An Anomaly Analysis Framework for Database Systems

Sokratis Vavilis[a,*], Alexandru Egner[a], Milan Petković[a,b], Nicola Zannone[a]

[a] *Eindhoven University of Technology, Postbus 513, 5600 MB Eindhoven, Netherlands*
[b] *Philips Research Eindhoven, High Tech Campus 34, Eindhoven 5656AE, Netherlands*

Abstract

Anomaly detection systems are usually employed to monitor database activities in order to detect security incidents. These systems raise alerts when anomalous activities are detected. The alerts raised have to be analyzed to timely respond to the security incidents. Their analysis, however, is time-consuming and costly. This problem increases with the large number of alerts often raised by anomaly detection systems. To timely and effectively handle security incidents, alerts should be accompanied by information which allows the understanding of incidents and their context (e.g., root causes, attack type) and their prioritization (e.g., criticality level). Unfortunately, the current state of affairs regarding the information about alerts provided by existing anomaly detection systems is not very satisfactory. This work presents an anomaly analysis framework that facilitates the analysis of alerts raised by an anomaly detection system monitoring a database system. The framework provides an approach to assess the criticality of alerts with respect to the disclosure of sensitive information and a feature-based approach for the classification of alerts with respect to database attacks. The framework has been deployed as a web-based alert audit tool that provides alert classification and risk-based ranking capabilities, significantly easing the analysis of alerts. We validate the classification and ranking approaches using synthetic data generated through an existing healthcare management system.

Keywords: Anomaly Detection, Data Leakage, Risk Assessment, Database Attack Classification, Alert Visualization

*Corresponding author

Email addresses: s.vavilis@tue.nl (Sokratis Vavilis), a.i.egner@tue.nl (Alexandru Egner), milan.petkovic@philips.com (Milan Petković), n.zannone@tue.nl (Nicola Zannone)

1. Introduction

Database management systems (DBMS) are used to collect, store, disseminate, and analyze sensitive data such as customer records and confidential business information. This makes DBMS an attractive target for attackers. A study conducted by Verizon in 2014 shows that DBMS are one of the most compromised assets of organizations [1]. Therefore, they are an essential asset that needs to be protected. However, it is very difficult, if not impossible, to prevent database attacks completely. Thus, prevention solutions are often coupled with detection and incident response management solutions. These solutions, however, should satisfy stringent constraints on how security incidents are handled. For instance, the new EU data protection regulation obliges organizations to take actions and notify data protection authorities within 24 hours after the occurrence of a data breach is detected [2]. Thus, organizations need solutions that allow *timely* detection and response to security incidents.

The detection of security incidents in DBMS is often addressed by employing anomaly detection systems [3, 4, 5, 6, 7]. Anomaly detection systems monitor user database activities (defined in terms of database queries) and raise alerts when suspicious activities are detected. Various techniques have been proposed to detect anomalies (see [8] for a survey). In broad terms, anomaly detection systems can be classified into signature-based and behavior-based. Signature-based approaches [9, 10] use predefined rules to specify activity patterns that correspond to anomalous behavior. These approaches, however, are only able to detect known attacks, leaving unknown attacks undetected. This issue is addressed by behavior-based approaches [3, 11], which automatically learn user profiles by observing "normal" activities and marking every deviation from such profiles as a potential threat.

Alerts raised by anomaly detection systems should be investigated before deciding the actions to be taken in order to respond to the security incidents [12]. There are two main challenges to be faced when analyzing alerts:

- *Gathering of alert information*: Deciding how to respond to a security incident requires a deep understanding of the alerts. Such an understanding is only possible if relevant information about alerts, such as *root causes* and *attack type*, are known. However, gathering such information manually is costly and time-consuming.

- *Large number of alerts*: The number of alerts raised by the anomaly detection system can be huge. For example, a large number of alerts can be raised in hospitals due to the usage of the break-the-glass protocol [13]. In

addition, existing anomaly detection systems and, in particular, behavior-based anomaly detection systems are often characterized by a high rate of false positives [14, 15], i.e. alerts that are not actual security incidents. Identifying the most critical incidents among a large number of alerts requires substantial efforts.

To facilitate the analysis of alerts, it is thus desirable to have means for *(i)* gathering the information necessary for the analysis and *(ii)* prioritizing alerts with respect to their criticality.

Unfortunately, most of the existing anomaly detection systems provide no or very limited information about alerts [16], thus failing to meet desideratum *(i)*. Few proposals [3, 5] give intuition on how to enrich alerts with relevant information, but do not provide a systematic approach for gathering such information. Some approaches in the field of anomaly-based intrusion detection [17, 18, 19] provide attack classifications of alerts. However, these approaches are mainly tailored for the analysis of network traffic and are not suitable for the analysis of database activities.

To identify the most severe security incidents (desideratum *(ii)*), few approaches provide support for the *quantification* and *ranking* of alerts based on their criticality. In particular, some approaches [3, 20] quantify alerts with respect to their anomaly level, allowing security officers to focus on the most abnormal cases. Other approaches [21, 22] aim to provide an estimation of the damage caused by data leakages on the basis of the sensitivity and amount of leaked information, allowing security administrators to focus on the most severe data breaches. These approaches, however, only provide a "partial view" on the criticality of alerts, which may lead to a ranking that does not reflect their actual criticality level. The criticality of security incidents is typically measured in terms of *risk*, which is defined as a function of the likelihood (probability) of an undesired event occurring and the impact (severity) that such an event has on an organization [23].

In this work, we propose a unified framework to facilitate the analysis and evaluation of alerts raised by an anomaly detection system monitoring user database activities. The framework aims to enrich alerts with information necessary for their analysis. First, we propose a risk-based data leakage quantification approach that allows an estimation of the criticality of alerts with respect to the disclosure of sensitive data. To provide a complete view on the criticality of alerts our risk metric encompasses both the probability of an alert to be a true alert and its severity with respect to the amount and sensitivity of the data leaked. For estimating the probability that an alert corresponds to a true security incident, we adopt and extend the white-box behavioral-based anomaly detection system presented in [3]. This system provides an anomaly score together with the alerts raised, which indicates

whether they are a real threat or not. In our work, we revise how the anomaly score is computed in order to express such a score as a probability. To calculate the severity of data leakages, we employ the data leakage quantification approach presented in [22]. We refer to Section 2 for an overview of these two approaches. Second, we propose a novel method for the classification of alerts with respect to database attacks. We study the most frequent database attacks [24] and elicit the features characterizing these attacks. We show how these features can be used to specify rules representing attack patterns. Based on these rules, we present an approach to classify alerts with respect to database attacks. To enable a timely analysis and assessment of alerts, we present a database anomaly audit tool for the visualization of alerts. In particular, the tool shows the relevant information about alerts, including their criticality level (with respect to data leakage) and attack type in a user-friendly manner. Thus, the tool facilitates alert analysis by making it possible to focus on the most critical data leakages or on specific types of database attacks.

We validate the risk-based data leakage quantification approach and the feature-based attack classification method using a case study in the healthcare domain. Healthcare is indeed an interesting domain to investigate as a large amount of sensitive data, such as patient healthcare records, have to be protected. For the experiments, we have generated, together with our industry partner, synthetic datasets of database queries representing normal and anomalous activities that typically occur in a hospital. Experimental results show that the risk-based data leakage quantification approach provides a more accurate estimation of the criticality level of alerts compared to existing alert quantification approaches. Moreover, our results show that the feature-based attack classification method is able to identify the attack type of an alert with high recall and acceptable precision. Nonetheless, the evaluation of the classification method shows that the quality of the classification can be improved by characterizing attack types using a richer set of features.

The remainder of the paper is organized as follows. The next section presents preliminaries on the SQL query language and the building blocks of our quantification approach. Section 3 presents the risk-based data leakage quantification approach, while Section 4 presents the feature-based attack classification method. Section 5 presents an audit tool for the visualization and analysis of alerts along with its potential use. The evaluation of the proposed approaches is presented in Section 6. Finally, Section 7 discusses related work, and Section 8 concludes the paper providing directions for future work.

2. Preliminaries

In this section we present the preliminaries of our work. First, we provide insights on SQL, which is used to represent database activities. Then, we present

```
select_command    ::=
    SELECT ( "*" | ( displayed_column { "," displayed_column } ) )
    FROM ( selected_table { "," selected_table } )
    [WHERE condition ]
    [UNION select_command ]

displayed_column ::= column_name | Nested Query
selected_table ::= table_name | Nested From Query
condition ::= logical_term |Nested Where Query
```

Figure 1: Syntax of SELECT queries

an overview of the white-box behavioral-based anomaly detection system [3] used for the estimation of the anomaly level of alerts and an overview of the data leakage quantification approach [22] used to calculate the severity of data leakages.

2.1. SQL

SQL is a query language largely used in DBMS. SQL allows users to manipulate the data stored in a database using *queries*. In particular, users can access, add, modify or remove data from the database using SELECT, INSERT, UPDATE and DELETE SQL queries respectively.

A typical SQL query consists of different parts (called *clauses*), where the SQL command (e.g., SELECT, UPDATE) in the main clause indicates the type of query. An SQL query can contain nested queries (subqueries) inside the different clauses. As an example, we present in Fig. 1 the syntax of SELECT SQL queries, the most frequently used and arguably most complex SQL statement. SELECT queries can contain multiple nested queries in every clause of the query and can be extended with additional SELECT queries using the UNION command. Hereafter, we refer to the entire query (i.e., the set of clauses associated with a query) and the nested queries in it as *Query (Q)* and *Nested Queries (NQ)* respectively.

The structural characteristics of other SQL queries (i.e., INSERT, UPDATE, DELETE) are similar but simpler than the ones of SELECT queries. For instance, DELETE queries consist of a mandatory FROM clause, which may contain only a database table name, and an optional WHERE clause. UPDATE and DELETE queries may contain nested SELECT queries usually in the WHERE clause. On the other hand, INSERT queries usually contain nested SELECT queries as part of the main clause.

