Journal of Computer Security 0 (0) 1
IOS Press

A Severity-based Quantification of Data Leakages in Database Systems [1]

Sokratis Vavilis [a,*] Milan Petković [a,b] Nicola Zannone [a]
[a] *Eindhoven University of Technology, Den Dolech 2, Eindhoven 5612AZ, Netherlands*
E-mail: {s.vavilis, m.petkovic, n.zannone}@tue.nl
[b] *Philips Research Eindhoven, High Tech Campus 34, Eindhoven 5656AE, Netherlands*
E-mail: milan.petkovic@philips.com

Abstract. The detection and handling of data leakages is becoming a critical issue for organizations. To this end, data leakage solutions are usually employed by organizations to monitor network traffic and the use of portable storage devices. However, these solutions often produce a large number of alerts, whose analysis is time-consuming and costly for organizations. To effectively handle leakage incidents, organizations should be able to focus on the most severe incidents. Therefore, alerts need to be analyzed and prioritized with respect to their severity. This work presents a novel approach for the quantification of data leakages based on their severity. The approach quantifies the severity of leakages with respect to the amount and sensitivity of the leaked information as well as the ability to re-identify the data subjects of the leaked information. To specify and reason on data sensitivity in an application domain, we propose a data model representing the knowledge within the domain. We validate our quantification approach by analyzing data leakages within a healthcare environment. Moreover, we demonstrate that the data model allows for a more accurate characterization of data sensitivity while reducing the efforts for its specification.

Keywords: Data Leakage Detection, Severity Metrics, Data Sensitivity Model, Alert Visualization

1. Introduction

In the recent years data breaches are becoming an increasingly major issue for organizations. This is demonstrated, on the one hand, by the increasing number of data breaches reported by public and private organizations. For instance, a study from Ponemon Institute in 2012 showed that 94% of US hospitals suffered serious data breaches [37]. On the other hand, data breaches have a serious impact on organizations. According to a 2014 report by Ponemon Institute and IBM [38], data breaches cost organizations $3.5 million dollars on average.

The main cause of data breaches is that IT systems often implement inadequate measures that allow users to access sensitive data, which they are not authorized to access. The problem is that it may not be always possible to specify fine-grained access control policies to protect sensitive data. For example, access control policies in hospitals often do not pose restrictions on the amount of health records that

[1]This work is an extended and revised version of [48]
[*]Corresponding author. E-mail: s.vavilis@tue.nl.

doctors can access. Moreover, access to information should not be restricted under certain circumstances. For instance, doctors should be able to access patient records to face an emergency. Typically, this is addressed using the break-the-glass protocol [2,6], which allows users to bypass security mechanisms, thus leading to potential data misuse.

Timely detection and management of data leakages is becoming a serious challenge for organizations [12]. According to the newly proposed EU data protection regulation, organizations are obliged to notify privacy authorities within 24 hours after the detection of a data breach [25]. To detect data leakages, organizations usually deploy data leakage detection (DLD) solutions. These solutions analyze the disclosed data and raise an alert when a leakage is detected. However, the number of alerts can be huge in certain situations, making their analysis and management difficult. For example, in hospitals a DLD solution might produce a large number of alerts due to the usage of the break-the-glass protocol. Before taking any action (e.g., notifying authorities), organizations typically evaluate a sample of the alerts manually. To effectively manage and mitigate the damage due to security incidents, organizations should be able to focus on the most severe incidents [47]. To this end, data leakages should be quantified based on their severity.

Data leakage quantification, however, is still an open problem. Many proposals [5,9,40] are founded on quantitative information flow. In particular, they quantify data leakages in terms of the number of "sensitive" bits which have been disclosed. Thereby, they do not consider the semantics of the leaked information in the assessment of data leakages. From our knowledge, only M-Score [20] assesses the severity of data leakages on the basis of the semantics of the leaked information. In particular, M-Score uses the amount and sensitivity of leaked information as well as an identifiability factor to measure the severity of leakages. The amount and sensitivity of leaked information characterize the "quantity" and "quality" aspects of the leakage respectively. These aspects are weighted with respect to an identifiability factor, which represents the ability to obtain the identity of the individual(s) to whom the leaked data refer. However, M-Score requires defining the sensitivity for all pieces of information explicitly. Such a task is time-consuming and error-prone. More importantly, M-Score only computes a coarse estimation of the sensitivity of leaked data, which may not reflect their actual sensitivity, and thus fails to accurately determine the severity of data leakages. A detailed study on the limitations of M-Score approach is presented in Section 6.

In this work we propose a novel approach to quantify data leakages, which relieves security experts from the burden of defining sensitivity annotations for piece of data and accounts for the actual sensitivity of leaked information in the calculation of the severity of data leakages. In particular, we make the following contributions:

- We propose a new metric that evaluates the severity of data leakages based on the amount and sensitivity of the leaked data as well as an identifiability factor;
- We propose a data model representing the knowledge of an application domain to specify and reason on the sensitivity of the information in the domain.

Our metric uses the same factors used by M-Score. However, compared to M-Score, our metric provides a more accurate discrimination of data leakages with respect to their severity. In addition, differently from M-Score, our approach does not require specifying the sensitivity for every piece of information characterizing the application domain explicitly. The data model makes it possible to infer the sensitivity of every piece of information through a sensitivity propagation mechanism based on an initial partial sensitivity assignment.

We validate our approach by analyzing a sample scenario in the healthcare domain. Healthcare is indeed an interesting domain to investigate as a large amount of sensitive data, such as patient healthcare records, has to be protected. In particular, we evaluate the ability of our quantification approach to correctly assess the criticality of data leakages. To this end, we have built a data leakage dataset and asked a group of security experts to annotate every leakage in the dataset with its criticality. The severity measurements calculated using our metric have been analyzed against the evaluation provided by the security experts. Moreover, we study the impact of the data model in determining data sensitivity. In particular, we evaluate the effectiveness of the data model in relation with the effort required to compute data sensitivity.

The remainder of the paper is organized as follows. Next section motivates the need of approaches for data leakage quantification using a running example in the healthcare domain. Section 3 presents an overview of our approach. Section 4 defines the data model along with the machinery to reason on data sensitivity, and Section 5 describes how leaked information is mapped to the data model. Section 6 presents our metric for data leakage quantification along with a comparison with M-Score. A discussion on the application of our approach and its potential benefits is presented in Section 7. An evaluation of the proposed metric and of the effectiveness of the data model is presented in Section 8. Finally, Section 9 discusses related work, and Section 10 concludes the paper by providing directions for future work.

2. Running Example

Consider a local hospital where patients of a small region are treated. The hospital offers treatment for various diseases, ranging from flu to serious cases such as heart attack and infectious diseases. Patient information is stored in a central database at the hospital in the form of electronic health records (EHR). Doctors and nurses can only access EHR of the patients they treat. 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. However, in emergency situations doctors and nurses can bypass access control mechanisms by invoking the break-the-glass protocol. Therefore, they can have access to the EHRs of all patients. The hospital has also administrative personnel for financial management and to make appointments with patients. Moreover, the database is maintained by a database administrator.

To detect data leakages, the hospital employs a DLD solution. In a typical day hospital employees access thousands of patient records. In addition, the number of invocations of the break-the-glass protocol can be huge [6]. Therefore, the DLD system can generate hundreds of alerts, making the evaluation of their severity difficult and time-consuming. Below we present three representative alerts of data leakages:

Alert 1 A query is made by a doctor requesting an unusual large number of patient records. In particular, the names and addresses of 10000 patients were retrieved.

Alert 2 A query for patient data is made by a doctor after his regular working hours. He retrieved 200 records containing the names and diseases of patients.

Alert 3 A query for data about patients affected by HIV is made by a medical researcher of the hospital, specialized on cardiovascular diseases. He retrieved 500 records containing the gender, age and treatment provided to patients.

To assist organizations in the evaluation of data leakages, leakages should be ranked on the basis of their severity. However, the quantification of data leakages is not a trivial task as the leakages may differ on several aspects. The *amount* of leaked information is a main aspect to quantify the severity of data leakages. For instance, the leakage described in Alert 1 contains thousands of patient records, while in Alert 2 only a relatively small amount (200) of records is retrieved. Another difference is the information leaked itself. In particular, the *sensitivity* of the information (i.e., the impact that its disclosure has on the patient) can be different. For instance, disease information (Alert 2) is more sensitive than patient addresses (Alert 1). Finally, data leakages also differ on the extent that an individual related to the data is identifiable. According to the EU Data Protection Directive (95/46/EC), personal data should be protected. However, the principles defined in the directive do not apply to anonymous data. Therefore, the ability to identify the individuals related to the leaked data has an impact on the severity of a leakage. For instance, in Alerts 1 and 2 the leaked data can be directly linked to patients' identity, while Alert 3 concerns data which do not allow the complete re-identification of patients.[1] Therefore, the first two alerts should be considered more severe than the third alert.

In order to obtain a ranking of alerts we need a method to quantify the severity of data leakages. Such quantification should take into account the amount and sensitivity of leaked information, and the extent to which the identity of the individuals related to the leaked information can be ascertained.

3. Approach

DLD solutions are often deployed by organization to detect data leakages. These solutions analyze the data leaving the system and raise an alert when a data leakage is detected. However, the number of alerts can be very large, making their analysis costly and time-consuming for organizations. To enable organizations to focus on the most severe incidents, data leakages have to be ranked based on their severity.

