containing sensitive information. Because this data is valuable, attacks against RDBMS have become more sophisticated. Though database systems implement effective access controls, the necessity for stronger security has become apparent. We consider a solution proposed by Ashish Kamra et al. that suggests an Intrusion Detection System (IDS) which extends a RDBMS and intercepts SQL statements to identify anomalous access patterns. The IDS protects from insider threats by identifying role intruders thorough mining the audit log of the database and running a Naive Bayes Classifier to categorize incoming SQL statements as safe or anomalous. This paper studies the proposed solution, applies a scenario and considers situations in which this system might fail.

1 Introduction

In today's world, where data has become increasingly valuable, threats of gaining and misusing access to information have been more and more on the rise. Institutions need to protect their data and even though RDBMS provide good access control mechanisms, it is essential for the new generation of security aware databases to employ more advanced security techniques. One needed security feature is the ability to detect anomalous behavior of applications and users, since impersonation attacks cannot be thwarted by access control mechanisms alone. A solution to this is extending the database with an Intrusion Detection System (IDS). Such IDS have been successfully implemented for networks and operating systems, but intrusion detection techniques specifically tailored towards database systems are still being researched. This is why in this report, we study the IDS proposed by Kamra et al. (Kamra, A., Terzi, E., Bertino, E. 2008. Detecting anomalous access patterns in relational databases. The VLDB Journal 17:1063–1077). The key idea of the system is to build profiles out of users interacting with the database through SQL statements. We assume a Role Based Access Control (RBAC) model, where permissions are assigned to roles and a user is assigned to one or more roles. Assuming a database with an RBAC model, we can correlate user behavior with a specific user role. The IDS builds a profile for each role based on the audit log of the database and is therefore able to match an incoming SQL statement to the most likely role. If the most likely role is different from the actual associated role, the statement is flagged as anomalous. Therefore, the system can identify role intruders, i.e. users holding a specific role that deviate from expected normal role behavior. We now introduce our application scenario to highlight the problem of role intruders and will later use this scenario to explain the system in detail.

Scenario Our application scenario is based on a fictitious university. This simplified university maintains a database with one table for the students who are currently enrolled. This database is being accessed by two specific user roles, the first role is called "administration" (AD) and the second is called "rectorate" (RE). Whenever a student moves into a new apartment, he/she needs to report his/her address change to the university. The administration hired a computer science student to update the student data in the table. Typical behavior of the AD role is accessing single rows and using SELECT and UPDATE commands in the SQL statements. The rectorate instead offers services, which include sending out mass emails and mass mail to students for departments and third parties. Typical behavior of this role RE is retrieving information in batch from the database, never selecting single rows.

Table 1.1: Sample students relation

ID	LastName	FirstName	Street	Number
0	Mueller	Thomas	Zelgstrasse	36
1	Ankli	Lea	Zelglistrasse	35
2	Case	Justin	Dubsstrasse	42
3	Maria	Ave	Zelglistrasse	35

Table 1.2: Sample students relation continued

City	Code	Email	Departement
Zuerich	8003	thomas.mueller@uni.ch	Informatics
Aarau	5000	lea.ankli@uni.ch	Law
Zuerich	8003	justin.case@uni.ch	Law
Aarau	5000	ave.maria@uni.ch	Informatics

2 Intrusion Detection System Achitecture

The RDBMS is extended with a separate and independent Intrusion Detection System (IDS). The components of the IDS are illustrated in this graphic taken from the paper of A. Kamra et al. and described in the following sections.