2.2. Computing the anomaly level of alerts

To compute the anomaly level of database activities, we adopt and extend the white-box behavioral-based anomaly detection system proposed in [3]. In this sec-

tion, we present an overview of this system and motivate its adoption; in Section 3 we show how it can be extended to assess the anomaly level of alerts in terms of probabilities.

The white-box behavioral-based anomaly detection system in [3] aims to detect and rank alerts within DBMS by assessing their anomaly level, i.e. to what extent database activities are anomalous. It uses an anomaly detection model that automatically learns the normal behavior by observing database activities and flags deviations from such a behavior as anomalies. Alerts are quantified with respect to their anomaly level, allowing security administrators to focus on the most anomalous cases. This anomaly detection system offers a number of advantages compared to other anomaly detection systems. First, while being able to detect unknown attacks because of its behavioral-based design, it results in a lower false positive ratio compared to other behavioral-based approaches. Moreover, to our knowledge, it is the only white-box behavioral-based approach designed for DBMS. In contrast to existing black-box solutions (e.g., [25, 26]), which only flag a query as anomalous or not, the white-box approach enables the extrapolation of the root causes of alerts. This additional information about alerts is at the basis of the feature-based attack classification presented in Section 4.

Database activities are represented by the SQL queries sent to the DBMS. Queries are described using a set of *features*; each feature is used to characterize a particular aspect of the queries (Table 1). Features are divided into three groups: *syntax-centric* features which characterize aspects related to the specification of the queries (e.g., SQL command, table list), *context-centric* features which characterize the context within which queries are posed (e.g., IP address, time), and *result-centric* features which describe data-centric information related to the response to a query (e.g., number of records retrieved).

Normal database activities are described using *profiles*, which characterize the database activities performed by an entity (e.g., a user or a role) in terms of (a subset of) features. Profiles are generated from a training data set using a learning algorithm and represented as a set of histograms, one for each feature. A histogram h_i for a feature f_i is a set of bins that partition the queries in the training data set (made by the entity to be profiled) based on the values that f_i can take. A *bin* (v, f_i) is characterized by the frequency of value v in the training set (denoted $freq(v, h_i)$) and a threshold $t(v, h_i)$ representing the minimum probability the bin should satisfy in order to be considered normal. Note that different features can have different data types (e.g., nominal, numeric, time) and could have wide ranges. To meaningfully capture the normal behavior, feature values are grouped into bins. For instance, numeric values can be expressed using ranges of values; time values using time intervals, etc.

To deal with the different size of different profiles (i.e., the number of queries

ID	User	Command	Table List	Column List	Query Length	IP address	Time	# Records
1	Bob	SELECT	Patients	Name, Age, Sex	250	192.168.1.6	15:00	60
2	Bob	SELECT	Patients, Treatment	Name, Age, Patient_id, Medicine	200	192.168.1.6	10:00	50
3	Bob	SELECT	Patients	Name, Sex	100	192.168.1.5	12:00	100
4	Bob	SELECT	Treatment	Patient_id, Medicine	200	192.168.1.6	16:00	200
5	Bob	INSERT	Patients	Name, Age, Sex	300	192.168.1.4	17:00	1
6	Bob	SELECT	Patients	Name, Sex	150	202.201.5.11	18:00	60
7	Bob	SELECT	Treatment	Patient_id, Medicine	130	192.168.1.6	09:00	40
8	Bob	INSERT	Patients	Name, Age, Sex	350	192.168.1.6	14:00	1
9	Bob	SELECT	Patients	Name	100	192.168.1.3	13.00	10
10	Bob	INSERT	Treatment	Medicine	160	192.168.1.6	14.00	1

Table 1: Characterization of database queries with respect to features

posed by the different entities), histograms are normalized into a probability distribution. The probability of a value v for a feature f_i in histogram h_i is

$$prob(v,h_i) = \frac{freq(v,h_i)}{N} \tag{1}$$

where N represents the number of queries performed by the profiled entity in the training data set. The probability of values together with the threshold is used to define anomalous feature values: if the bin probability is lower than the threshold (i.e., $prob(v,h_i) < t(v,h_i)$), the bin is marked as anomalous; otherwise, it is marked as normal.

Fig. 2 shows the profile representing Bob's normal activity behavior based on the database queries presented in Table 1. In the figure, the domains of features query length, IP address and time are expressed as ranges of values. For instance, an IP address range (i.e., 192.168.x.x) is used to indicate that database queries were posed within the local network of the organization.

Once profiles have been defined, they are used in the anomaly detection model to detect anomalies in database activities. During detection, new queries made by a user are analyzed and matched against the corresponding profile. In particular, each feature value of a query is compared against the corresponding histogram in the profile. A feature value is marked as anomalous if it falls in an anomalous bin.

A query is considered to be anomalous if at least one of its feature values is marked as anomalous. For each anomalous query, an *anomaly score* is calculated to express the anomaly level. The anomaly score of a query Q is

$$AnomalyScore(Q) = \prod_{f_i \in AF(Q)} \frac{1}{prob(v,h_i)} \tag{2}$$

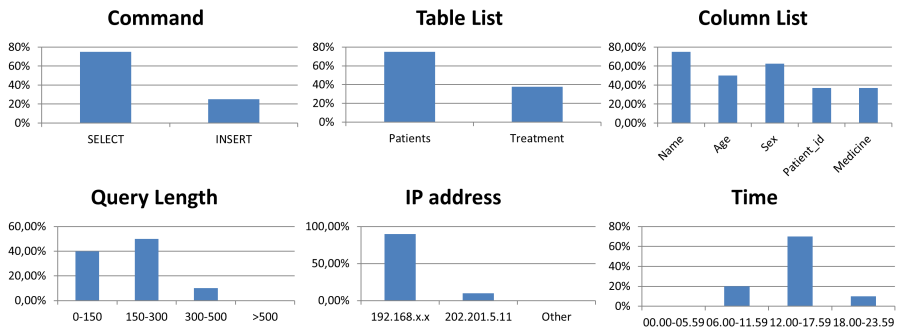

Figure 2: Histograms for database queries in Table 1

where $AF(Q)$ denotes the set of anomalous features in the query. Note that the approach requires a minimum probability (greater than 0) for feature values which do not occur in the training set to avoid division by zero. Moreover, the calculation of the anomaly score requires features to be independent from each other.

2.3. Computing the severity of alerts

To our knowledge, only two solutions [21, 22] address the problem of quantifying the severity of data leakages in database environment. In particular, these solutions aim to provide an estimation of the damage that data leakages cause to the organization hosting the data. They are connected to a data leakage detection system:[1] upon receiving an alert, these solutions assess the severity of the alert on the basis of the semantics (e.g., data sensitivity, identifiability) and amount of the leaked information. Potential data leakages are ranked with respect to their severity. In the remainder of this section, we present an overview of the data leakage quantification approach presented in [22]. As shown in [22], this approach is able to discriminate alerts more accurately compared to the approach presented in [21].

Let $\mathcal{A}$ be a set of attributes. A table $D(a_1,\ldots,a_n)$ is a set of records over a set of attributes $\{a_1,\ldots,a_n\} \subseteq \mathcal{A}$. The records in $D(a_1,\ldots,a_n)$ are denoted as $R^{D(a_1,\ldots,a_n)}$. Given a record $r \in R^{D(a_1,\ldots,a_n)}$, $a_i[r]$ represents the value of attribute a_i in r.

The severity of data leakages is assessed on the basis of three aspects, namely the sensitivity, distinguishing factor and amount of the leaked data. The sensitivity of a piece of information reflects the (privacy) impact and costs caused by disclosing that piece of information according to the data subject/owner or to the organiza-

[1]Data leakage detection systems can be seen as an instance of anomaly detection systems in which the focus is on the anomalies that may cause data leakage.

tion hosting the data. The sensitivity of a leaked record is determined by summing the sensitivity of the attributes values occurring in the record. Let $L(a_1,\ldots,a_n)$ be a leaked table defined over the set of attributes $A = \{a_1,\ldots,a_n\} \subseteq \mathcal{A}$ with records $R^{L(a_1,\ldots,a_n)}$. The sensitivity of a record $r \in R^{L(a_1,\ldots,a_n)}$ is calculated as

$$RecordSensitivity(r) = \sum_{a_i \in A} Sensitivity(a_i[r]) \tag{3}$$

where $Sensitivity(a_i[r])$ represents the sensitivity of attribute value $a_i[r]$. Eq. (3) requires that a sensitive value is defined for each piece of information. This, however, is a costly and time consuming task. To assign a sensitivity value to every piece of information in the domain, the approach in [22] uses a data model representing the domain knowledge to reason on data sensitivity. The data model offers sensitivity propagation and inference mechanisms to facilitate the annotation of the data model with sensitivity values from a partial assignment of sensitivity values. We refer to [22] for details on the data model and mechanisms for reasoning on data sensitivity.

The severity of a leaked table is determined on the basis of the sensitivity of the records in the leaked table along with a distinguishing factor. The distinguishing factor (identifiability) is a weighting factor representing the ability to identify the data subject related to the leaked data, i.e. to what extent the leaked data can be linked to an individual. Given a record r, $DF(r) \in [0,1]$ denotes the distinguishing factor of r over the database. The severity of a leaked table $L(a_1,\ldots,a_n)$ with records $R^{L(a_1,\ldots,a_n)}$ is given by

$$Severity(L(a_1,\ldots,a_n)) = \sum_{r \in R^{L(a_1,\ldots,a_n)}} DF(r) \times RecordSensitivity(r) \tag{4}$$

3. Risk-based Data Leakage Quantification

Risk is a widely used metric for the prioritization of security incidents with respect to their criticality. In particular, the concept of risk has been introduced to assess and evaluate the potential losses caused by incidents. Risk is defined in ISO/IEC Guide 73 [23] as the combination of the likelihood of an event and its consequence. Intuitively, risk depicts the criticality of an event in disrupting the system [27].