To address this issue, we propose a novel data leakage quantification system (Fig. 1). The system is connected to a DLD solution. When the DLD solution raises an alert indicating a potential data leakage, the quantification system analyzes the disclosed data to estimate the severity of the leakage. Since leakages can originate from different sources, data can be structured or unstructured. For instance, data originating from a database are structured, while data in an e-mail are usually unstructured. In this work, we focus on structured data where portions of the database's tables are leaving the database as result of a user query. However, the system can be extended using technologies like natural language processing and information retrieval, to extract information from unstructured data.

The severity of data leakages depends both on the amount of the data leaked and on the data themselves. Therefore, the quantification of data leakages should consider both these factors. In particular, data leakage quantification should reflect the cost of data disclosure according to the data subject/owner or to the organization hosting the data. We represent such a cost in terms of the sensitivity of data.

To enable the quantification of data leakages, it is thus necessary to assign a sensitivity value to every piece of data. Assigning a sensitivity value to all pieces of data, however, is time-consuming and error prone. To address this issue, we employ a data model representing the knowledge of the application domain to reason on data sensitivity. The data model makes it possible to specify the sensitivity of some pieces of information and infer the sensitivity for the other pieces of information based on this initial assignment (Section 4). To calculate the severity of data leakages, leaked data are mapped to the data model. Intuitively, the attributes and values in the leaked tables are mapped to the corresponding piece of information in the data model (Section 5). The sensitivity of data along with a discrimination factor,

[1]Here we assume that information about gender and age is not sufficient to fully re-identify a patient.

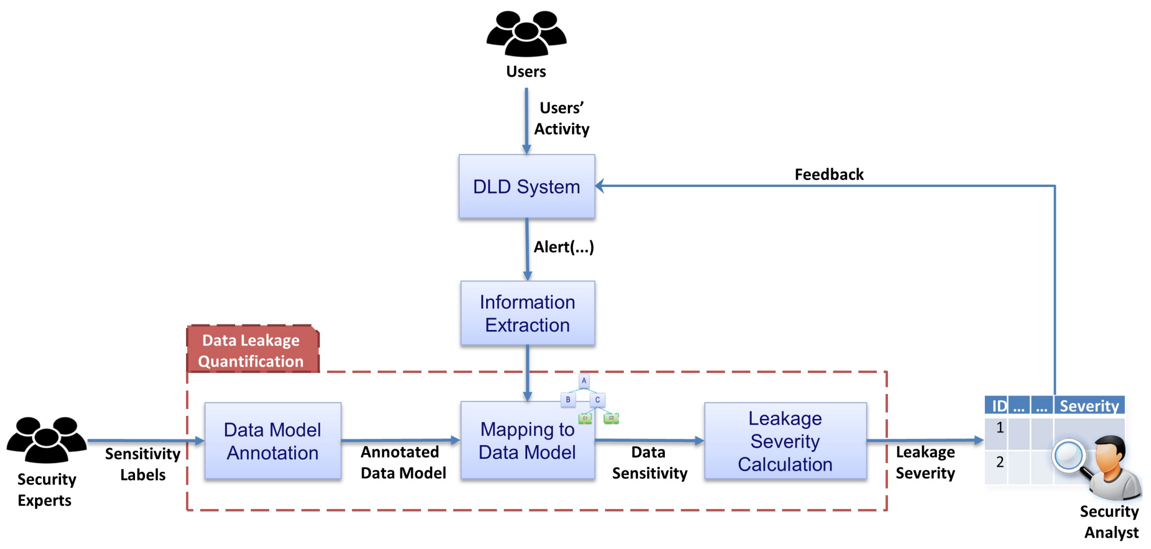


Fig. 1. Data Leakage Quantification Process

which determines to what extent data can be related to an individual, and the amount of leaked data is used to quantify the severity of data leakages (Section 6).

Data leakages are ranked and visualized on the basis of their severity. A security analyst, thus, can evaluate data leakages focusing on the more severe incidents. Based on this analysis, organizations can take the appropriate actions to prevent or mitigate the losses. If the analysis reveals that a leakage is a false positive (i.e., wrongly recognized by the DLD solution as a leakage), feedback explaining the assessment is sent to the DLD system to reduce the number of false alerts in the future.

4. Modeling and Reasoning on Data Sensitivity

To determine the sensitivity of data we represent the knowledge of the application domain using a data model. A data model provides a representation of the data within an application domain along with semantic relations between the data.

Definition 1 *A* data model *is a tuple $DM = (T, I, HR, IR, SL, PL)$, where:*

- *T is a set of data types.*
- *I is a set of data instances.*
- *$HR \subset T \times T \cup I$ is a hierarchy relation representing a specialization relationship.*
- *$IR \subset I \times I$ represents an inference relation on I.*
- *$SL : T \cup I \to \mathbb{R}_0^+$ is a labeling function that assigns a sensitivity value to data types and instances.*
- *$PL : I \times I \to [0,1]$ is a labeling function that defines the degree of inference, that is to what extent knowledge about a data instance can be inferred based on the knowledge about another data instance.*

Fig. 2 shows an example of data model for the healthcare domain. Data types are nodes represented by rectangles, while data instances by ovals. Hierarchy relations are represented with straight edges between two nodes. For instance, the hierarchy relation between Viral diseases and Flu nodes indicates that flu is

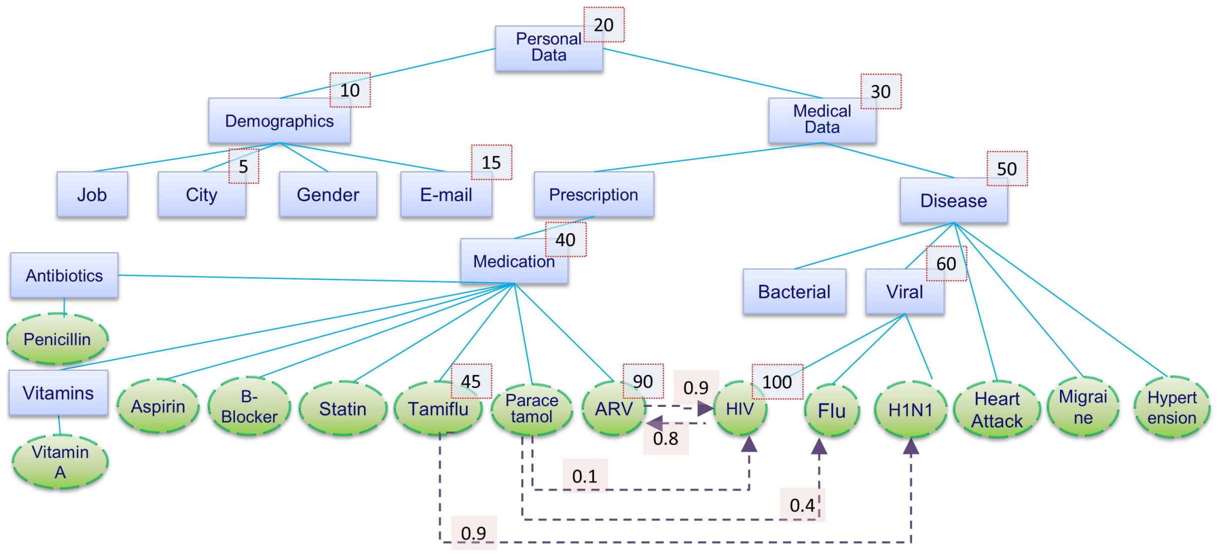


Fig. 2. Data Model Example

a viral disease. Inference relations are represented by dashed edges. For example, the inference relation between Anti-RetroViral (ARV) and HIV indicates that there is a correlation between ARV medication and HIV virus.

Nodes can be annotated with a sensitivity label that indicates the sensitivity of the data represented by the respective node. For instance, the sensitivity label of node HIV is $SL(HIV) = 100$. Inference relations are annotated with a label that is used to represent the degree of inference. For instance, label $PL(ARV,HIV) = 0.9$ indicates that a patient treated with ARV medication is very likely (90%) infected by HIV.

Both domain and security experts need to be involved in the construction of the data model for a given domain. Domain experts should define data types and instances along with the hierarchy relations between them. Moreover, they should determine the inference relations between instances with the respective probability labels. On the other hand, security experts should annotate the data model with sensitivity labels.

The annotation of the data model with sensitivity labels, however, can be difficult as the number of nodes can be large. Ideally, security experts should be able to assign sensitivity labels to few nodes and, based on this initial assignment, the system should determine the sensitivity of the other nodes. Here, we provide the machinery to reason over the data model in order to determine data sensitivity. In Section 8.2 we evaluate the effectiveness and efficiency of such an approach. To reason over the hierarchical structure underlying the data model, we introduce the notion of *sensitivity propagation*.

Definition 2 *Let $DM = (T, I, HR, IR, SL, PL)$ be a data model.* Sensitivity propagation *is a function $SP : T \cup I \to \mathbb{R}_0^+$ such that given a node $x \in T \cup I$*

$$SP(x) = \begin{cases} SL(x) & if\ SL(x)\ exists \\ \max\{SP(y) | y \in T \wedge (y,x) \in HR\} & otherwise \end{cases} \tag{1}$$

Sensitivity propagation is used to assign a sensitivity value to the nodes in the data model based on hierarchy relations. In particular, Equation 1 encodes a propagation strategy similar to the most specific overrides strategy presented in [26,33]: if a node does not have a sensitivity label, then its sensitivity is inherited from the node higher in the hierarchy. This approach provides great flexibility in the definition of sensitivity annotations as it allows a security expert to model cases where the sensitivity of a node is lower than the sensitivity of the parent node. It is worth noting that a node can have more than one parent node. In this case, a node is assigned the sensitivity value of the parent node that has the highest sensitivity value (among all parent nodes of the node).