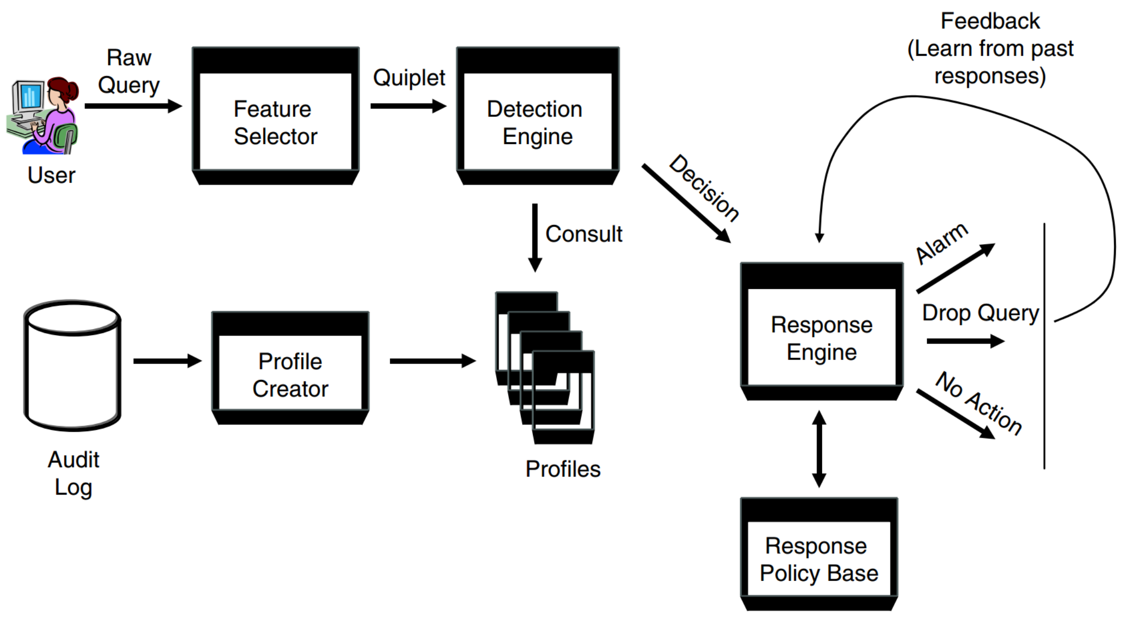

Figure 2.1: System Architecture

2.1 Audit Log

The first component is the audit log. It consists of intrusion-free SQL statements that users issued against the database in the past. These SQL statements are needed to create the initial profiles in the next step.

Scenario Consider the following SQL statements from our running example. The first eight queries were issued by the AD role, searching and modifying specific student records in the database. The last three queries were issued by the RE role, querying emails and addresses from students.

```
 1 - SELECT Students.ID FROM Students
      WHERE Students.LastName="Mueller" AND Students.FirstName="Thomas";
 2 - SELECT Students.LastName, Students.FirstName, Students.Street,
      Students.Number, Students.City FROM Students WHERE ID=1;
 3 - UPDATE Students SET Students.Street="Hirschgeweihstrasse",
      Students.Number=20 WHERE Students.ID=1;
 4 - SELECT Students.LastName, Students.FirstName, Students.Street,
      Students.Number, Students.City FROM Students WHERE Students.ID=1;
 5 - SELECT Students.Email FROM Students WHERE Students.ID=1;
 6 - SELECT Students.ID FROM Students WHERE Students.LastName="Amboise"
      AND Students.FirstName="Claire";
 7 - UPDATE Students SET Students.Street="Schörlistrasse", Students.Number=3,
      Students.City="Zuerich", Students.CityCode=8003 WHERE Students.ID=3;
 8 - SELECT Students.Email FROM Students WHERE Students.ID=3;
 9 - SELECT Students.Email FROM Students;
10 - SELECT Students.Email FROM Students WHERE Students.Departement="Informatics";
11 - SELECT Students.Email FROM Students;
12 - SELECT Students.LastName, Students.FirstName, Students.Street,
      Students.Number, Students.City FROM Students WHERE Students.Departement="Law";
```

2.2 Profile Creator

The profile creator processes the audit log AL to form profiles. The profiles are created by converting the raw SQL statements into a format supported by the IDS, called quiplets. Kamra et al. distinguish between three types of quiplets that extract information at a different level of granularity from an SQL statement: the c-quiplet, the m-quiplet, and the f-quiplet. All quiplets are of the form $Q(c, P_R, P_A, S_R, S_A)$, where the first field c corresponds to the SQL command used. In the coarse or c-quiplet the fields P_R and P_A denote the number of relations, respectively the number of attributes used in the projection clause of the statement. Similarly, fields S_R and S_A denote the number of relations, respectively the number of attributes used in the selection clause of the statement. The medium-grain quiplet, or m-quiplet for short, substitutes the single integer values for fields P_R, P_A, S_R, S_A with vectors of size equal to the number of relations in the database. The relation counters P_R and S_R become bit vectors, where the ith bit is set to 1 if the query projects, respectively selects from the ith relation in the database. The ith position of fields P_A and S_A denotes the number of attributes projected, respectively selected from the ith relation. The third data unit, the fine-grained quiplet or f-quiplet, differs from the m-quiplet by expanding the attribute fields P_A and S_A to be matrixes. $P_A[i, j]$ and $S_A[i, j]$ are set to 1 if the jth attribute of the ith relation is projected, respectively selected in the SQL statement.