The approaches for alert quantification presented in Section 2 only provide a "partial view" on the criticality of security incidents. The anomaly score is mainly tailored to the detection of anomalies by assessing the anomaly level of an alert to be an anomalous activity. Severity assesses the impact of data breaches, but it does not consider the uncertainties associated with the security incidents. Thus,

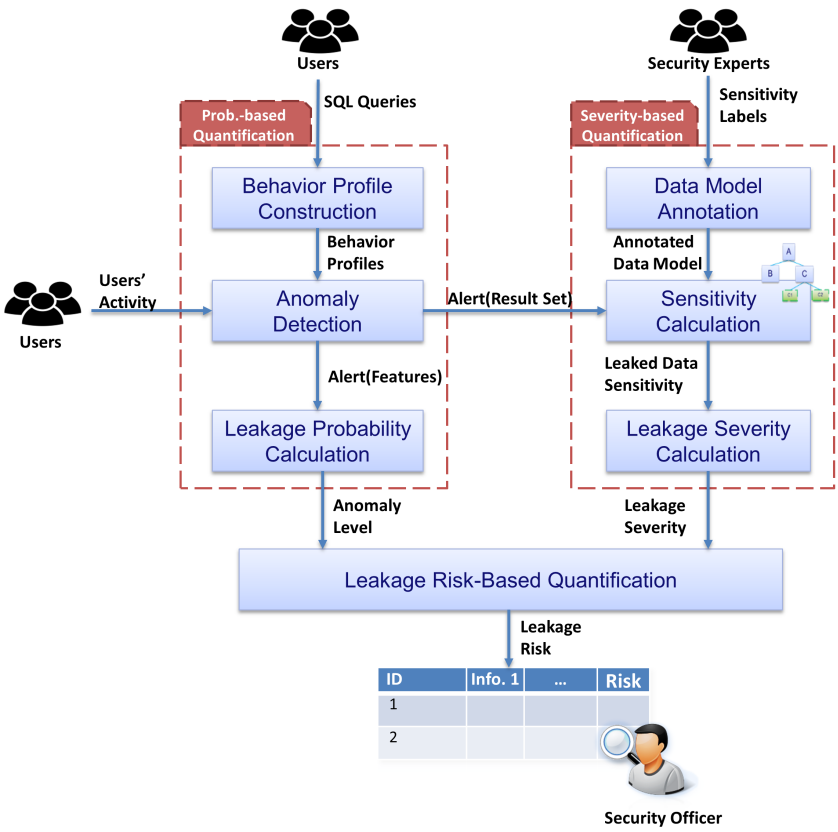

Figure 3: Risk-based Data Leakage Quantification

these partial views can lead to a ranking of alerts that may not reflect their actual criticality.

To enable a risk-based ranking of alerts, we propose an approach that encompasses probability-based and severity-based alert quantification (Fig. 3). In particular, to calculate the anomaly level of alerts, we use a white-box behavioral-based approach similar to the one presented in Section 2.2. We use the data leakage quantification approach presented in Section 2.3 to assess the severity of alerts with respect to the disclosure of sensitive data. The combination of these approaches makes it possible to better characterize the impact of data leakages on an organization and enables the ranking of alerts based on their criticality. Note that we aim to quantify alerts related to data leakages and, thus, we restrict the analysis to SELECT SQL queries.

To detect data leakages and determine their criticality, we build behavior profiles representing normal user activities. In particular, the histograms forming be-

havior profiles are constructed based on a training data set that contains queries representing the normal user behavior. Data leakages are detected by the anomaly detection component by comparing the feature values of new SQL queries against the behavior profiles constructed during the training phase. User queries that have at least one feature value in an anomalous bin are marked as anomalous. An alert is raised for each anomalous activity. The alerts are then quantified with respect to their anomaly level and severity.

To determine the anomaly level of alerts, we propose a new metric which aggregates the probabilities of feature values (Eq. 1). The value probability $prob(v,h_i)$ determines the probability that value v occurs. Given an anomalous feature f_i in histogram h_i, we normalize the probability of a feature value v of f_i with respect to the bin threshold $t(v,h_i)$. In particular, the normalized probability is defined as

$$\pi(v,h_i) = \frac{prob(v,h_i)}{t(v,h_i)} \tag{5}$$

It is worth noting that if the value probability $prob(v,h_i)$ is close to the threshold $t(v,h_i)$, the normalized probability is close to 1. On the other hand, if v never occurred in the training data set, its value probability is equal to 0 and, thus, is the normalized probability. We use the normalized probability of (anomalous) feature value to determine the anomaly level of a query. Given an anomalous query Q, the anomaly level of Q is given by:

$$AnomalyLevel(Q) = 1 - \prod_{f_i \in AF(Q)} \pi(v,h_i) \tag{6}$$

where $AF(Q)$ is the set of anomalous features in Q. It can be observed in Eq. (6) that queries having at least one feature value that was never met in the training data set are associated with an anomaly level equal to 1. This is not surprising as these queries are most likely anomalous. Moreover, considering multiple anomalous feature values leads to a higher anomaly level that considering the same anomalous feature values individually.

The alerts raised by the anomaly detection component are also quantified with respect to their severity. To this end, we employ the approach presented in Section 2.3. In particular, severity assessment of data leakages is performed based on a data model representing the domain knowledge annotated with sensitivity labels. The risk of data leakages is computed based on both their anomaly level and severity. Let Q be an anomalous query whose result set is table $L(a_1,\ldots,a_n)$. The risk level of Q is given by:

$$Risk(Q) = AnomalyLevel(Q) \times Severity(L(a_1,\ldots,a_n)) \tag{7}$$

ID	User	Comm.	Table List	Column List	Query Length	IP address	Time	# Records	Anomaly Score	Severity	Risk (Anom. Level)
1	Bob	SELECT	Patients	Name, Age, Sex	300	192.168.1.6	01:00	1	2000	50	50 (1)
2	Bob	SELECT	Patients, Treatment	Age, Medicine	170	202.201.5.11	22:00	10	100	170	143 (0.84)
3	Bob	SELECT	Treatment	Patient_id, Medicine	160	192.168.1.6	22:00	20	10	1900	1140 (0.60)
4	Bob	SELECT	Patients	Name	60	192.168.1.12	08:30	60	5	900	180 (0.20)
5	Bob	SELECT	Treatment	Patient_id, Medicine	400	202.201.5.11	19:00	9	1000	855	804 (0.94)

Table 2: Risk-based ranking of database queries. The anomaly score is computed using Eq. (2), leakage severity using Eq. (4), risk level using Eq. (7), and anomaly level using Eq. (6).

where $AnomalyLevel(Q)$ represents the anomaly level of Q (computed using Eq. (6)), and $Severity(L(a_1,\ldots,a_n))$ represents the severity of the data leakage (computed using Eq. (4)).

An example of the application of our approach is presented in Table 2. The left part of the table shows a number of anomalous queries along with their features; anomalous feature values are underlined in the table. For each query, the anomaly score, leakage severity, and risk level along with the corresponding anomaly level (in parenthesis) are presented in the right part of the table. The calculation of the anomaly score and anomaly level is based on the behavior profile in Fig. 2. To calculate the severity of data leakages we assume that patients can be uniquely identified by their name; thus the distinguishing factor is equal to 1. In addition, demographic data such as name, age and sex, are labeled with a low sensitive value. On the other hand, information about medicine is labeled with a medium sensitive value (e.g., Paracetamol) or a high sensitive value (e.g., ARV medication).

As shown in Table 2, the anomaly score does not properly quantify the criticality of data leakages. For instance, although the first alert has a higher anomaly score, it does not correspond to a critical data leakage. Indeed, the alert corresponds to the request of only one patient's name, age and sex to the database. Moreover, the third alert, which corresponds to the leakage of patients' name and prescribed medication, has a very low anomaly score, while its severity (and thus its risk level) is very high due to the amount and sensitivity of the information leaked. Considering severity alone can also provide misleading information about alerts. For instance, the forth alert is ranked higher than the fifth alert with respect to severity, although the later is obviously more anomalous (and its risk level higher). These issues are addressed by the risk-based quantification approach. In particular, the risk metric characterizes alerts better than the anomaly score and severity by combining the anomaly level and severity of alerts. For instance, the risk metric provides a better estimation of the first alert. Moreover, the ranking of the fourth

and fifth alerts with respect to risk takes into account the uncertainties associated with them providing a better assessment of the criticality of the incidents. Last but not least, it is worth noting that the anomaly level used for the computation of the risk level preserves the ranking induced by the anomaly score. Thus, the risk-based quantification approach correctly accounts for the uncertainties regarding the detected security incidents.

4. Feature-based Analysis of Attacks

The approach presented in Section 3 makes it possible to focus on the most critical alerts. However, the investigation of anomalous activities requires considering additional information about alerts and, in particular, their root causes. For instance, security officers may be required to take actions on the basis of the type of attack that was performed. In this section we present a feature-based approach for the classification of alerts with respect to database attacks. First, we discuss the most frequent attacks on DBMS. Then, we elicit the features that characterize such attacks and present the alert classification approach.

4.1. Attack Taxonomy

Attacks refer to security incidents that can violate the security or any other desired functionality of the system [28]. In particular, an attack represents the actions that an attacker performs to compromise the availability, integrity and confidentiality of a DBMS. In this work we consider the most frequent database attacks [24]. In particular, we focus on attacks that can be observed by analyzing database activities. Therefore, attacks which occur at the network or web-application level, such as attacks to data storage and back-up files or exploitation of buffer overflow vulnerabilities, are out of the scope of this work.