Example 1 *In Fig. 2 the sensitivity label of* Paracetamol *is not defined. Therefore, this node inherits the sensitivity value of the parent node (*Medication*), i.e.,* $SP(Paracetamol) = SP(Medication) = 40$.

Although sensitivity propagation simplifies the task of assigning sensitivity values to nodes, it may lead to an inaccurate assignment by underestimating the sensitivity of some pieces of information. For instance, a security expert might assign a low sensitivity value to a data instance whose disclosure allows the inference of a data instance annotated with a higher sensitivity value [36]. To this end, we use inference relations to validate the propagated values and eventually adjust the sensitivity of the nodes to a higher value.

Definition 3 *Let* $DM = (T, I, HR, IR, SL, PL)$ *be a data model.* Node sensitivity *is a function* $NS : T \cup I \to \mathbb{R}_0^+$ *such that given a node* $x \in T \cup I$

$$NS(x) = \max\{SP(x), IS(x)\} \tag{2}$$

where $SP(x)$ *is the sensitivity derived through sensitivity propagation (Definition 2) and the* inferred sensitivity $IS(x)$ *is computed using function* $IS : T \cup I \to \mathbb{R}_0^+$*:*

$$IS(x) = \begin{cases} \sum_{(x,y)\in IR} PL(x,y) \times NS(y) & if\ x \in I \\ 0 & if\ x \in T \end{cases} \tag{3}$$

Intuitively, the sensitivity of data types is obtained through sensitivity propagation. On the other hand, the sensitivity of data instances also depends on the sensitivity of the information that can be inferred through the inference relations.

Example 2 *Consider the sensitivity value of node* Paracetamol *calculated in Example 1. The node has an inference relation with nodes* HIV *and* Flu*, which have sensitivity* 100 *and* 60 *respectively. The inferred sensitivity is* 34*. As this value is lower than the sensitivity obtained through propagation, the node sensitivity for* Paracetamol *is* 40.

5. Mapping Information on the Data Model

To quantify the severity of data leakages, the leaked data have to be mapped onto the data model in order to determine their sensitivity. Recall that in this work we focus on structured data leaving a database. Thus, the mapping consists in determining, for each entry in the leaked table, the corresponding node in the data model. In this section we first introduce the notation used to represent data; then, we present the mapping.

Let $\mathcal{A}$ be a set of attributes. Attributes can be classified in two (possibly non disjoint) sets: *quasi-identifiers* and *sensitive* attributes. Quasi-identifiers $Q = \{q_1, \ldots, q_k\} \subseteq \mathcal{A}$ can be used to reveal the identity of an individual, possibly using an external data source (any subset of quasi-identifiers is a quasi-identifier itself). Sensitive attributes $S = \{s_1, \ldots, s_m\} \subseteq \mathcal{A}$ are the attributes that need to be protected and are used to evaluate the severity of exposing the data. Note that the distinction between quasi-identifiers and sensitive attributes has been introduced to illustrate the quantification metric and, in particular, to highlight how different aspects contribute to the assessment of the severity of data leakages. In practice, certain attributes may belong to both sets. For instance, the gender of a person is a quasi-identifier, as it can be used to partly reveal an individual's identity. Moreover, according to the proposal of the new EU Data Protection Regulations by the LIBE Committee[2] the "gender identity" of an individual is considered to be sensitive personal information. Thus, gender can be considered as both a quasi-identifier and a sensitive attribute. Our framework allows a security expert to decide how to classify attributes based on the purpose and context of use. In particular, a security expert can decide to consider an attribute a only to determine the sensitivity of leaked data (i.e., $a \in S$), only to determine the identifiability factor (i.e., $a \in Q$), or both (i.e., $a \in S \cap Q$).

A database table $D(a_1, \ldots, a_n)$ is a set of records over a set of attributes $\{a_1, \ldots, a_n\} \subseteq \mathcal{A}$. We denote the records in $D(a_1, \ldots, a_n)$ 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. Attributes take values from a close set of values defined by their domain. Given an attribute $a \in \mathcal{A}$, C_a denotes the domain of a.

We assume that all attributes and the corresponding domains are defined in the data model. In particular, attributes correspond to data types in the data model. Formally, $\mathcal{A} \subseteq T$. The values of an attribute can correspond either to an instance or a data type node, which is located in the subtree of the data model rooted in the node corresponding to the attribute. Formally, $\{C_a\}_{a \in \mathcal{A}} \subseteq T \cup I$.

Next we present an example of database table along with the classification of attributes used in the remainder of the paper.

Example 3 *Consider the scenario in Section 2. The database includes table D(Job, City, Gender, Disease, Medication). Attributes Job and City are quasi-identifiers, while attributes Disease and Medication are sensitive attributes. Attribute Gender belongs to both sets. Each sensitive attribute takes values from a pre-specified domain. For instance, Disease can take a value from {HIV, Heart Attack, Hypertension, Migraine, H1N1, Flu}, and Medication from {ARV, b-Blocker, Tamiflu, Statin, Antibiotics, Aspirin, Paracetamol, Vitamins}. We assume that a doctor can prescribe antibiotics without referring explicitly to a particular medical product, allowing a patient and/or a pharmacist to choose an antibiotic from a list of equivalent medication.*

We have now the machinery to present how to obtain the sensitivity of leaked data. Suppose that a table $D(a_1, \ldots, a_n)$ is leaked and S is the set of sensitive attributes. We are interested in the sensitivity of the sensitive attributes that are leaked, i.e. $\{a_i | a_i \in \{a_1, \ldots, a_n\} \cap S\}$. As mentioned above, quasi-identifies are only used to determine the identifiability factor (unless they are also declared by the security expert as sensitive attributes) and their sensitivity is not accounted for in the computation of the severity of data leakages.

Note that, in addition to the attributes and values contained in the leaked table, we also consider pre-acquired knowledge as part of the leaked information. In particular, conditional clauses such as WHERE clauses in SQL may leak information. For instance, consider a user query requesting the medication

[2]Note that the new EU Data Protection Regulations is not passed yet.

prescribed to patients infected by HIV (i.e., WHERE Disease = 'HIV'). Although the leaked table only contains values concerning attribute Medication, we also assume that value HIV is leaked. For the sake of simplicity, hereafter we explicitly represent the attributes (and values) corresponding to pre-acquired knowledge in the leaked table.

To obtain the sensitivity of the leaked data, the values of the sensitive attributes in the leaked table need to be mapped onto the data model. For the mapping, search methods can be employed. However, the efficiency of the search methods depends on the size of the data model. To facilitate the search process, the attributes in the leaked table can be first mapped to the corresponding data type node in the data model. The value of the attribute can be then mapped starting the search from the data type node corresponding to the attribute and continuing downward the hierarchy defined by the data model.

Example 4 *Consider table D(Job, City, Gender, Disease, Medication) in Example 3 and the data model in Fig. 2. Suppose that a leaked record contains value Hypertension for attribute Disease. First, Disease is mapped by searching from node Personal Data downward the hierarchy until a data type node with the same name is found. Then, value Hypertension is mapped by searching the corresponding node from node Disease.*

6. Data Leakage Quantification

The estimation of the severity of data leakages requires metrics that assess the sensitivity and the amount of the data leaked. In this section, we present an overview of M-Score [20] and study its accuracy by applying it to some data leakage samples. Based on this analysis, we present our proposal for data leakage quantification.

6.1. M-Score

M-Score [20] has been proposed to estimate data misuse in database environments. It is based on the calculation of the severity of a (portion of) table, which may have been leaked. M-Score evaluates the severity of a data leakage by evaluating three main aspects of the leaked data: the sensitivity, quantity and distinguishing factor. The sensitivity of data is defined through a *sensitivity score* function.

Definition 4 *Let $\mathcal{A}$ be a set of attributes and C_a the domain of an attribute $a \in \mathcal{A}$. The sensitivity score function $f : C_a \to [0,1]$ assigns a sensitivity value to each value in C_a.*

Given a record $r \in R^{D(a_1,\ldots,a_n)}$, the sensitivity score of a value $a_i[r] \in C_{a_i}$ is denoted by $f(a_i[r])$. The sensitivity of a record is captured by the *raw record score*. In particular, the calculation of the raw record score of a record r, denoted as RRS_r, encompasses the sensitive attributes of a table and their values in r.

Definition 5 *Let $D(a_1,\ldots,a_n)$ be a table, $S = \{s_l,\ldots,s_m\} \subseteq \mathcal{A}$ the set of sensitive attributes in $D(a_1,\ldots,a_n)$, and f the sensitivity score function. Given a record $r \in R^{D(a_1,\ldots,a_n)}$, the* raw record score *of r is*

$$RRS_r = \min(1, \sum_{s_i \in S} f(s_i[r])) \tag{4}$$

Intuitively, the raw record score of a record is obtained by summing the sensitivity score of every piece of sensitive information in the record, with a maximum of 1.

The *distinguishing factor* of a record r with respect to a table, denoted as $DF_r^{D(a_1,\dots,a_n)}$, is the amount of efforts required to identify the individual which r refers to. The distinguishing factor of a record is calculated on the basis of quasi-identifier attributes.