Table 2.1: Quiplet Construction Illustration.

SQL command	c-quiplet	m-quiplet	f-quiplet
SELECT Students.LastName, Students.FirstName, Grades.AverageGrade, FROM Students, Courses WHERE Students.ID=Grades.StudentID	Select<2><3><2><2>	Select<1,1><2,1><1,1><1,1>	Select<1,1><[1,1,1,0,0,0,0,0,0],[1,1]><1,1><[1,0,0,0,0,0,0,0,0],[1,0]>

This illustration considers a schema with two relations, namely the Students, and an additional Grades relation containing only two attributes, StudentID and AverageGrade.

Scenario We consider only c-quiplets, because they suffice to show all interesting aspects of the system. The sample audit log gets processed into c-quiplets. We indicate the role belonging to the SQL statement with character being appended to the quiplet. This processed file is the dataset for the classifier.

```
 1 - SELECT<1><1><1><2> AD
 2 - SELECT<1><5><1><1> AD
 3 - UPDATE<1><2><1><1> AD
 4 - SELECT<1><5><1><1> AD
 5 - SELECT<1><1><1><1> AD
 6 - SELECT<1><1><1><2> AD
 7 - UPDATE<1><4><1><1> AD
 8 - SELECT<1><1><1><1> AD
 9 - SELECT<1><1><1><0> RE
10 - SELECT<1><1><1><1> RE
11 - SELECT<1><1><1><0> RE
12 - SELECT<1><5><1><1> RE
```

So far we discussed the components needed to set up the initial profiles. These components will be re-used each time the IDS updates itself as part of the machine learning process. Now the system is ready to receive raw SQL statements to be assessed, issued by specific roles.

2.3 Feature Selector

The Feature Selector is the first stage in the normal workflow of the IDS. It takes a raw SQL statement being issued by a user holding a specific role or roles, and converts it into one of the three quiplet types supported by the IDS. The quiplet, along with the information about a user's role or roles, is passed to the next stage, the Detection Engine.

Scenario The informatics student working for the administration wrote a malicious script in his spare time. He attached it to a PDF file titled "semesterfee.pdf", which he plans to send to all colleagues. For this reason, he issues the SQL statement "SELECT Students.Email

FROM Students;". This statement gets processed by the feature selector into a sample c-quiplet: Select<1><1><1><0> AD.

2.4 Detection Engine

The Detection Engine assesses the quiplet created by the Feature Selector. Using the Naïve Bayes Classifier (NBC), it calculates the probability of this quiplet belonging to a specific role based on the profiles extracted from the audit log . The classifier assigns the quiplet the most likely role and then checks from which role this statement was issued. If the two mismatch, the query gets marked as anomalous. For an incoming quiplet $Q(c, P_R, P_A, S_R, S_A)$, we compute the most likely role r_{ML} from a set of predefined roles R by assessing the quiplet-values (attributes) c, P_R, P_A, S_R, S_A and considering the existing profiles. Thus, we search the most likely role given the quiplet values:

$$r_{ML} = \arg\max_{r \in R} p(r|c, P_R, P_A, S_R, S_A) \tag{2.1}$$

The Bayes Theorem provides a way of calculating the posterior probability $p(r|c, P_R, P_A, S_R, S_A)$ by multiplying the likelihood function $p(c, P_R, P_A, S_R, S_A|r)$ with the prior probability $p(r)$, and dividing it by the marginal likelihood $p(c, P_R, P_A, S_R, S_A)$. The equation can therefore be written as

$$r_{ML} = \arg\max_{r \in R} \frac{p(c, P_R, P_A, S_R, S_A|r)p(r)}{p(c, P_R, P_A, S_R, S_A)} \tag{2.2}$$

When calculating the most probable class value belonging to that query, the value of the marginal likelihood (i.e. the denominator) is a constant, as it is not affected by the choice of r, and can therefore be omitted. The posterior probability is therefore proportional to prior probability times likelihood function:

$$r_{ML} = \arg\max_{r \in R} p(c, P_R, P_A, S_R, S_A|r)p(r) \tag{2.3}$$

Since the NBC assumes conditional independence between attribute values, we can multiply all the distinct attribute probabilities and rewrite the equation as follows:

$$r_{ML} = \arg\max_{r \in R} p(c|r)p(P_R|r)p(P_A|r)p(S_R|r)p(S_A|r)p(r) \tag{2.4}$$