Next we discuss the most frequent attacks against a DBMS.

- **Privilege misuse**: Database users can abuse legitimate privileges assigned to them to insert, acquire or alter information that they are not authorized to access [24]. Privilege misuse often occurs when excessive privileges are granted to users. For instance, a database administrator may access sensitive data, such as patient disease information, although he is not supposed to.

- **Masquerade attack**: An attacker can impersonate a database user to gain access to the DBMS and thus be able to send queries to it [5]. However, the context, in which malicious queries are performed, is usually different from the context in which queries are performed by a legitimate user. For instance, an authorized user might send his queries within the organization premises

during regular working hours, while the attacker might send the queries from another IP address during the night.

- **SQL Injection**: An attacker may attempt to bypass authentication mechanisms and acquire unrestricted access to the DBMS by executing malicious code [24, 29]. To this end, an attacker often manipulates queries by injecting a special crafted SQL code (usually in the WHERE clause of the query).

- **Resource Consumption**: An attacker may prevent legitimate users from accessing the DBMS by consuming database resources such as available memory or CPU utilization and, thus, making the DBMS unavailable (*denial of service*). To this end, an attacker can request a large amount of data (in terms of the size of the result set) or force the DBMS to perform complex and time-consuming calculations on large datasets.

- **Error-code Analysis**: An attacker may attempt to acquire information related to the DBMS by sending not well-formed SQL queries or requesting the execution of actions not supported by the DBMS. From the returned error messages the attacker can retrieve information about the used DBMS.

- **Password Guessing**: An attacker can attempt to authenticate to the DBMS by sending queries with different combinations of login/password pairs. The retrieved credentials can be exploited to acquire access to the DBMS or to an account with higher privileges such as an administrator account.

- **Query Flood**: An attacker may prevent legitimate users from accessing the DBMS by saturating the database traffic. In particular, an attacker can send a huge number of SQL queries to the DBMS to make it unavailable to serve more requests and, thus, causing denial of service.

- **Inference Attack**: Inference attack is a complex method employed by an attacker to implicitly acquire knowledge on data he does not have direct access. To this end, an attacker can send multiple queries requesting statistical information about the data, such as the average value of an attribute, and then aggregate the responses to infer sensitive information.

Note that our goal is not to provide an exhaustive list of attacks against DBMS but to demonstrate our approach for the classification of alerts with respect to database attacks. Additional attacks have been presented in the literature (e.g., excessive privileges, denial of service); however, in many cases they are vulnerabilities within the DBMS or attacker objectives rather than attacks.

	Syntax-centric Features							
Attacks/Features	**Command**	**Column List**	**Table List**	**Search Space**	**Length**	**Malicious Patterns**	**Operators/ Functions**	**Syntax Error**
Privilege Misuse	Query	Query	Query	✘	✘	✘	Where	✘
Masquerade Attack	✘	✘	✘	✘	✘	✘	✘	✘
SQL Injection	NQ	Where	Where	✘	Query Where	✔	Where	✘
Resource Consumption	✘	✘	✘	# Tables # Columns # NQ	✘	✘	Where	✘
Error-code Analysis	Query	✘	✘	✘	✘	✘	Query	✔
Password Guessing	✘	✘	✘	✘	✘	✘	✘	✘
Query Flood	✘	✘	✘	✘	✘	✘	✘	✘
Inference Attack	✘	✘	✘	✘	✘	✘	Query	✘

Legenda: ✔ Related Feature ✘ Unrelated Feature

Table 3: Attack Taxonomy with respect to Syntax-centric Features

	Context-centric Features				Result-centric Features			
Attacks/ Features	**IP Addr.**	**Timestamp**	**Issuer**	**# Queries**	**Result Set**	**Result Size**	**Response Code**	**Response Time**
Privilege Misuse	✘	✘	User Role	✘	✔	Records	✘	✘
Masquerade Attack	✔	✔	User	✘	✘	✘	✘	✘
SQL Injection	✘	✘	✘	✘	✘	✘	✘	✘
Resource Consumption	✘	✘	✘	✘	✘	Records Bytes	✔	✔
Error-code analysis	✘	✘	✘	✘	✘	✘	✔	✘
Password Guess	✔	✘	User	✔	✘	✘	✔	✘
Query Flood	✘	✘	✘	✔	✘	✘	✘	✘
Inference Attack	✘	✘	✘	✔	✔	✘	✘	✘

Legenda: ✔ Related Feature ✘ Unrelated Feature

Table 4: Attack Taxonomy with respect to Context and Result-centric Features

4.2. Feature - Attack Relation

A common characteristic of the attacks presented in the previous section is that attackers' activities differ from the behavior profiles characterizing the database activities of legitimate users. In particular, attacks can be characterized by combinations of anomalous features that can be extracted from the query, such as the SQL command or result set. The relation between features and attacks is presented in Table 3 and Table 4.

Privilege misuse indicates that a user accesses different (type of) data compared to the data he usually accesses or performs abnormal actions on data (e.g., insertion of new data in contrast to SELECT queries that are usually performed by a user). Such an anomalous behavior can be captured by analyzing syntax- and result-centric features. For instance, the command of the query determines the op-

eration performed on the data. Table list, column list and conditions used in the WHERE clause (operators/functions) determine the type of data requested by the user. In addition, the amount of records in the response (result size) and the result set itself provide additional information on the data that were accessed.

Masquerade attacks are characterized by the context in which database activities are performed. For instance, malicious queries often originate from an anomalous IP address or occur at an unusual time. Such characteristics can be captured by analyzing context-centric features, such as IP address and timestamp.

Queries used for SQL injection often have different syntax characteristics than legitimate queries. In particular, the injected code is usually constructed using different character encodings and combination of special characters, operators and functions. SQL injection is typically identified using signature-based approaches by inspecting SQL queries for SQL injection patterns [29, 30, 31]. We integrate these approaches by introducing the feature malicious patterns in our analysis. This feature is marked anomalous if an SQL injection pattern is detected in the query. However, signature-based approaches are not able to detect SQL injections that do not contain a known pattern. To this end, we consider other features to enable the classification of new or unknown instances of SQL injection attacks. For instance, due to the nature of SQL injection, the length of the query and most frequently of the WHERE clause are usually longer than the one of legitimate queries. Moreover, the attacker may try to perform unusual actions using nested queries or he might request anomalous information in the WHERE clause. To this end, the attacker might manipulate the functions in the WHERE clause. These characteristics can be captured using syntax-centric features such as command, column list, table list and functions/operators.

Resource consumption attacks are typically characterized by the complexity of queries. Several features can be used to analyze such a complexity. For instance, the number of tables and columns specified in a query defines the search space of the query. The number of nested queries and operators/functions characterizes the complexity of the computation required for the execution of a query. The complexity of a query can be also observed from the result size (in terms of number of *records* and *bytes*[2]) and response time. Moreover, many DBMS raise special response codes associated with the limited availability of resources. This feature provides additional indications about the complexity of a query.

Error-code analysis attacks are typically characterized by the erroneous form of the SQL query and the unusual associated response code raised by the DBMS.

[2]Note that the number of records and the number of bytes retrieved might differ significantly as a record may contain variable amount of information.

Such characteristics can be captured by the syntax error and response code features respectively. Moreover, the attacker might try to validate the support of a certain functionality offered by the DBMS that is not commonly used by users. For instance, the execution of the xp_cmdshell[3] function, which is used to run operating-system commands in SQL Server DBMS, may raise suspicion. Such cases can be detected by examining syntax-centric features, such as command and operations/functions.

Password guessing attacks are characterized by an unusual high number of authentication queries with a "failed login" response code. These characteristics can be captured by examining the number of queries and the associated response code. In addition, these attacks are usually carried out from an unusual location. Such characteristics can be captured using context-centric features like IP address.

Query flood attack is the simplest method employed by attackers to perform a denial of service attack against a DBMS. Intuitively, the main characteristic of this attack is the unusual large number of queries sent to the DBMS (per second).

Inference attacks are characterized by an attacker aiming to infer sensitive information by sending multiple queries to the DBMS. In inference attacks each individual query sent by the attacker can be normal (i.e., no anomalous features). However, relating and analyzing these queries together may raise suspicion. In particular, the use of specific syntax features over multiple queries might differ significantly from legitimate user activities. For instance, an attacker may use an unusual sequence of min() and max() functions or conditional operators over multiple queries, to obtain information that allows the inference of sensitive information, which he is not authorized to access. Such characteristics are captured by analyzing the operators/functions feature among multiple queries. In addition, an attacker may issue a number of queries, each of them requesting a small portion of data. Although the data retrieved by each query individually might not be anomalous, the result set obtained by combining the result set of every query forming the attack might provide evidence that an attack occurred.

In the next section we present an approach for attack classification. In particular, we focus on the classification of attacks that can be identified from a single query. Attacks whose detection requires observing multiple queries (i.e., password guessing, query flood, inference attacks) demand an approach for correlating queries [32, 33]. We leave the investigation of such an approach for future work.

[3] http://technet.microsoft.com/en-us/library/ms175046(v=sql.105).aspx

IF Malicious Pattern == Anomalous
THEN SQL Injection

IF Query Length == Anomalous **AND** WHERE Functions == Anomalous
THEN SQL Injection

Figure 4: Sample feature-based classification rules for SQL Injection

4.3. Attack Classification

Based on the relation between features and attacks presented in the previous section, we propose a feature-based approach for the attack classification of anomalous queries. Intuitively, each attack type is associated with a set of feature-based classification rules which specify the (combinations of) features characterizing that attack type. Feature-based classification rules are expressions of the form:

IF *Condition* **THEN** *Attack*

where *Condition* is a conjunction of features and *Attack* is a label identifying the attack. In particular, the condition enumerates the features that enable the detection of the attack (as shown in Table 3 and Table 4). A sample set of rules for the classification of SQL injection attacks is shown in Fig. 4. The first rule examines the presence of a known SQL injection pattern in a query, while the second rule checks whether both the query length and the functions used in the query are anomalous.