Definition 6 *Let $D(a_1,\dots,a_n)$ be a table, $Q=\{q_l,\dots,q_k\}\subseteq\mathcal{A}$ the set of quasi-identifier attributes in $D(a_1,\dots,a_n)$ and $r\in R^{D(a_1,\dots,a_n)}$ a record in $D(a_1,\dots,a_n)$. Given $\{q_1[r],\dots,q_k[r]\}$ with $q_i[r]\in C_{q_i}$ the set of quasi-identifier values in r, the* distinguishing factor *of r with respect to $D(a_1,\dots,a_n)$ is*

$$DF_r^{D(a_1,\dots,a_n)}=\frac{1}{|R'|} \tag{5}$$

where $R'=\{r_i|\forall q_i\in Q\ q_i[r]=q_i[r_i]\}$, i.e. the set of records in $D(a_1,\dots,a_n)$ that have $\{q_1[r],\dots,q_k[r]\}$ as quasi-identifier values and $|R'|$ is the number of such records.

The *final record score* for a leaked table $L(a_1,\dots,a_m)$, denoted as RS_L, is calculated based on the raw record score and distinguishing factor.

Definition 7 *Let $ST(a_1,\dots,a_n)$ be a source table and $L(b_1,\dots,b_m)$ a leaked table with $\{b_1,\dots,b_m\}\subseteq\{a_1,\dots,a_n\}$. Given the raw record score RRS_r and distinguishing factor $DF_r^{ST(a_1,\dots,a_n)}$ for every record $r\in R^{L(b_1,\dots,b_m)}$, the* final record score *of $L(b_1,\dots,b_m)$ is*

$$RS_L=\max_{r\in R^{L(b_1,\dots,b_m)}}(RRS_r\times DF_r^{ST(a_1,\dots,a_n)}) \tag{6}$$

It is worth noting that the distinguishing factor is calculated with respect to the source table. To capture the quantity aspect of the leakage, M-Score determines the severity of leakages based on the final record score and the number of records disclosed.

Definition 8 *Let $L(a_1,\dots,a_n)$ be a leaked table. Given the final record score RS_L of $L(b_1,\dots,b_m)$, the* M-Score *of $L(a_1,\dots,a_n)$ is*

$$M\text{-}Score_L=|R^{L(a_1,\dots,a_n)}|^{\frac{1}{x}}\times RS_L \tag{7}$$

where $|R^{L(a_1,\dots,a_n)}|$ represents the number of records in $L(a_1,\dots,a_n)$ and $x\in\mathbb{Z}_{>0}$.

Parameter x is a weighting factor used to determine the importance of the amount of data (versus their sensitivity) in the computation of the severity of data leakages. In particular, greater is x, more importance is given to the sensitivity of data compared to the amount of leaked data. In the next section, we analyze the impact of this parameter on the severity of leakages.

6.2. Analysis of M-Score

In this section we study the accuracy of M-Score by applying it to a number of leakage examples. These examples are based on table D(Job, City, Gender, Disease, Medication) presented in Section 5. In the examples we analyze the calculation of the severity of leakages with respect to amount and sensitivity of the leaked data. Therefore, we assume that the distinguishing factor is the same for all leakages.

Disease		Medication	
$f(HIV) = 1$	$f(Migraine) = 0.3$	$f(ARV) = 1$	$f(Antibiotics) = 0.4$
$f(HeartAttack) = 0.7$	$f(Flu) = 0.1$	$f(b\text{-}Blocker) = 0.8$	$f(Aspirin) = 0.3$
$f(Hypertention) = 0.6$		$f(Statin) = 0.6$	$f(Paracetamol) = 0.1$
$f(H1N1) = 0.4$		$f(Tamiflu) = 0.5$	$f(Vitamins) = 0.1$

Table 1

Sensitivity score function

The sensitivity score function used to assess the severity of leakages is shown in Table 1. The sensitivity score assigned to diseases is related to the impact the disclosure of disease information has on the life of an individual. In particular, diseases whose disclosure has a major impact on the life of the patient (e.g., HIV) are assigned a higher sensitivity than diseases with less critical impact (e.g., Flu). The sensitivity of medication is related to its degree of specialization; medication can be general and specialized. General medication is prescribed to treat mild symptoms of different diseases, such as headache. This category includes medication such as Antibiotics, Aspirin and Paracetamol. Specialized medication is prescribed to treat symptoms related to a particular disease. For instance, ARV is usually prescribed to patients infected with HIV. We assume that specialized medication has higher sensitivity than general medication.

We apply M-Score to three cases. In the first two cases, we focus on the impact of data sensitivity on the severity of leakages. Thus, we consider the same number of leaked records and set parameter x of M-Score equal to 1. In the third case, we focus on the impact of the amount of leaked records on the severity of leakages. Thus, we vary the number of records in the leakages. In M-Score the impact of the amount of records is accounted by parameter x (see Definition 8). To study the effect of this parameter we consider three values for x, namely $x = 1$, $x = 10$ and $x = 100$.

Case 1: Consider the leakages in Tables 2a and 2b. In *Case* 1.1 the records contain general medication prescribed to patients suffering from serious health issues. In *Case* 1.2 the records contain information about specialized medication prescribed to patients suffering from serious health issues. We expect *Case* 1.2 to be more severe than *Case* 1.1 as it contains more sensitive information. However, M-Score calculates the same severity value (2.000) in both cases. The problem lies in the use of the *min* function in the calculation of *RRS*. In particular, this measure has an upper bound equal to 1, which leads to the same *RRS* for all records whose sensitivity is greater than 1.

Case 2: Consider the leakages in Tables 2c and 2d. In *Case* 2.1 the records contain general medication. In contrast, the records in *Case* 2.2 contain information about specialized medication. In both cases we consider only a small percentage of records (1 record) about patients suffering from a serious health issue. Therefore, *Case* 2.2 should be estimated more severe than *Case* 2.1, as it contains more sensitive information. In contrast, M-Score calculates the same severity value (2.000) in both cases. The problem lies in the use of the *max* function in the calculation of *RS*. In particular, *RS* only accounts for the sensitivity value of the record that has the highest sensitivity, regardless of the sensitivity value of the other records in the leaked table.

Case 3 Consider the data leakages in Tables 2e and 2f. *Case* 3.1 is composed of records with general medication with only a small percentage of records (1 record) referring to patients suffering from a serious health issue. *Case* 3.2 is the same of *Case* 1.2. A notable difference between the two leakages is the number of leaked records. *Case* 3.1 contains seven records, whereas *Case* 3.2 contains four records. Although *Case* 3.2 has fewer records, it should be more severe than *Case* 3.1, since the sensitivity of data in *Case* 3.2 is much higher than in *Case* 3.1. As mentioned above, we study three values of parameter

Job	**City**	**Gender**	**Disease**	**Medication**
Lawyer	LA	Male	HIV	Vitamins
Lawyer	LA	Male	Heart Attack	Aspirin
Lawyer	LA	Male	Migraine	Paracetamol
Lawyer	LA	Male	Hypertension	Aspirin
M-Score: 2.000				

(a) Case 1.1

Job	**City**	**Gender**	**Disease**	**Medication**
Lawyer	LA	Male	HIV	ARV
Lawyer	LA	Male	Hypertension	Statin
Lawyer	LA	Male	Heart Attack	b-Blocker
Lawyer	LA	Male	Migraine	b-Blocker
M-Score: 2.000				

(b) Case 1.2

Job	**City**	**Gender**	**Disease**	**Medication**
Lawyer	LA	Male	HIV	Vitamins
Lawyer	LA	Male	Flu	Paracetamol
Lawyer	LA	Male	Flu	Aspirin
Lawyer	LA	Male	Migraine	Aspirin
M-Score: 2.000				

(c) Case 2.1

Job	**City**	**Gender**	**Disease**	**Medication**
Lawyer	LA	Male	HIV	ARV
Lawyer	LA	Male	H1N1	Tamiflu
Lawyer	LA	Male	H1N1	Antibiotics
Lawyer	LA	Male	Flu	Antibiotics
M-Score: 2.000				

(d) Case 2.2

Job	**City**	**Gender**	**Disease**	**Medication**
Lawyer	LA	Male	HIV	Vitamins
Lawyer	LA	Male	Flu	Paracetamol
Lawyer	LA	Male	Flu	Aspirin
Lawyer	LA	Male	Migraine	Aspirin
Lawyer	LA	Male	Migraine	Paracetamol
Lawyer	LA	Male	H1N1	Aspirin
Lawyer	LA	Male	H1N1	Paracetamol
$x = 1$	M-Score: 3.500			
$x = 10$	M-Score: 0.607			
$x = 100$	M-Score: 0.509			

(e) Case 3.1

Job	**City**	**Gender**	**Disease**	**Medication**
Lawyer	LA	Male	HIV	ARV
Lawyer	LA	Male	Hypertension	Statin
Lawyer	LA	Male	Heart Attack	b-Blocker
Lawyer	LA	Male	Migraine	b-Blocker
$x = 1$	M-Score: 2.000			
$x = 10$	M-Score: 0.574			
$x = 100$	M-Score: 0.507			

(f) Case 3.2

Table 2
M-Score evaluation

x (see Definition 8). When $x = 1$, the value of M-Score is 3.500 for *Case* 3.1 and 2.000 for *Case* 3.2. These results are different from what expected. The main reason lies in the use of *min* and *max* functions for determining *RRS* and *RS* as discussed in Case 1 and Case 2. In the second case ($x = 10$), we give a balanced importance to both the amount of rows. Accordingly, the value of M-Score is 0.607 for *Case* 3.1 and 0.574 for *Case* 3.2. Similarly to the first case, M-Score calculates a higher severity value for *Case* 3.1. In the third case ($x = 100$), the M-Score values of the two leakages converge around 0.50, failing to discriminate them.