The probability of a quiplet value is based on a frequency count for the quiplet value over the quiplets in the role profile RP_r with role value r. For example, we have $P(c|r) = \frac{n_{c_r}}{|RP_r|}$, where n_{c_r} is the number of quiplets that use command c in the profile of role r and where $|RP_r|$ is the size of the profile for role r. But if any probability calculated this way evaluates to zero, the equation (2.4) is equal to zero and the classifier is biased. To avoid this risk we substitute all conditional probabilities $p(a|r)$ with the m-estimate of the probability, which is formally defines as follows:

$$p_m(a|r) = \frac{n_{a_r} + m \times \frac{n_a}{|AL|}}{|RP_r| + m} \tag{2.5}$$

n_{a_r} denotes the number of times this attribute occurs in the role profile RP_r of role r. n_a is the number of times this attribute occurs in the audit log AL, where |AL| corresponds to the number of quiplets in the audit log. Likewise, $|RP_r|$ denotes the number of quiplets in the profile of role r.The constant m, also called a pseudocount, often takes on a value of 0.5 (as used in this paper) known as Jeffrey's prior approach, or 1, known as Laplace smoothing. Karma et al. used a value of 100 for m, without further explanation why this value was chosen or how a value m should be chosen. In general, choosing one of the two suggested values (m=0.5 or m=1) suffices to prevent probabilities from becoming zero. In case of $n_a = 0$, we assume $p_m(a|r) = \frac{1}{|RP_r|}$.The following shows equation (2.4), extended with the m-estimate of the probability:

$$r_{ML} = \arg\max_{r \in R} p_m(c|r)p_m(P_R|r)p_m(P_A|r)p_m(S_R|r)p_m(S_A|r)p(r) \tag{2.6}$$

Scenario We will apply this formula to the scenario and the sample c-quiplet to be assessed by the Detection Engine, which is Select<1><1><1><0>AD. Since there are two roles r_{AD} andr_{RE} , we calculate the probabilities for both r_{AD} and r_{RE} and therefore identify which role the classifier would assign as r_{ML}. We set the value of the constant m to 0.5 (Jeffrey's prior approach), which suffices to exclude the possibility of any probability rendering to zero. In the following, we first explain the calculations for the administration role. The prior probability for role r_{AD} is given by $p(r_{AD}) = \frac{|RP_{AD}|}{|AL|} = \frac{8}{12} = \frac{2}{3}$. We calculate a conditional probability of, e.g. the command attribute with $p_m(c|r) = \frac{n_{c_AD} + m \times \frac{n_c}{|AL|}}{|RP_{AD}| + m} = \frac{6 + 0.5 \times \frac{10}{12}}{8 + 0.5} \approx 0.7549$. The other attributes are computed in the same way, see figure 3 with the results. Note that both $p_m(P_R|r_{AD})$ and $p_m(S_R|r_{AD})$ in this case result to one, because only one relation exists and therefore the probability that a role chooses this relation in a SQL statement is 100% . The noteworthy characteristic about the SQL command under investigation is that in its selection part no attribute is mentioned, which has never been done by the AD role before. By multiplying all of these values, we get a total posterior probability value of around 0.0025 for the AD role and a total posterior probability value of around 0.1108 for the RE role. As a result, our classifier assigns this quiplet the RE role, observes that the classified role RE and the actual role AD, with which this query was issued, mismatch, and marks the query as anomalous. Note that for simplification purposes, and because it sufficeth to show all relevant parts of the system, we only address the case of an IDS using c-quiplets. For the computation on basis of m- or f-quiplets, we refer the reader to the original paper by A. Kamra et al.

Table 2.2: Probabilities of sample quiplet attributes as calculated by the Naive Baies Classifier

role	p(r)	$p_m(c\|r)$	$p_m(P_R\|r)$	$p_m(P_A\|r)$	$p_m(S_R\|r)$	$p_m(S_A\|r)$	overall
administration	0.6667	0.7549	1	0.5049	1	0.0098	0.0025
rectorate	0.3333	0.9815	1	0.7315	1	0.4630	0.1108

2.5 Response Engine

The Response Engine receives the assessment of the query from the Detection Engine and triggers an adequate action within the system, by consulting the policy base of existing response mechanisms, the Response Policy Base. The Response Engine can be configured to take different actions in case of an anomaly. Some possible actions in conjunction with other systems would be to send an alert to the administrators, drop the query, log the query with other information in a special security log, disconnect the user or disable a whole role. In case where the Detection Engine reports the query as being non-anomalous, it can forward the query to the database. Moreover, as part of its machine-learning mechanism the IDS adds the query together with the role information to the database audit log and triggers the Profile Creator to update the existing profiles.