When an anomalous query is detected by the anomaly detection system, the query is classified by analyzing it against the feature-based attack classification rules. In particular, given a set of feature-based attack classification rules R and an anomalous query Q, the satisfiability of each rule $r \in R$ is examined on the basis of the anomalous features of Q. If the *Condition* of a rule is satisfied, then the query is labeled with the *Attack* associated with the rule.

It is worth noting that more than one rule can be satisfied by an anomalous query. To determine which classification rule is the most appropriate, we introduce the notion of *classification confidence*. Classification confidence represents the level of certainty that an anomalous query corresponds to a specific attack. Given an anomalous query Q and a feature-based attack classification rule $r \in R$, the confidence of Q with respect to r is:

$$ClassificationConfidence(Q,r) = 1 - \sum_{f_i \in AF(Q)|_r} \frac{\pi(v,h_i)}{|AF(Q)|_r|} \tag{8}$$

where $\pi(v,h_i)$ is the normalized probability of the value v of feature f_i with respect to the bin threshold (Eq. 5) and $AF(Q)|_r$ is the set of anomalous features of Q that

occur in the *Condition* of r. Note that in order for a rule to be applicable to a query, all the features listed in the condition of the rule should be anomalous in the query. Therefore, the features in $AF(Q)|_r$ are exactly the features in the condition of a rule r. Accordingly, the classification confidence (Eq. 8) is computed by averaging the normalized probability of the (anomalous) values of the features occurring in the condition of the rule are normal.

To determine the level of certainty that an anomalous query corresponds to a certain attack, we extend the notion of classification confidence to sets of rules. Given the set R_i of classification rules for an attack i and an anomalous query Q, the classification confidence with respect to the attack is:

$$\mathit{ClassificationConfidence}(Q,R_i) = \max_{r \in R_i} \{\mathit{ClassificationConfidence}(Q,r)\} \quad (9)$$

An anomalous query is labeled with the attack that has the highest classification confidence.

5. Alert Visualization and Analysis

Anomaly detection solutions often produce a large number of alerts which have to be handled and investigated by security officers. To properly handle alerts and mitigate the potential damages caused by security incidents, security officers should be able to timely focus on the most critical cases and have the information necessary for the investigation (e.g., root causes, attack type). However, existing anomaly detection solutions often do not provide such essential information to security officers. Thus, a security officer should analyze all alerts and gather the information necessary for the analysis, making the investigation of security incidents a time-consuming and costly task.

To facilitate the task of security officers, we have developed a web-based database anomaly audit tool[4] for the visualization of alerts raised by an anomaly detection system, enriched with additional information obtained using the approaches presented in Sections 3 and 4. A screenshot of the tool is shown in Fig. 5. For each alert, the associated profile (e.g., user, role) along with the requested tables and columns in performed query is displayed (first, second and third column respectively). The root causes of the anomaly (i.e., anomalous feature values) and the risk of corresponding to a data leakage are shown in a user-friendly way. In particular, the risk level is visualized using a three value scale, namely High Risk (shown in Red), Medium Risk (shown in Orange) and Low Risk (shown in Yellow). Note

[4] Available at `http://security1.win.tue.nl/visualization/`

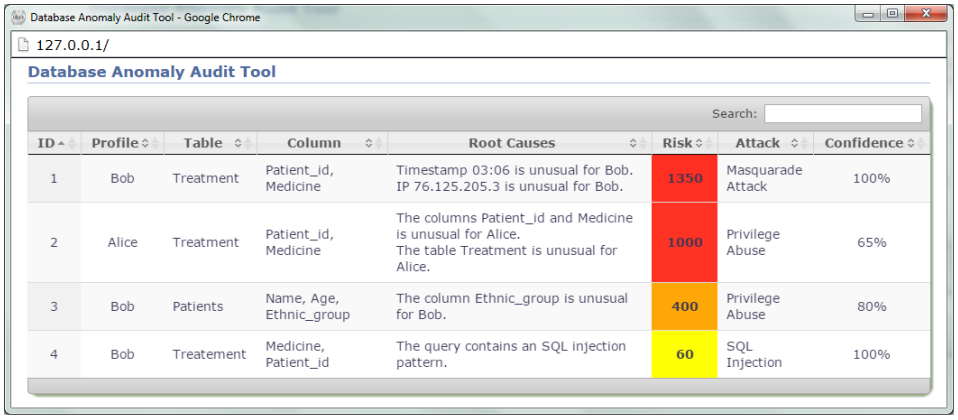

Figure 5: Web-based Database Anomaly Audit Tool

that the definition of such a scale is domain dependent; it is based on the sensitive values used to annotate data within the domain as well as the security requirements of the organization. The last two columns show the attack type of the alert along with the corresponding classification confidence.

The visualized information about alerts, such as root causes and attack type, allows security officers to have an insight on security incidents and thus support them in deciding which actions should be taken to mitigate the losses or prevent security incidents in the future. For instance, consider the first alert in Fig. 5, which has been classified as a masquerade attack. Based on this information, a security officer can notify Bob about this incident and ask him to change his login credentials. In addition, the tool provides capabilities that facilitate the analysis of alerts. For instance, it allows security officers to rank alerts on the basis of their data leakage risk and, thus, focus on the most critical data leakages. Alerts can also be grouped with respect to the attack type, which enables security officers to focus on a particular type of security incident.

We remark that the aim of the tool and analysis methods proposed in this paper is to assist security officers in handling alerts raised by the employed anomaly detection system and, in particular, to facilitate the investigation of security incidents by providing additional insights. However, they do not aim to relieve security officers from their duties. A thorough investigation of a security incident may reveal additional information about the incident. For example, consider the third alert in Fig. 5. In this case, a security officer should investigate Bob's activities based on the information provided by the tool. Such an investigation could reveal that Bob was on vacation and he could not have accessed the database when the incident occurred. Thus, the security officer could deem a masquerade attack took place

and thus investigate this further.

The analysis of a security incident can reveal that an alert is a false positive (i.e., wrongly recognized by the anomaly detection system as a security incident) or that a richer set of features is needed for a more accurate classification of alerts. In this case, feedback explaining the assessment can be used to enhance the anomaly detection system and attack classification rules. For instance, in the previous example a context-centric feature concerning user absence (e.g., vacation) could be considered to better characterize user activities.

6. Evaluation

In this section we evaluate the risk-based quantification approach (Section 3) and the feature-based attack classification approach (Section 4). First, we present the methodology and evaluation framework for the design and assessment of experiments, followed by a description of the scenario and training setup used for the experiments. Then, we report and discuss the experimental results for both approaches. Finally, we discuss the complexity and performance of the alert analysis framework.

6.1. Methodology and Evaluation Framework

The goal of the evaluation is to assess the quality of the annotations obtained using the risk-based quantification approach and the feature-based attack classification approach. We refer to annotation as the generic term describing the results of the two approaches (i.e., risk assessment and attack label). We also compare the risk-based quantification approach with the quantification approaches presented in Section 2.

For the evaluation, we have simulated a typical scenario in the healthcare domain (Section 6.2). The scenario was used to create a query dataset describing normal user activities. This dataset was used for the training of the anomaly detection system. Based on the same scenario, we generated two datasets of anomalous queries: one for the evaluation of the quantification approaches (Section 6.3) and one for the evaluation of the feature-based attack classification approach (Section 6.4). The two datasets have been manually annotated based on the purpose of the analysis and types of anomaly. We refer to such annotations as ground-truth annotations. The annotated datasets were used as a reference model for the analysis and comparison of the approaches.

To assess the quality of the results of experiments obtained using both the quantification and classification methods, we have considered an evaluation framework consisting of a number of standard quality metrics [34, 35]: recall, precision

and F_1-measure. Here, we briefly introduce these metrics, while we refer to Sections 6.3 and 6.4 for their definition and a discussion of the results.

- *Recall* measures the quality of an approach in finding relevant information with respect to the dataset. It is calculated dividing the number of relevant annotations detected using the approach by the number of all relevant ground-truth annotations from the dataset.

- *Precision* measures the quality of an approach in terms of finding relevant information with respect to the generated output of the approach. It is calculated by dividing the number of correctly detected annotations over the number of all detected annotations.

- F_1*-measure* is an aggregate measure of performance which combines precision and recall into a single measure. In particular, it is computed as the harmonic mean of precision and recall.

6.2. Scenario and Training Setup

To validate the proposed approaches, we have considered a typical scenario in healthcare. Patient information is stored in a central database at a hospital in the form of electronic health records. The hospital provides treatment for a variety of diseases, ranging from common flu to AIDS. Typically, a patient is assigned to a doctor who is responsible for his treatment (primary care doctor). On the other hand, different nurses may assist a patient. In emergency situations every doctor and nurse should be able to provide the correct treatment to a patient. Therefore, doctors and nurses can access every patient record in the database. However, they should only access the information of those patients assigned to them. The hospital has also administrative personnel to arrange financial issues and make appointment with patients. Moreover, the database is maintained by a database administrator.