As shown in the examples above, M-Score may not be able to accurately determine the severity of leakages; in particular, this metric only provides an estimation of the severity of data leakages rather than computing their actual severity. The main problem lies in the way in which the raw record score and the final score are calculated and, in particular, in the use of the *min* and *max* functions respectively. The *min* function allows a maximum sensitivity score of 1 per record, thus failing to account for the actual sensitivity of a record. In particular, M-Score is not able to discriminate data leakages that contain at least one highly sensitive record (i.e., $\sum_{s_i \in S} f(s_i[r]) \geq 1$).[3] The *max* function used in the computation of the final record score leads to consider only the record with the highest raw record score when calculating M-Score, which may result in a counterintuitive estimation of the sensitivity of a leaked table.[4] These imperfections can be exacerbated when the amount of records leaked is considered. Indeed, M-Score computes the severity of a data leakage as the product of the final record score and the number of records in the leaked table. The importance of the amount of records leaked is expressed by parameter x of M-Score. For low values of x (i.e., $x \approx 1$) considerable importance is given to the amount of records. In particular, the severity of a leakage only depends on the record with the highest raw record score and the number of records, regardless of the sensitivity of the other records in the leaked table. Thus, leakages with a larger number of records can result to be more severe than leakages disclosing a smaller number, but highly sensitive records, even if there is only one record that is highly sensitive. Otherwise, for $x \gg 1$, more importance is given to sensitivity. In particular, M-Score converges to the highest raw record score for $x \to \infty$. As discussed above, this value does not reflect the actual sensitivity of the leaked data. Last but not least, M-Score requires an analyst define an appropriate value for parameter x, increasing the complexity in the use of the metric.

6.3. L-Severity

This section presents L-Severity, a new metric for quantifying data leakages that addresses M-Score's drawbacks. Similarly to M-Score, L-Severity assesses the severity of data leakages based on the sensitivity, distinguishing factor and amount of leaked data. However, we pursue a different approach than M-Score in that we propose a metric that computes the actual severity of data leakages rather than an estimation. In particular, our metric computes the severity of data leakages by considering the sensitivity of every piece of data in the leaked table.

Definition 9 *Let $ST(a_1,\ldots,a_n)$ be a source table, $L(b_1,\ldots,b_m)$ a leaked table with $\{b_1,\ldots,b_m\} \subseteq \{a_1,\ldots,a_n\}$, $S = \{s_l,\ldots,s_m\} \subseteq \mathcal{A}$ the set of sensitive attributes in $L(b_1,\ldots,b_m)$ and $DM = (T,I,HR,IR,SL,PL)$ a data model. Given $r \in R^{L(b_1,\ldots,b_m)}$ a record in $L(b_1,\ldots,b_m)$ and $DF_r^{ST(a_1,\ldots,a_n)}$, the* record sensitivity *of r is*

$$RSENS_r = DF_r^{ST(a_1,\ldots,a_n)} \times \sum_{s_i \in S} NS(s_i[r]) \tag{8}$$

where NS is the node sensitivity of the node in the data model that corresponds to the value $s_i[r]$ of a sensitive attribute s_i.

[3]Consider a record r_1 with sensitivity equal to 1 (i.e., $\sum_{s_i \in S} f(s_i[r_1]) = 1$) and a record r_2 with sensitivity equal to 5 (i.e., $\sum_{s_i \in S} f(s_i[r_2]) = 5$). The raw record score of both these records is equal to 1 (i.e., $RRS_{r_1} = RRS_{r_2} = 1$), even if the second record is much more sensitive than the first one.

[4]Consider a leaked table $L_1^{(a_1,\ldots,a_n)}$ consisting of nine records with raw record score equal to 0.1 and one record with raw record score equal to 1 and a leaked table $L_2^{(b_1,\ldots,b_m)}$ consisting of ten records with raw record score equal to 0.9 (for the sake of simplicity, assume that the distinguishing factor is equal for all records in the two tables). The final record score for the first table is higher than the final record score of the second table (i.e., $RS_{L_1^{(a_1,\ldots,a_n)}} = 1$ vs. $RS_{L_2^{(b_1,\ldots,b_m)}} = 0.9$), even if the data leaked in the second table as a whole are much more sensitive than the ones leaked in the first table.

Case	**L-Severity**	**M-Score**		
		$x = 1$	$x = 10$	$x = 100$
Case 1.1	2.050	2.000		
Case 1.1	2.900	2.000		
Case 2.1	1.150	2.000		
Case 2.2	2.100	2.000		
Case 3.1	1.950	3.500	0.607	0.509
Case 3.2	2.900	2.000	0.574	0.507

Table 3
Comparison between L-Severity and M-Score

In the calculation of record sensitivity we make use of the data model (Section 4). In particular, we use *NS* to calculate the sensitivity of each sensitive attribute value in a record. For the sake of simplicity, we have used the distinguish factor proposed by M-Score to measure the anonymity level of data. However, other metrics have been proposed to assess the anonymity level of data, e.g. k-anonymity [42], ℓ-diversity [30] and t-closeness [29]. An interesting direction for future work is to understand and evaluate the impact of these metrics on the computation of the severity of data leakages.

To calculate the severity of data leakages, we introduce L-Severity metric.

Definition 10 *Let $ST(a_1,\ldots,a_n)$ be a source table and $L(b_1,\ldots,b_m)$ a leaked table with $\{b_1,\ldots,b_m\} \subseteq \{a_1,\ldots,a_n\}$. Given the record sensitivity $RSENS_r$ for each record $r \in R^{L(b_1,\ldots,b_n)}$, the* leakage severity (L-Severity) *of $L(b_1,\ldots,b_m)$ is*

$$L\text{-}Severity_L = \sum_{r \in R^{L(b_1,\ldots,b_m)}} RSENS_r \tag{9}$$

To demonstrate L-Severity we applied it to the same cases used to evaluate M-Score (Section 6.2). To make a fair comparison with M-Score we use the sensitivity score function in Table 1 to determine the sensitivity of data.

A summary of the severity scores obtained by L-Severity and M-Score is shown in Table 3. Accordingly, the value of L-Severity is 2.050 for *Case* 1.1 and 2.900 for *Case* 1.2. Hence, L-Severity is higher for *Case* 1.2 than for *Case* 1.1. Similarly, the L-Severity value for *Case* 2.2 (2.100) is higher that the value for *Case* 2.1 (1.150). Finally, *Case* 3.2 has a higher L-Severity value (2.900) than *Case* 3.1 (1.950). Thus, L-Severity provides values that better characterize the severity of leakages with respect to the intuition (Section 6.2).

7. From Theory to Practice

In this section we discuss the practical application of the proposed data leakage quantification approach along with its potential benefits for a security analyst. First, we show how existing domain ontologies can be employed as a baseline for the data model. Then, we discuss issues concerning the computation of data sensitivity. Finally, we demonstrate how the data leakage quantification approach can be used to facilitate the analysis of alerts.

Data Model	**Ontology**
Data type	Class
Data instance	Individual
Hierarchy relation	*IS-A* (is-a-subclass-of) relations
Inference relation	Relations between individuals
Sensitivity label	Attribute of classes and individuals
Inference label	Attribute of relations between individuals

Table 4

Correspondence between the Data Model and Ontologies

7.1. Defining the Data Model

The severity of data leakages depends on the sensitivity of the leaked data. To reason on data sensitivity we employ a data model that provides a description of the data characterizing an application domain along with the semantic relations between data. In our approach we take advantage of existing domain ontologies to represent the data model. Ontologies [19] are often adopted to capture the knowledge of a specific application domain. The basic elements of ontologies are *Classes, Individuals, Attributes and Relationships*. Classes are abstract groups of objects, while individuals represent instances of classes. Attributes are used to represent properties and characteristics of classes and individuals. Relationships represent ways in which classes and individuals are related to one another. Ontologies can be used as a basis for the definition of a data model. Table 4 shows the correspondence between the elements of the data model and the elements of an ontology.

As an example we illustrate how the data model can be instantiated in the healthcare domain using an existing ontology. Several ontologies have been proposed for the healthcare domain [14,34,35,41,45,46]. Here, we consider SNOMED Clinical Terms (SNOMED-CT) [41], one of the most exhaustive and used medical ontologies. Fig. 3 presents a fragment of SNOMED-CT related to the AIDS disease and the HIV virus. In SNOMED-CT classes and individuals represent clinical terms. SNOMED-CT uses relationships to link each term to other terms. These relationships provide formal definitions of medical terms. One type of relationship in SNOMED-CT is the *IS-A* relationship which relates a term to a more general term. Based on this relationship, medical terms are organized in hierarchies, from the general to the specific. Therefore, the *IS-A* relationship can be mapped to the hierarchy relation of the data model. SNOMED-CT also uses other types of relationship to define a term. For instance, relation *ASSOCIATED WITH* is used to relate an individual of class Disease to an individual of class Substance. Relation *CAUSATIVE AGENT* is used to relate an individual of class Disease to an individual of class Organism. For instance, relation *CAUSATIVE AGENT* is used to relate the term Acquired Immune Deficiency Syndrome (AIDS) to the term Human Immunodeficiency Virus (HIV), indicating that HIV is an organism and AIDS is the condition it causes. Relations between individuals can be seen as a realization of our inference relation, as they make it possible to obtain additional information based on the knowledge of a specific instance, e.g. by knowing that a patient have AIDS, one can infer that the patient is HIV-positive.