3 Shortcomings

The scenario we discussed so far showed a working example where the resulting output, the alarm being raised, matched our expectation. To differentiate between the different actions being taken by the IDS and our personal assessment of the result, we will use the following terminology: A true positive TP is when the IDS correctly (true), based on our own judgment, marks a quiplet as anomalous (positive). A false positive FP, on the other hand, corresponds to a false alarm where, the IDS marks a quiplet as being anomalous (positive), whereas we deem the query as being of non-malicious nature. In the same way, a true negative TN is a quiplet of non-malicious nature and being correctly assessed by the system as non-anomalous, whereas a false negative FN is a query that should have raised an alarm, but was deemed by the classifier as non-anomalous.

Next we discuss two scenarios in which the IDS makes a false decision. In one case the system raises an alarm for a benign quiplet (i.e. a false positive), and in the other case the system raises no alarm for a malignant quiplet (i.e. a false negative).

3.1 False Positive

A false positive is provoked whenever a benign SQL statement submitted by one role shows a specific behavior that is typical for another role. So if there are two roles defined, that naturally show similar access patterns, chances are that the IDS will always favor one role and block the other role from executing statements with this behavior. Since only queries categorized as non-anomalous can influence the classifier through its machine-learning updating mechanism, the access-behavior will be permanently assigned to a specific role, based on the audit log.

To provoke a false positive (FP) in our running example, we extend our previously used scenario by a third user role, the “Creator” (CR) which creates all the new student records in the database. Typical behavior for this role would be to primarily use the “INSERT” SQL command. We extend our previous scenario with the following logs:

```
 1 - UPDATE Students SET Street="Zelglistrasse", Number=15
WHERE Students.LastName='Ankli' AND Students.FirstName='Lea'
AND Students.City='Zuerich' ; AD
 2 - INSERT INTO Students VALUES ("Huber", "Stephanie", "Fliederweg", "6",
"Holziken", "5043", "stephanie.huber@uni.ch", "Economy"); CR
 3 - INSERT INTO Students VALUES ("Müller", "Franz", "Albisstrasse", "1",
"Zuerich", "8038", "franz.mueller@uni.ch", "Law"); CR
 4 - INSERT INTO Students VALUES ("Baker", "John", "Abendweg", "1",
"Zuerich", "8005", "john.baker@uni.ch", "Economy"); CR
 5 - INSERT INTO Students VALUES ("Ahluwalla", "Rahul", "Hagenbuchrain", "4",
"Zuerich", "8047","rahul.ahluwalia@uni.ch","Informatics"); CR
```

Now the specific case happens that actually a misspelling in the name took place in the last insert statement. The user of this role CR catches the error and tries to correct it with another SQL query, this time an Update query. This becomes therefore the next query to be assessed by the IDS: “UPDATE Students SET Students.Lastname=”Ahluwalia” WHERE Students.Firstname=”Rahul” AND Students.Street=”Shilstrasse” AND Students.Number=4;”. The Feature Selector generates this query: Update<1><1><1><3> CR. Then the Classifier calculates the rML with the updated profiles:

Table 3.1: Probabilities of FP quiplet attributes as calculated by the Naive Baies Classifier

role	p(r)	$p_m(c\|r)$	$p_m(P_R\|r)$	$p_m(P_A\|r)$	$p_m(S_R\|r)$	$p_m(S_A\|r)$	overall
administration	0.5294	0.3251	1	0.4427	1	0.1084	0.0083
rectorate	0.2353	0.0196	1	0.7124	1	0.0065	2.1E-05
creator	0.2353	0.0196	1	0.0458	1	0.0065	1.4E-06