We implemented the hospital database using GNU Health[5], a healthcare management system used by several healthcare providers worldwide. We assume that the hospital employs an anomaly detection solution as the one described in Section 2.2. To represent normal user activities, we constructed behavior profiles based on a synthetic dataset obtained using GNU Health. To this end, we simulated the normal behavior of an administrator and hospital staff, where doctors and nurses take care of patients suffering from different diseases. To understand and define the normal user behavior, we held discussions with domain experts from our industry partner, Roessingh Hospital. Based on these discussions, a set of tasks, which are

[5] `http://health.gnu.org`

typically performed within a hospital, was defined. For example, doctors access patients' medical data to prescribe medication and treatment; administrators perform database maintenance and manage users' accounts. We simulated these tasks in GNU Health and captured the queries sent by the healthcare management system to the database along with the corresponding database responses. (Note that GNU Health may generate several queries for each user action.) The generated dataset contains 30492 queries. The dataset was examined and validated by both domain experts and medical staff of Roessingh Hospital, ensuring that the queries reflect the defined tasks. The validated datasets was used in the learning phase of the anomaly detection solution.

6.3. Evaluation of Risk-based Data Leakage Quantification

In this section we evaluate the ability of the risk-based quantification approach to correctly assess the criticality of alerts (with respect to data leakage) and compare such an approach with the other quantification approaches presented in Section 2.

Setting. Similarly to the generation of the dataset used for the learning phase of the anomaly detection solution, we discussed with the medical staff of Roessingh Hospital realistic anomalous behaviour that can occur in a hospital. For instance, administration personnel access patients' disease information during the night. We simulated these anomalous activities in GNU Health. The corresponding queries were analyzed using the anomalous detection system.

Based on the alert raised by the anomaly detection system, we constructed a *leakage dataset* consisting of 194 queries. A number of security experts, coming from both academia and industry, were invited to assess the criticality of the alerts in the leakage dataset. In particular, we developed a questionnaire describing these leakages. To facilitate the understanding of the alerts, we provided security experts additional information about the alerts such as root causes, sensitivity of leaked records and anomaly level. The security experts were invited to answer the questionnaire and evaluate the criticality of each data leakage using a three-valued scale (i.e., Low, Medium, High) based on the provided information. For each data leakage in the dataset, we assigned a criticality level reflecting the level given by the majority of the experts. It is worth noting that the variance of the evaluation given by the experts was minimal. This assessment was the underlying reference dataset for the evaluation and comparison of data leakage quantification methods.

For the evaluation, we computed recall, precision and F_1-measure per criticality level (i.e., Low, Medium, High), as well as for the overall leakage dataset. Recall is measured as:

- $\mathit{Quantification\text{-}Recall}_{overall} = \frac{\text{\# of correct assignments}}{\text{\# of alerts in reference dataset}}$

	Recall				Precision			
	Low	**Medium**	**High**	**Overall**	**Low**	**Medium**	**High**	**Overall**
Anomaly Score	74.2%	27.4%	35.7%	51.0%	58.0%	45.6%	32.3%	51.0%
Severity	68.8%	74.6%	89.3%	73.7%	92.8%	66.7%	56.8%	73.7%
Risk	90.3%	78.3%	89.3%	86.1%	93.4%	83.0%	73.5%	86.1%

Table 5: Data Leakage Quantification: Recall and Precision

	F_1-Measure			
	Low	**Medium**	**High**	**Overall**
Anomaly Score	65.1%	34.2%	33.9%	51.0%
Severity	79.0%	70.1%	69.4%	73.7%
Risk	91.2%	81.1%	80.6%	86.1%

Table 6: Data Leakage Quantification: F_1-Measure

- $Quantification\text{-}Recall_{class} = \frac{\text{\# of correct assignments to a particular class}}{\text{\# of alerts in class as defined in reference dataset}}$

Precision is measured as:

- $Quantification\text{-}Precision_{overall} = \frac{\text{\# of correct assignments}}{\text{total \# of returned assignments}}$
- $Quantification\text{-}Precision_{class} = \frac{\text{\# of correct assignments to a particular class}}{\text{total \# of returned assignments to a particular class}}$

F_1-measure is measured as:

- $Quantification\text{-}F_1\text{-Measure}_{overall} = 2\frac{Quantification\text{-}Recall_{overall} \times Quantification\text{-}Precision_{overall}}{Quantification\text{-}Recall_{overall} + Quantification\text{-}Precision_{overall}}$
- $Quantification\text{-}F_1\text{-Measure}_{class} = 2\frac{Quantification\text{-}Recall_{class} \times Quantification\text{-}Precision_{class}}{Quantification\text{-}Recall_{class} + Quantification\text{-}Precision_{class}}$

Results. The results of the quantification approaches are shown in Table 5 and Table 6. For each approach, the tables report recall, precision and F_1-measure for each criticality class as well as for the overall leakage dataset. We note that in the overall case, the approaches return annotation for every alert in the dataset. Therefore, the precision, recall and F_1-measure metrics converge to the same value.

The results show that the anomaly score computed as proposed in [3] fails to assess the criticality of alerts properly (with respect to the security experts' assessment). In particular, this approach yields a low recall, and precision is close or below 50%. These significant differences between the anomaly score and experts' assessment are due to the fact that the anomaly score takes into account only the probability of features being anomalous and not the criticality of the actual leaked data. On the other hand, the severity-based approach has an overall recall and precision of 73.7%, which is a major improvement compared to the anomaly score. The most notable improvements are noted in the assessment of leakages in the Medium and High criticality classes, where recall is 74.6% and 89.3% respectively. These results show that the severity of data leakages has a significant impact

on their criticality. However, the low recall rate for the Low criticality class and the low precision for the other classes show that the severity-based approach assigns higher criticality to leakages compared to the experts' assessment. For instance, the severity-based approach assigns several leakages in the Low criticality class to the Medium criticality class. This is due to the fact that severity-based quantification does not consider the uncertainties associated with the security incidents. Thus, severity can be seen as an overestimation of the criticality of alerts.

Table 5 and Table 6 show that the risk-based quantification approach performs better than the other two approaches. In particular, a major improvement is noted in the quantification of leakages in the Low criticality class, for which recall is 90.3%. Moreover, a significant increase in precision is obtained for the Medium and High criticality classes. Intuitively, the risk-based approach corrects the overestimation of the severity-based approach by also considering the anomaly level. We can conclude that the risk-based approach better reflects the experts' evaluation of the criticality of data leakages than the other two quantification solutions, as it considers both the anomaly level and severity of incidents.

6.4. Evaluation of Feature-based Attack Classification

In this section we evaluate the ability of the feature-based attack classification approach to correctly classify alerts raised by an anomaly detection system with respect to database attacks.

Setting. For the experiments, we created an *attack dataset* consisting of 125 single query database attacks. In particular, we specified 25 different attack instances for each single query attack type presented in Section 4. These attack instances cover different variants for each attack. For instance, approximately 50% of the queries (13 instances) corresponding to an SQL injection contain a known SQL injection pattern; the other 50% (12 instances) do not. The latter SQL injection instances were constructed using other techniques such as tautologies, UNION based, piggy-baked queries and alternate encodings [29]. This makes it possible to study the ability of the attack classification approach to classify different variants of the same attack type. All the queries in the attack dataset were identified by the anomaly detection system as anomalous queries.

For the evaluation, we computed recall, precision and F_1-measure per attack type as well as for the overall attack dataset. Recall is measured as:

- $Classification\text{-}Recall_{overall} = \frac{\text{\# of correct classifications}}{\text{\# of attacks}}$
- $Classification\text{-}Recall_{attack} = \frac{\text{\# of correct classifications for a particular attack}}{\text{\# of alerts in attack as defined in attack dataset}}$

Precision is measured as:

	Privilege Misuse	Masquerade Attack	SQL Injection	Resource Consumption	Error-code Analysis	Overall
Recall	96.0%	92.0%	80.0%	92.0%	88.0%	89.6%
Precision	89.0%	95.8%	95.2%	92.0%	95.7%	93.3%
F_1-Measure	92.3%	93.8%	86.9%	92.0%	91.1%	91.4%
Unclassified	0.0%	0.0%	8.0%	4.0%	8.0%	4.0%

Table 7: Feature-based attack classification validation results

- $Classification\text{-}Precision_{overall} = \frac{\text{\# of correct classifications}}{\text{\# of attacks}}$
- $Classification\text{-}Precision_{attack} = \frac{\text{\# of correct classifications for a particular attack}}{\text{total \# of returned classifications to a particular attack}}$

F_1-Measure is measured as:

- $Classification\text{-}F_1\text{-Measure}_{overall} = 2\frac{Classification\text{-}Recall_{overall} \times Classification\text{-}Precision_{overall}}{Classification\text{-}Recall_{overall} + Classification\text{-}Precision_{overall}}$
- $Classification\text{-}F_1\text{-Measure}_{attack} = 2\frac{Classification\text{-}Recall_{attack} \times Classification\text{-}Precision_{attack}}{Classification\text{-}Recall_{attack} + Classification\text{-}Precision_{attack}}$

Results. The results of the feature-based attack classification approach are presented in Table 7. The table reports recall, precision and F_1-measure for each attack type as well as for the overall attack dataset. The last row shows the percentage of queries in the attack dataset which were not annotated by the feature-based attack classification approach.