SNOMED-CT does not provide a native support to specify data sensitivity and inference labels. To this end, SNOMED-CT can be extended by annotating terms and relations with two additional attributes. In particular, sensitivity labels can be specified as an attribute of classes and individuals, and inference labels as an attribute of relations.

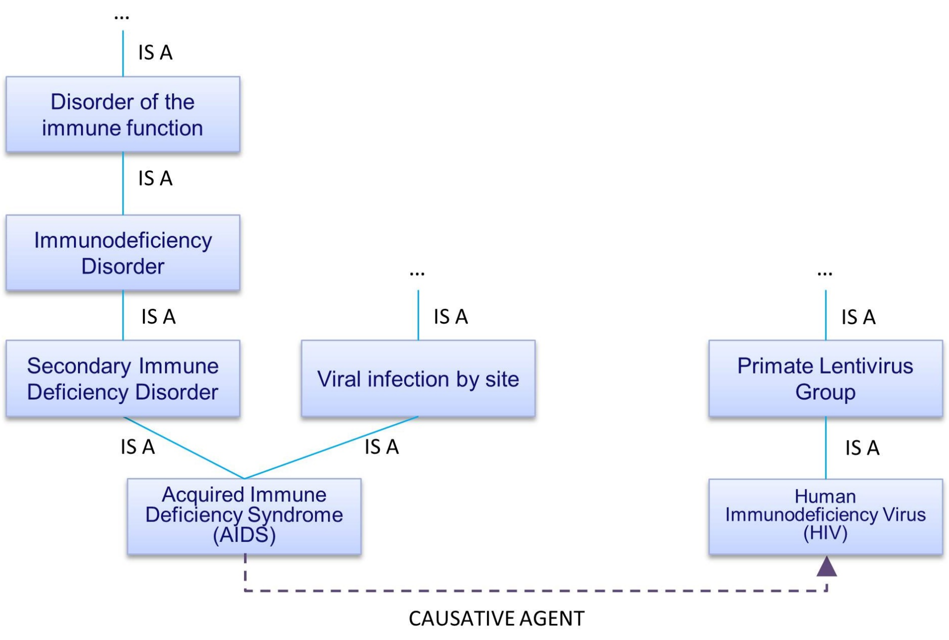

...
IS A
Disorder of the immune function
IS A
Immunodeficiency Disorder
IS A
Secondary Immune Deficiency Disorder
...
IS A
Viral infection by site
...
IS A
Primate Lentivirus Group
IS A
IS A
Acquired Immune Deficiency Syndrome (AIDS)
IS A
Human Immunodeficiency Virus (HIV)
CAUSATIVE AGENT

Fig. 3. Fragment of SNOMED-CT

7.2. Determining Data Sensitivity

As shown in Section 4, the data model makes it possible to determine a sensitivity value for each piece of data based on an initial (partial) assignment. Therefore, such an assignment plays a crucial role in the quantification of leakages. Different solutions have been proposed in the literature to define the sensitivity of data. These solutions can be divided in two main streams. On one side, we can find user-centric solutions [11,31] which require users to define sensitivity values for the data based on their preferences and context of use. Users may use different criteria to define such values. For instance, medical personnel may express data sensitivity as the impact that the disclosure of the data has on the patient. In contrast, administrative personnel may express sensitivity as the financial impact that the disclosure of the data has to the healthcare institution. Thus, the resulting annotation contains a degree of subjectivity that depends on the criteria and belief of the user defining sensitivity values. On the other side, we can find approaches based on guidelines and standards. These approaches aim to provide a standardized and consistent ground for sensitivity annotations. Contrarily to user-centric solutions, solutions relying on standards and guidelines limit user intervention, thus reducing the degree of subjectivity in the annotation.

In our approach we rely on HL7 Healthcare Privacy and Security Classification System (HCS) [22] to annotate the data model with sensitivity labels. HL7 HCS provides guidelines and a tagging system for automated labeling and segmentation for protecting healthcare information. We have chosen this tagging system for two main reasons. First, it is largely accepted by the community as demonstrated by the several projects and initiatives contributing and implementing these guidelines [4,13,23,24,44]. Moreover, the tagging system provided by HL7 HCS is based on SNOMED-CT (and other code systems) and, thus, well suited for our data model.

HL7 HCS proposes to annotate data with security labels which provide a structured representation of various security aspects concerning a piece of data, such as confidentiality level and sensitivity. The con-

HL7 Confidentiality tag	**Sensitivity value**
Very Restricted	$\geqslant 100$
Restricted	$\geqslant 80$
Normal	$\geqslant 60$
Moderate	$\geqslant 40$
Low	$\geqslant 20$
Unrestricted	0

Table 5

Mapping from HL7 confidentiality tags to sensitivity value

fidentiality level is the most important aspect and is used to indicate the classification of healthcare information. Its values range from *Unrestricted* to *Very Restricted*, defining six different confidentiality levels in total. Sensitivity tags define compartments of IT resource by categorizing their value and importance. For instance, tag *HIV* is used to denote the sensitivity of HIV/AIDS information, tag *SEX* the sensitivity of sexuality and reproductive health information, *PSY* the sensitivity of psychiatry information and so on. Sensitivity tags are associated with guidelines that define on how strictly pieces of data should be handled. For instance, HIV/AIDS information is deemed highly sensitive and its handling is subject to stringent requirements.

However, HL7 HCS only provides a "qualitative" annotation of healthcare information. To this end, we have extended the HL7 HCS tagging system to define sensitivity values from confidentiality and sensitivity tags. First, we defined an initial sensitivity value for pieces of data according to their confidentiality level as shown in Table 5. For instance, every piece of data that has a *Restricted* confidentiality level are assigned with a sensitivity value of 80. These initial sensitivity values are then adjusted based on sensitivity tags (if defined for the piece of data) to comply with the respective sensitivity guidelines. For example, Penicillin has *Normal* confidentiality level and no sensitivity tag (Table 6). Therefore, it is assigned a sensitivity value equal to 60, i.e. the initial value is not adjusted. On the other hand, Acute HIV infection and Bipolar disorder have been tagged in HL7 HCS with *Restricted* confidentiality level, thus they should be assigned a sensitivity value of at least 80. Moreover, Acute HIV infection is associated with sensitivity tag *HIV*, which indicates that this information should be handled as highly sensitive information, whereas Bipolar disorder is associated with sensitivity tag PSY, which requires less strict handling. Based on these sensitivity tags, we adjust the initial sensitive value assigned to Acute HIV infection and Bipolar disorder by assigning a sensitivity value of 100 to Acute HIV infection and a sensitivity value of 90 to Bipolar disorder.

HL7 HCS provides a partial classification of classes and individuals in SNOMED-CT. We used the resulting sensitivity values for these classes and individuals as the initial assignment of sensitivity values in the data model. Then, we derived the sensitivity for other classes and individuals using the approach described in Section 4. In particular, the *IS-A* relationship was used for sensitivity propagation, and the other types of relationship were used to compute inferred sensitivity.

7.3. Analyzing Alerts

Organizations often employ a DLD system to detect potential data leakages. DLD systems, however, may produce a huge number of alerts, making their evaluation a time-consuming and error-prone task. To properly handle alerts and mitigate the potential damages caused by data leakages, security analysts should be able to timely focus on the most critical cases and have the information necessary for the investigation.

Data	HL7 Confidentiality tag	HL 7 Sensitivity tag	Sensitivity value
Acute HIV infection	Restricted	HIV	100
Bipolar disorder	Restricted	PSY	90
AZT (Zidovuline)	Restricted	HIV	100
Penicillin	Normal	-	60
Paracetamol	Moderate	-	40

Table 6

Sample HL7 security tags and assigned sensitivity values

To facilitate the task of security analysts, we have developed a web-based data leakage audit tool for the visualization of alerts. The tool is connected to a DLD system. When an alert is raised by the DLD solution, the tool analyzes the alert by mapping the leaked data to the data model and assessing its severity. The alerts are then visualized along with information that provides insights on the security incident.

A screenshot of the tool is shown in Fig. 4. For each alert, the tool displays the associated profile (e.g., user, role) along with the performed query (second and fourth columns). Moreover, other contextual information, such as the timestamp and IP address, are shown (fifth and sixth columns respectively). The tool also displays the severity of data leakages (last column). In particular, severity is visualized using a three value scale, namely High Severity (shown in Red), Medium Severity (shown in Orange) and Low Severity (shown in Yellow). Note 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.

Depending on the adopted DLD solution, other information about the alerts may be available for the analysis. We have connected our quantification system to the DLD solution presented in [10]. This DLD solution is a white-box behavioral-based anomaly detection system which provides information related to the root causes of alerts. This information can be displayed by the visualization tool to provide additional insights on the alert (third column).

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 severity or group alerts with respect to other aspects like profile, enabling security analysts to focus on specific types of alerts.