The NBC calculates the AD role to be the most likely associated with this query. When consulting the probability-values shown in Figure 5, we derive the following reasons: any query entering the system in this exact setting is most likely to come from the AD role according to p(r), since nine out of seventeen statements already stored in the system stem from this role. The value of the next probability $p(c|r)$ is also highest for the AD role, since alle UPDATE statements in our AL are so far exclusively from this role. The $p(P_A|r)$ probability is highest for the RE role, since the RE role had the highest ratio of queries with only projecting on one attribute, namely four out of five. The biggest impact although stems from the SA-probability, since there is only one statement in the AL that selected on exactly three attributes. The IDS therefore assigns the most likely role of AD to this query and deems the query to be anomalous since the role issuing the query is CR. The Response Engine would now consult the Response Policy Base, raise an alarm and drop the query, depending on our configuration. This is an undesired behavior, since we deem this query not to be of malicious nature but actually a desired behavior in order to ensure a certain quality of the data. We therefore would see this as a false alarm. In such cases, it would be important to ensure that desired behavior of specific roles, although it might be atypical behavior, get defined and stated in the Response Policy Base for the IDS to not issue FP responses.

3.2 False Negative

To find a false negative, meaning a query that did not raise an alarm but is deemed as not desired behavior of a role, we have a look at our scenario again. In this scenario, the student wants to get all the Emails from all students to send his malicious script to. This is why he issued the query “SELECT Students.Email FROM Students;”. This query then went through our IDS, got assessed as anomalous and was dropped. The student recognizes that his query got dropped since he did not receive any results. But since he still has access, he decides to rewrite his initial query so that it looks more like a query that his role would issue, but that still returns the same result as the blocked query. His new query is “SELECT Students.Email

FROM Students WHERE Students.ID >= 0;". The resulting quiplet, generated by the Feature Selector, is Select<1><1><1><1>. Remember that our system does not update itself with anomalous queries, so the audit log and therefore the profiles have not changed. The classifier assigns the most likely role of AD, as shown in Figure 6. Since the student's role is AD, the query is deemed non-anomalous and no alarm is raised.

Table 3.2: Probabilities of FN quiplet attributes as calculated by the Naive Baies Classifier

role	p(r)	$p_m(c\|r)$	$p_m(P_R\|r)$	$p_m(P_A\|r)$	$p_m(S_R\|r)$	$p_m(S_A\|r)$	overall
administration	0.6667	0.7549	1	0.5049	1	0.7451	0.1893
rectorate	0.3333	0.9815	1	0.7315	1	0.5185	0.1241

The determining factor for classifying the the student's initial query as anomalous in the scenario was that the initial query did not select any attributes in its WHERE clause This was assessed as being rather atypical for the AD role, compared with the SE role. Since the student has changed the number of Selected Attributes in his second query, this time the query gets assessed as non-anomalous and the student receives his desired email-addresses. What we would like is our IDS to raise an alarm since this is still batch retrieval and undesired behavior of the AD role. In this specific case, the alarm had already been raised and the administrators will know what has happened. The student still got what he wanted, because he did still have access. We could of course instantly block the access of a specific role as soon as an alarm is raised by the IDS, but this is not practical because of FPs and the time it takes for the administrators to assess and investigate every alarm raised before granting access again, this resulting in lost time and therefore big costs generated in large companies. If the attacker could generate an FN without an FP beforehand as in this case, his attack would go unnoticed for probably a long time or even never getting noticed. The attackers disadvantage is that he has no knowledge of the specific values our classifier uses, since no database-role should have access to the AL, and therefore an attacker can never be certain not to raise an alarm. The information an eventual wrongdoer has, are the queries he normally issues. Therefore the wrongdoer will craft a query that will get the desired result but will be as close to the structure of the expected behavior of his role. This IDS cannot really protect against rewriting or crafting queries to simulate expected behavior but still be of malicious user. We would need to specify a white or blacklist within the Response Policy Base to minimize the risk of such attacks.

4 Summary

We studied the inner working and the concept on which the Intrusion Detection System of Kamra et al. is built upon. We have illustrated a scenario in a university-like setting and assessed a number of incoming SQL queries using the Naive Bayes Classifier as described by Kamra et al. We also had a look at some ways where the system cannot successfully prevent an attack or might also interfere with desired behavior of authorized users. In general, the approach of Kamra et al. for an Intrusion Detection System as a separate component that extends traditional relational databases is useful, should be rather easily deployable and is worth further investigation. As shown in this paper, the biggest risk comes from False Negatives, were successful attacks happen that go unnoticed. False Positives can generate additional labor and generate costs for administrators that need to investigate every benign query for which the IDS raised an alarm . Also apparent in the examples is that our IDS cannot prevent well designed attacks like simulating expected behavior, therefore there need to be additional security mechanisms deployed for the database to cover such cases.