The results show that the proposed approach classifies correctly 89.6% of the attacks with a high precision (i.e., 93.3%). Moreover, 4.0% of the attacks were not assigned to any particular attack class, as the anomalous feature values of these attack queries did not satisfy any classification rule. The highest recall is achieved for privilege misuse attacks, where 96% of the attack instances were correctly identified. This, however, comes with the lowest precision (89%) due to the fact that other attack queries have anomalous features satisfying classification rules associated to privilege misuse. For the classification of masquerade attacks we have a recall of 92% and a precision of 95.8%. In particular, two instances of masquerade attack were not classified in this class due to the absence of some anomalous context-centric features, such as IP address. The lowest recall is obtained for the classification of SQL injection attacks, where 80% of the attack instances were correctly assigned to the class. In particular, all the instances that contain a known malicious pattern were identified, while approximately 58% of the remaining instances were classified exclusively using other features. In addition, the precision for SQL injection classification is 95.8% indicating the high quality of the classification. In particular, only one query was falsely identified as SQL injection. Our approach was able to classify 92% of the resource consumption attack. In particular, two attack queries were not classified properly as they originated from

the administrator who usually performs resource-consuming maintenance queries. Moreover, 8% (i.e., 92% precision) of the other attacks were falsely classified as resource consumption. For instance, one query corresponding to privilege misuse was wrongly classified as resource consumption due to the fact that it consists of an anomalous number of requested columns and tables. In the error-code analysis attack classification, recall and precision were 88% and 95.7% respectively. Similarly to the resource consumption attack, the queries which have not been correctly classified have similarities with (legitimate) troubleshooting queries sent by the database administrator. Based on these results we conclude that the feature-based attack classification method is able to classify attacks with a high recall and precision. However, the classification performance can be increased by introducing a richer set of features (especially context-centric). This will allow a more fine-grained characterization of attacks and thus the definition of more robust classification rules.

6.5. Discussion on the complexity of the alert analysis framework

In order to be applicable in practice, an alert analysis framework, like the one presented in this work, should not affect the performance and effectiveness of the DBMS to which it is connected. The alert analysis process underlying our framework subsumes three main phases, namely training, detection and risk/attack annotation. In the remainder of this section, we discuss the complexity and performance of these phases.

The goal of the training phase is twofold: (i) constructing behavior profiles for anomaly detection and (ii) annotating the data model with sensitivity labels. The complexity of (i) depends on the number of features employed to represent database activities and especially on the number of database activities recorded in the training dataset. On the other hand, the complexity of (ii) depends on the size of the data model (i.e., the number of nodes and relations in the data model). Thus, the training phase can be time-consuming when considering a very large training dataset and/or data model. However, we stress that this phase is performed off-line and occurs only once, before any alert is detected and annotated. Therefore, the training phase does not affect the effectiveness and performance of the DBMS.

Our alert analysis framework relies on the alerts generated by an (external) anomaly detection system. Although the detection of anomalies is not in the scope of this work, as our focus is on the analysis and annotation of alerts, the detection phase can have a significant impact on the performance of the DBMS. In our work, we employed the white-box behavioral-based solution presented in [3] for anomaly detection. This solution has been evaluated using a research prototype, which was not optimized for performance. Experiments on synthetic and real-live datasets

show that checking whether a database query is anomalous requires, on average, 0.040 ms.

In this work, we have proposed two different types of analysis of alerts: an approach for assessing the criticality of alerts regarding potential data leakages and a method for determining the attack that caused the alert. To assess the criticality of data leakages, our approach computes the anomaly level and severity of alerts. The anomaly level is directly obtained from the white-box behavioral-based solution used for the detection of anomalies. Therefore, its calculation does not require any additional overhead. The severity of the alerts, on the other hand, depends on the amount of leaked data and size of the data model. Recall that severity is calculated on the basis of the sensitivity of each sensitive attribute value leaked. Therefore, for each sensitive attribute value we have to traverse the data model and retrieve the sensitivity label associated to the node corresponding to the attribute value. In our prototype we have used the depth-first search algorithm to traverse the data model, which has a complexity linear in the size of the data model (i.e., $O(|H|)$ where H denotes the set of hierarchical relations in the data model). Although this may not be efficient to deal with large data models, the performance of search algorithms can be significantly improved by employing caching and indexing techniques (an optimization of the prototype is left for future work). Finally, the complexity of the attack classification is linear in the number of feature-based attack classification rules.

To conclude, we remark that the aim of this work is to assist security officers in the analysis and handling of the raised alerts rather than in the run-time prevention of security incidents, which would impose additional constraints on performance. In particular, our framework is based on a posteriori analysis of logs recording users' database activities. Therefore, it does not directly interfere with the execution of SQL queries in the DBMS. Nonetheless, based on the observation above, we believe that the introduced overhead does not significantly affect the effectiveness of the DBMS. Therefore, our framework can be suitable for the run-time analysis of alerts for those systems in which performance and response time are not the most critical issues.

7. Related Work

Several proposals aiming at the detection of database security incidents can be found in the literature, and in particular in the field of anomaly detection [3, 4, 5, 6, 7, 10, 36, 37]. Various techniques have been proposed to detect anomalies [8]. In particular, anomaly detection solutions can be classified into signature-based and behavior-based approaches. Signature-based approaches [9, 10] use blacklists defining sets of patterns representing well-known threats or undesired behaviors,

which have to be manually specified based on prior knowledge. The major drawback of signature-based approaches is that they can only detect known misuse patterns. In contrast, behavior-based solutions [3, 11] use a whitelisting approach in which profiles of normal user behavior are defined by observing users' activities, and raise an alert when a query deviates from such profiles, thus allowing the detection of unknown attacks. These approaches usually employ machine learning techniques, such as statistical analysis or classifiers, to dynamically learn the behavior profiles needed to detect anomalous activities. However, the ability of detecting unknown attacks comes at a cost: behavior-based anomaly detection systems are usually affected by a large number of false alerts, i.e. alerts that do not correspond to an actual security incident. These systems, thus, can benefit from our anomaly analysis framework. Indeed, our framework provides analysis capabilities which allow the discrimination of alerts based on their criticality.

The problem of identifying the most critical security incidents has been investigated in the literature. Some approaches [3, 20] quantify alerts with respect to their anomaly level, allowing security officers to focus on the most anomalous cases. However, these approaches do not consider the criticality of the data leaked as a result of a security incident. This issue is mostly studied in the field of quantitative information flow [38, 39, 40]. These solutions measure the amount of information leaking from a high confidentiality input to a low confidentiality output. Leakages are usually quantified in terms of bits using metrics based on information theory and information entropy. The major drawback of quantitative information flow methods is that they do not consider the semantics of the leaked information to quantify data leakages. In particular, the sensitivity of leaked data is not considered in the calculation of the severity of a leakage. Only two proposals [21, 22] use semantic information to compute the severity of data leakages. These proposals measure the severity of leakages in database environment on the basis of the amount and sensitivity of the leaked data. However, they do not consider the uncertainties associated with the security incidents. Therefore, they are not able to discriminate false alerts. In this work, we proposed a risk-based alert quantification approach that leverages both anomaly level and severity of alerts. Experimental results show that our approach provides a better characterization of the criticality of alerts compared to existing alert quantification solutions.

A number of solutions for the quantification of security incidents with respect to their risk level have been proposed in the literature [41, 42, 43]. In particular, these risk management frameworks quantify the impact of security incidents in financial terms (e.g., damage to the reputation of the organization, losses of revenue). However, these solutions are generic and not tailored to the analysis of data leakages within database environments. As a consequence, they do not propose a systematic and comprehensive approach to assess the severity and probability of alerts

and thus to evaluate the risk level of data breaches within database environments. For instance, Farahmand et al. [42] determine the uncertainties related to security incidents through a survey-based subjective probability assessment. In contrast, our work proposes a systematic method for the evaluation of the risk level of data breaches by monitoring database activities. In particular, to calculate the risk level of data leakages we extend and integrate a behavioral-based anomaly detection solution [3] for assessing the anomaly level of alerts and a data leakage quantification solution [22] for assessing their severity. Compared to the other solution for assessing their severity [21], the solution in [22] provides a more accurate discrimination of alerts based on their severity. On the other hand, the behavioral-based anomaly detection system presented in [3] offers a number of advantages compared to other anomaly detection systems. First, while still being able to detect unknown attacks because of its behavioral-based design, it generates a lower false positive ratio compared to other behavioral-based approaches. Moreover, to our knowledge, it is the only white-box behavioral-based approach tailored to DBMS. In contrast to existing black-box solutions (e.g., [25, 26]), which only flag a query as anomalous or not, the white-box approach enables the extrapolation of the root causes of alerts in terms of anomalous features. Besides being useful for understanding the alerts, the anomalous features are used for attack classification (Section 4). Therefore, we believe that these two solutions provide a solid basis for our work.

The relation between the alerts generated by existing anomaly detection solutions and database attacks is not sufficiently investigated in the literature. On the one hand, signature-based database detection approaches can easily determine the type of attack related to an alert. In particular, the rules used by these approaches encode patterns representing known attacks, allowing the classification of alerts with respect to those attacks. However, novel attacks or variation of known attacks might remain undetected or be incorrectly classified. On the other hand, attack classification in behavior-based anomaly detection systems is still an open problem. Most behavior-based solutions [25, 26] are not able to provide the reason of an alert because of their black-box nature. Few proposals [3, 5] provide an intuition on how features can be related to certain attacks. For instance, Costante et al. [3] use syntax and result-centric features to identify privilege abuse attacks. However, these proposals do not provide a systematic approach to classify attacks. More advanced approaches exist in the field of anomaly-based intrusion detection systems. For instance, Panacea [17] analyzes attack payloads using n-gram analysis and classifies attacks using SVM or RIPPER classifiers. However, anomaly-based intrusion detection systems are usually tailored for the analysis of network traffic. From our knowledge, this is the first work that provides a systematic approach to the problem of attack classification for behavior-based anomaly detection systems in a database environment. In particular, we exploited the white-box behavioral-

based solution presented in [3] to extract the anomalous features in a query and used such features for the attack classification of the query.

8. Conclusions and Future Work

In this work we presented a framework that facilitates the analysis of alerts raised by an anomaly detection system within database environments. The framework encompasses two approaches to enrich alerts with information tailored for their analysis. In particular, we proposed an approach to quantify alerts on the basis of the risk of the disclosure of sensitive information. Moreover, we proposed a novel method for feature-based classification of alerts with respect to database attacks. The information gathered using these two proposals is visualized in a web-based database anomaly audit tool. The tool provides ranking and classification capabilities, easing the investigation of alerts. We validated our proposals by analyzing the alerts raised by an anomaly detection system using a case study in the healthcare domain. Our experiments show that the proposed risk-based quantification approach offers a significant improvement over existing alert quantification approaches and is able to better characterize the criticality of alerts. Moreover, our alert classification approach shows promising results for several types of database attacks.