8. Experiments

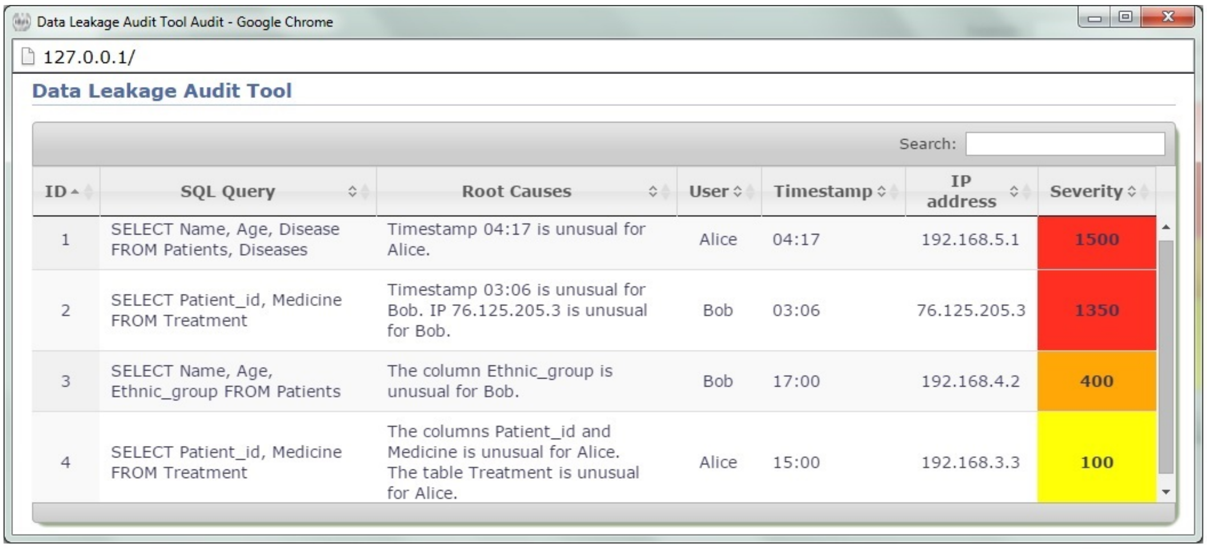
In this section we evaluate the severity-based data leakage quantification approach (Section 6) and the data sensitivity calculation method using the data model (Section 4).

8.1. Assessing Data Leakage Severity

In this section we evaluate the ability of the severity-based data leakage quantification approach to assess the criticality of data leakages. In addition, we compare our solution with M-Score.

Experiment Setting For the evaluation we have considered a real setting based on the healthcare scenario presented in Section 2. The hospital database was implemented using GNU Health[5], a healthcare

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



Data Leakage Audit Tool

ID	SQL Query	Root Causes	User	Timestamp	IP address	Severity
1	SELECT Name, Age, Disease FROM Patients, Diseases	Timestamp 04:17 is unusual for Alice.	Alice	04:17	192.168.5.1	1500
2	SELECT Patient_id, Medicine FROM Treatment	Timestamp 03:06 is unusual for Bob. IP 76.125.205.3 is unusual for Bob.	Bob	03:06	76.125.205.3	1350
3	SELECT Name, Age, Ethnic_group FROM Patients	The column Ethnic_group is unusual for Bob.	Bob	17:00	192.168.4.2	400
4	SELECT Patient_id, Medicine FROM Treatment	The columns Patient_id and Medicine is unusual for Alice. The table Treatment is unusual for Alice.	Alice	15:00	192.168.3.3	100

Fig. 4. Data Leakage Audit Tool

management system used by several healthcare providers worldwide. The system was used to generate a number of data leakages. In particular, we simulated user actions that may correspond to data leakages in GNU Health. Based on these actions, we extracted a *leakage dataset* consisting of 194 queries for which an alert was raised by the DLD system. The dataset was validated by our industry partner, Roessingh Hospital in the Netherlands.

The leakage dataset was manually analyzed by a group of security experts. In particular, we developed a questionnaire describing the leakages; each leakage was described along with its key features. The security experts were invited to answer the questionnaire and evaluate the criticality of the leakages. In particular, the security experts assessed the criticality of the leakages using a three-valued scale (i.e., low, medium, high). This assessment was the underlying reference dataset used for the evaluation and comparison of the two data leakage quantification methods.

Evaluation Framework To assess the quality of the quantification results obtained using both data leakage quantification approaches, we have considered an evaluation framework consisting of a number of standard quality metrics, namely recall, false discovery rate (FDR), and F-measure [7,49]. For the evaluation, we computed recall, false discovery rate and F-measure per criticality level (i.e., Low, Medium, High), as well as for the overall leakage dataset.

Recall measures the quality of an approach in finding relevant information and is calculated as:

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

False Discovery Rate (FDR) measures the quality of an approach in terms of false positive and is calculated as:

- $FDR_{overall} = \frac{\text{\# of false assignments}}{\text{\# of alerts}}$
- $FDR_{class} = \frac{\text{\# of false assignments to a particular class}}{\text{total \# of returned assignments to a particular class}}$

	Recall				**FDR**				**F-Measure**			
	Low	**Medium**	**High**	**Overall**	**Low**	**Medium**	**High**	**Overall**	**Low**	**Medium**	**High**	**Overall**
M-Score	78.5%	50.7%	57.1%	64.9%	30.5%	35.1%	50%	35.1%	73.7%	56.9%	53.3%	64.9%
L-Severity	67.7%	76.7%	85.7%	73.7%	8.7%	34.8%	38.5%	26.3%	77.8%	70.4%	71.6%	73.7%

Table 7

Data Leakage Quantification: Recall, FDR and F-Measure

F-measure is an aggregate measure of performance which combines precision and recall into a single measure and is calculated as:

- $F\text{-}Measure_{overall} = 2\frac{Recall_{overall} \times Precision_{overall}}{Recall_{overall} + Precision_{overall}}$
- $F\text{-}Measure_{class} = 2\frac{Recall_{class} \times Precision_{class}}{Recall_{class} + Precision_{class}}$

Note that, although we compute precision for the computation of F-measure, we do not report the results for this quality metric. Indeed, precision is implicitly given by FDR, i.e. $Precision = 1 - FDR$. The choice of showing FDR instead of precision is to highlight the false positive ratio.

Results The results of the evaluation are shown in Table 7. For each approach, the table reports recall, FDR and F-measure for each criticality class as well as for the overall leakage dataset. To make a fair comparison between the two approaches we used the same sensitivity values for the data in both approaches. Moreover, we have set parameter x of M-Score to 1.

The results indicate that M-Score quantification approach is not able to properly assess the criticality of approximately 35% of the alerts with respect to the security experts' assessment. In particular, M-Score yields a low recall of approximately 55% for Medium and High criticality leakages. Interestingly, M-Score achieves the highest recall (78,5%) for the Low criticality class. M-Score's FDR is over 30% for all classes and reaches 50% for the High criticality leakages. The explanation of these results lies in the weaknesses of *RRS* and *RS* functions used by M-Score as already discussed in Section 6.2. Recall that the maximum sensitivity value of a record can be equal to 1 in M-Score; thus, leakages containing at least one highly sensitive record (e.g., HIV or a large number of leaked attributes) yield a severity value that converges to the number of leaked records. Therefore, leakages with low number of records have a low M-Score value regardless of the actual sensitivity of leaked data. Similarly, leakages with low sensitive information but large number of records have a high M-Score value. Such a fact is displayed on the relatively high FDR rates in all the criticality classes.

On the other hand, L-Severity yields an overall recall of 73.7% and an overall FDR of 26.3%, which improves M-Score of approximately 10% with respect to both metrics. The most notable improvements are noted in the assessment of the Medium and High criticality classes, where recall is 76.7% and 85.7% respectively. Our approach has a lower recall only in the Low criticality class; however, the FDR for this class drops under 10%. In addition, F-Measure shows that L-Severity generally provides higher quality assessment compared to M-Score. This can be explained by the fact that L-Severity assesses the severity of data leakages based on the actual sensitivity of leaked data.

Although our method shows promising results in the quantification of data leakages, it only uses severity to characterize the criticality of data leakages. The questionnaire provided to the security experts also aimed to identify which aspects were considered to determine the criticality of data leakages. This study has shown that other aspects, like the anomaly level of database activities, should be considered alongside severity for the quantification of data leakages. We leave the investigation of these aspects and, in particular, the definition of a security metric able to combine severity and anomaly level as future work.

8.2. Assessing Data Sensitivity

In this section we evaluate the effectiveness of the data sensitivity annotation process based on the (semantic) information available for reasoning on data sensitivity.

Experiment Setting For the evaluation of the data model we have considered a fragment of the SNOMED-CT ontology. The selected fragment consists of 100 terms, of which 54 data type nodes and 46 instance nodes. The nodes are connected with hierarchy relations. In addition, the fragment comprises 26 inference relations between instance nodes. In the experiments we have analyzed and compared three approaches to determine data sensitivity. The first approach uses all available information for the annotation of data with sensitivity labels. In particular, it uses sensitivity propagation (Definition 2) to reason over hierarchy relations as well as inference relations (Definition 3). This corresponds to the approach described in Section 4, and we refer to it as $\mathcal{DM}$. The second approach only uses sensitivity propagation, while inference relations are not used in the annotation process. We refer to this approach as $\mathcal{DH}$. The third approach does not use any relation for the annotation. Intuitively, a node has the label that has been directly assigned to it. We refer to this approach as $\mathcal{NR}$. It is worth noting that this is the approach adopted by M-Score.