The methods proposed in this work are based on alerts raised by the analysis of single database queries. Anomalous database activities, however, can span over multiple queries that individually may appear to be legitimate. The analysis and correlation of multiple queries will allow the detection of those anomalous activities. Based on such an analysis, an interesting direction for future work is the extension of the feature-based classification approach to cover attacks whose detection requires observing multiple queries (e.g., password guessing, query flood, inference attacks).

The visualization of alerts is a fundamental aspect for the understanding and timely management of security incidents. Our web-based anomaly audit tool makes a first step towards a user-friendly interface for the visualization of alerts. However, additional visualization capabilities could be provided to support the analysis of alerts. An interesting direction is the grouping of alerts based on other criteria (e.g., similar features), which can help identify correlations between alerts.

Acknowledgments. This work is funded by the Dutch program COMMIT under the THeCS project.

References

[1] Verizon, 2014 Data Breach Investigations Report, 2014.

[2] Information Age, New EU data laws to include 24hr breach notification, http://www.information-age.com/technology/security/1686838/new-eu-data-laws-to-include-24hr-breach-notification, 2012.

[3] E. Costante, J. den Hartog, M. Petković, S. Etalle, M. Pechenizkiy, Hunting the Unknown: White-Box Database Leakage Detection, in: Data and Applications Security and Privacy XXVIII, LNCS 8566, Springer, 2014, pp. 243–259.

[4] A. Kamra, E. Terzi, E. Bertino, Detecting anomalous access patterns in relational databases, The VLDB Journal 17 (2008) 1063–1077.

[5] S. Mathew, M. Petropoulos, H. Ngo, S. Upadhyaya, A data-centric approach to insider attack detection in database systems, in: Recent Advances in Intrusion Detection, LNCS 6307, Springer, 2010, pp. 382–401.

[6] M. Mohammadian, D. Hatzinakos, An Adaptive Intelligent Fuzzy Logic Classifier for Data Security and Privacy in Large Databases, in: SmartData: Privacy Meets Evolutionary Robotics, Springer, 2013, pp. 161–172.

[7] B. Shebaro, A. Sallam, A. Kamra, E. Bertino, PostgreSQL Anomalous Query Detector, in: Proceedings of 16th International Conference on Extending Database Technology, ACM, 2013, pp. 741–744.

[8] V. Chandola, A. Banerjee, V. Kumar, Anomaly Detection: A Survey, ACM Comput. Surv. 41 (2009) 15:1–15:58.

[9] M. Hart, P. Manadhata, R. Johnson, Text classification for data loss prevention, in: Privacy Enhancing Technologies, LNCS 6794, Springer, 2011, pp. 18–37.

[10] T. Takebayashi, H. Tsuda, T. Hasebe, R. Masuoka, Data loss prevention technologies, Fujitsu Scientific and Technical Journal 46 (2010) 47–55.

[11] R. Koch, Towards next-generation intrusion detection, in: Proceedings of International Conference on Cyber Conflict, IEEE, 2011, pp. 1–18.

[12] I. A. Tndel, M. B. Line, M. G. Jaatun, Information security incident management: Current practice as reported in the literature, Computers & Security 45 (2014) 42–57.

[13] S. Banescu, N. Zannone, Measuring privacy compliance with process specifications, in: Proceedings of International Workshop on Security Measurements and Metrics, IEEE, 2011, pp. 41–50.

[14] D. Hadiosmanovi, L. Simionato, D. Bolzoni, E. Zambon, S. Etalle, N-gram against the machine: On the feasibility of the n-gram network analysis for binary protocols, in: Research in Attacks, Intrusions, and Defenses, LNCS 7462, Springer, 2012, pp. 354–373.

[15] R. J. Santos, J. Bernardino, M. Vieira, Approaches and challenges in database intrusion detection, SIGMOD Rec. 43 (2014) 36–47.

[16] E. Costante, S. Vavilis, S. Etalle, J. den Hartog, M. Petkovic, N. Zannone, Database anomalous activities - detection and quantification, in: Proceedings of the 10th International Conference on Security and Cryptography, SciTePress, 2013, pp. 603–608.

[17] D. Bolzoni, S. Etalle, P. H. Hartel, Panacea: Automating attack classification for anomaly-based network intrusion detection systems, in: Recent Advances in Intrusion Detection, LNCS 5758, Springer, 2009, pp. 1–20.

[18] K. Goseva-Popstojanova, G. Anastasovski, A. Dimitrijevikj, R. Pantev, B. Miller, Characterization and classification of malicious web traffic, Computers & Security 42 (2014) 92–115.

[19] G. Gowrison, K. Ramar, K. Muneeswaran, T. Revathi, Minimal complexity attack classification intrusion detection system, Applied Soft Computing 13 (2013) 921–927.

[20] K. Viswanathan, C. Lakshminarayan, V. Talwar, C. Wang, G. Macdonald, W. Satterfield, Ranking anomalies in data centers, in: Proceedings of IEEE Network Operations and Management Symposium, IEEE, 2012, pp. 79–87.

[21] A. Harel, A. Shabtai, L. Rokach, Y. Elovici, M-score: A misuseability weight measure, IEEE Transactions on Dependable and Secure Computing 9 (2012) 414–428.

[22] S. Vavilis, M. Petković, N. Zannone, Data Leakage Quantification, in: Data and Applications Security and Privacy XXVIII, LNCS 8566, Springer, 2014, pp. 98–113.

[23] ISO/IEC, Risk management-vocabulary-guidelines for use in standards, ISO/IEC Guide 73, 2009.

[24] Imperva, Top Ten Database Security Threats: The Most Significant Risks and How to Mitigate Them, White Paper, 2014.

[25] M. Gafny, A. Shabtai, L. Rokach, Y. Elovici, Applying unsupervised context-based analysis for detecting unauthorized data disclosure, in: Proceedings of the 18th Conference on Computer and Communications Security, ACM, 2011, pp. 765–768.

[26] G. Z. Wu, S. L. Osborn, X. Jin, Database intrusion detection using role profiling with role hierarchy, in: Proceedings of the 6th VLDB Workshop on Secure Data Management, LNCS 5776, Springer, 2009, pp. 33–48.

[27] Y. Asnar, R. Moretti, M. Sebastianis, N. Zannone, Risk as dependability metrics for the evaluation of business solutions: A model-driven approach, in: Proceedings of 3rd International Conference on Availability, Reliability and Security, IEEE, 2008, pp. 1240–1247.

[28] G. Elahi, E. Yu, N. Zannone, A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities, Requirements Engineering 15 (2010) 41–62.

[29] W. Halfond, J. Viegas, A. Orso, A classification of SQL-injection attacks and countermeasures, in: Proceedings of IEEE International Symposium on Secure Software Engineering, pp. 13–15.

[30] G. Buehrer, B. W. Weide, P. A. G. Sivilotti, Using Parse Tree Validation to Prevent SQL Injection Attacks, in: Proceedings of 5th International Workshop on Software Engineering and Middleware, ACM, 2005, pp. 106–113.

[31] K. Kemalis, T. Tzouramanis, SQL-IDS: A Specification-based Approach for SQL-injection Detection, in: Proceedings of ACM Symposium on Applied Computing, ACM, 2008, pp. 2153–2158.

[32] E. Bertino, G. Ghinita, Towards mechanisms for detection and prevention of data exfiltration by insiders, in: Proceedings of 6th ACM Symposium on Information, Computer and Communications Security, ACM, 2011, pp. 10–19.

[33] R. Sadoddin, A. A. Ghorbani, An incremental frequent structure mining framework for real-time alert correlation, Computers & Security 28 (2009) 153–173.

[34] Y. Benjamini, Y. Hochberg, Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing, Journal of the Royal Statistical Society. Series B (Methodological) 57 (1995) 289–300.

[35] Y. Yang, An evaluation of statistical approaches to text categorization, Inf. Retr. 1 (1999) 69–90.

[36] I. M. Abbadi, M. Alawneh, Preventing insider information leakage for enterprises, in: Proceedings of 2nd International Conference on Emerging Security Information, Systems and Technologies, IEEE, 2008, pp. 99–106.

[37] M. B. Salem, S. Hershkop, S. J. Stolfo, A survey of insider attack detection research, in: Insider Attack and Cyber Security, Advances in Information Security 39, Springer, 2008, pp. 69–90.

[38] M. Backes, B. Kopf, A. Rybalchenko, Automatic discovery and quantification of information leaks, in: Proceedings of 30th IEEE Symposium on Security and Privacy, IEEE, 2009, pp. 141–153.

[39] K. Borders, A. Prakash, Quantifying information leaks in outbound web traffic, in: Proceedings of 30th IEEE Symposium on Security and Privacy, IEEE, 2009, pp. 129–140.

[40] G. Smith, On the foundations of quantitative information flow, in: Foundations of Software Science and Computational Structures, LNCS 5504, Springer, 2009, pp. 288–302.

[41] B. Blakley, E. McDermott, D. Geer, Information security is information risk management, in: Proceedings of 4th Workshop on New Security Paradigms, ACM, 2001, pp. 97–104.

[42] F. Farahmand, S. B. Navathe, P. H. Enslow, G. P. Sharp, Managing vulnerabilities of information systems to security incidents, in: Proceedings of 5th International Conference on Electronic Commerce, ACM, 2003, pp. 348–354.

[43] A. Garg, J. Curtis, H. Halper, Quantifying the financial impact of it security breaches, Information Management & Computer Security 11 (2003) 74–83.