To evaluate the three sensitivity calculation approaches we have considered five initial sensitivity value assignments (annotations). We have defined an initial annotation consisting of 15 nodes (A_1). This annotation includes a sensitivity value for the root node. This annotation has been progressively extended to include 30 nodes (A_2), 50 nodes (A_3) and 70 nodes (A_4). In the last annotations (A_5) all nodes have been annotated with a sensitivity value. The number of annotated nodes in an initial assignment provides an indication of the effort required by the security expert.

Evaluation Framework The aim of the evaluation is to assess the effectiveness of data sensitivity calculation approaches in relation with the effort required by the security expert to build an annotation for all data elements. To this end, we have considered three metrics, namely coverage, underestimated nodes and undefined nodes.

Coverage (M_1) measures the number of nodes with a proper sensitivity value assigned to them:

- $M_1 = \frac{\text{\# of nodes with proper sensitivity value}}{\text{\# of nodes in model}}$

Underestimated nodes (M_2) measures the number of nodes with a lower sensitivity value assigned to them:

- $M_2 = \frac{\text{\# of nodes with lower sensitivity value}}{\text{\# of nodes in model}}$

Undefined nodes (M_3) measures the number of nodes with no sensitivity value assigned.

- $M_3 = \frac{\text{\# of nodes with no sensitivity value}}{\text{\# of nodes in model}}$

Results The results of the evaluation are shown in Table 8. For each approach, the coverage (M_1), underestimated nodes (M_2) and undefined nodes (M_3) are calculated. In the table, we only report M_3 for $\mathcal{NR}$. Since the initial annotations include a sensitivity value for the root node, M_3 is always equal to 0 for $\mathcal{DM}$ and $\mathcal{DH}$.

As expected, the results show that $\mathcal{NR}$ is subject to a high undefined rate. This is due to the fact that this approach does not support any sensitivity propagation mechanisms and, thus, all nodes have to be explicitly annotated with a sensitivity value. Therefore, the coverage increases in relation to the number of nodes annotated manually. A vast improvement in the effectiveness is noted for $\mathcal{DH}$, where

	$\mathcal{DM}$		$\mathcal{DH}$		$\mathcal{NR}$		
Annotations	M_1	M_2	M_1	M_2	M_1	M_2	M_3
A_1 - 15 nodes	1	0	0.81	0.19	0.14	0.01	0.85
A_2 - 30 nodes	1	0	0.90	0.10	0.29	0.01	0.70
A_3 - 50 nodes	1	0	0.94	0.06	0.48	0.02	0.50
A_4 - 70 nodes	1	0	0.96	0.04	0.66	0.04	0.30
A_5 - 100 nodes	1	0	0.96	0.04	0.96	0.04	0

Table 8

Data Model Validation

the sensitivity of nodes is calculated using sensitivity propagation via the hierarchy relations. With 15 nodes annotated (A_1), the coverage is equal to 81%, while the sensitivity was underestimated for 19 nodes. Similarly to $\mathcal{NR}$, the coverage rate increases with the number of pre-annotated nodes. $\mathcal{DH}$ reaches maximum effectiveness with 96% coverage when 70 nodes were annotated with a sensitivity value. In this case, a small percentage (4%) of the nodes is still assigned with a lower sensitivity value. It is worth noting that this holds also in the last case where all nodes are pre-annotated with a sensitivity value. This fact denotes that, even when every node has a pre-assigned sensitivity value, errors in the estimation of the sensitivity of the data can occur.

The underestimation of sensitivity is primarily caused due to the absence of information on the relations between instance nodes, an issue that is addressed in $\mathcal{DM}$. The results in Table 8 show that $\mathcal{DM}$ outperforms the other alternatives in terms of effectiveness. In particular, even with a small number of nodes annotated (first case) the coverage reaches 100% with no underestimated values assigned. Summarizing, $\mathcal{DM}$ maximizes the coverage of the sensitivity values assigned while keeping the effort required for the annotation low.

9. Related Work

Several proposals aiming at data leakage detection and protection can be found in the literature [1,39,43]. Data leakage detection (DLD) solutions differ in the approach and technologies used to detect leakages. In particular, DLD solutions can be classified into two main streams: specification-based [17,18,21,43] and behavior-based [10,27,28,32] approaches. Specification-based approaches use predefined policies (e.g., access control policies, firewall rules, patterns) that define which operations are allowed or not. For instance, such approaches include the use of keywords, regular expressions, text classification [21], and information retrieval [17,18] to detect the presence of sensitive data leaving the organization perimeter. The major drawback of specification-based approaches is that they can only detect known leakage patterns. In behavior-based DLD solutions the permitted usage of data is defined by observing users' behavior. These approaches usually employ machine learning or statistical techniques to learn a model of normal behavior and flag every deviation from such a model as an anomaly. Although behavior-based approaches have the ability to detect unknown leakage patters, they often produce a large number of false alerts, i.e. alerts that do not correspond to an actual data leakage. In addition, most existing DLD solutions only focus on detecting leakages and do not assess their severity. Our work is complementary to DLD solutions. In particular, our approach can extend DLD solutions by providing insights into the raised alerts.

A number of proposals for the quantification of data breaches exist in the literature [8,15,16]. These proposals measure the impact of a security incident in financial terms. For instance, security incidents are quantified on the basis of the damage on the reputation of the organization and the losses on the revenue. Another approach for measuring the severity of security incidents is proposed in [3,6]; this approach evaluates privacy infringements by quantifying deviations from the intended usage of data. These solutions, however, are generic and not tailored to analyze data leakages within database environments. As a consequence, they do not propose a systematic and comprehensive approach to assess the severity of data breaches within database environments.

Data leakage quantification is studied in the field of quantitative information flow [5,40]. 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 metric based on information theory and information entropy. Quantitative information flow has also been applied to quantify leakages at network level [9]. In particular, it has been used to measure the amount of leaked information (measured in bytes) in the hypertext transfer protocol. Another approach to data leakage quantification is presented in [10]. This approach quantifies alerts of data leakages with respect to their anomaly level, allowing security analysts to focus on the most abnormal cases. The major drawback existing data leakage quantification approaches is that they do not consider the semantics of the leaked information to quantify the criticality of data leakages. In particular, the sensitivity of leaked data is not considered in the calculation of the severity of a leakage.

To the best of our knowledge, M-Score [20] is the only proposal that uses semantic information to compute the severity of data leakages. In particular, M-Score measures the severity of leakages in database environment on the basis of the amount and sensitivity of the data leaked. However, M-Score is not able to accurately distinguish data leakages. In this work, we have presented a data leakage quantification approach that considers the semantic information related to the leakage and is able to properly distinguish data leakages. Experimental results show that our approach provides a better characterization of the severity of leakages compared to M-Score.

10. Conclusion

In this work we have presented a novel approach for the quantification of data leakages with respect to their severity. The assessment of the severity of data leakages considers the amount and sensitivity of the leaked information together with the ability to identify the individuals related to the leaked information. To specify and reason on data sensitivity, we defined a data model representing the knowledge in a given domain. We validated the approach by analyzing data leakages in a typical healthcare environment.

Our data leakage quantification metric uses a distinguishing factor to determine the level of data anonymization. This factor is based on the number of occurrences of quasi identifiers in the dataset. An interesting direction for future work is to evaluate the impact of the distinguishing factor on the severity of data leakages by considering more sophisticated approaches to data anonymization like differential privacy. Moreover, the alerts generated by a DLD solution may not correspond to data misuses, i.e. alerts may turn out to be false positive. Therefore, the severity of a leakage may not correspond to its risk level (risk is usually defined as the combination of the severity and probability of an event). An interesting direction for further investigation is the integration of our approach with DLD solutions able to determine the probability that an alert is indeed a data breach. This would allow a risk-based ranking of leakages.

Acknowledgment This work has been funded by the Dutch national program COMMIT under the THeCS project.

References

[1] I. M. Abbadi and M. Alawneh. Preventing insider information leakage for enterprises. In *Proceedings of International Conference on Emerging Security Information, Systems and Technologies*, pages 99–106. IEEE, 2008.

[2] A. Adriansyah, B. F. van Dongen, and N. Zannone. Controlling break-the-glass through alignment. In *Proceedings of International Conference on Social Computing*, pages 606–611. IEEE, 2013.

[3] A. Adriansyah, B. F. van Dongen, and N. Zannone. Privacy analysis of user behavior using alignments. *it – Information Technology*, 55(6):255–260, 2013.

[4] AXLE Project. `http://axleproject.eu/`. (accessed July 27, 2015).

[5] M. Backes, B. Kopf, and A. Rybalchenko. Automatic discovery and quantification of information leaks. In *Proceedings of Symposium on Security and Privacy*, pages 141–153. IEEE, 2009.

[6] S. Banescu and N. Zannone. Measuring privacy compliance with process specifications. In *Proceedings of International Workshop on Security Measurements and Metrics*, pages 41–50. IEEE, 2011.

[7] Y. Benjamini and 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(1):289–300, 1995.

[8] B. Blakley, E. McDermott, and D. Geer. Information security is information risk management. In *Proceedings of New Security Paradigms Workshop*, pages 97–104. ACM, 2001.

[9] K. Borders and A. Prakash. Quantifying information leaks in outbound web traffic. In *Proceedings of Symposium on Security and Privacy*, pages 129–140. IEEE, 2009.

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

[11] E. Costante, F. Paci, and N. Zannone. Privacy-aware web service composition and ranking. In *Proceedings of International Conference on Web Services*, pages 131–138. IEEE, 2013.

[12] E. Costante, S. Vavilis, S. Etalle, J. den Hartog, M. Petković, and N. Zannone. Database anomalous activities - detection and quantification. In *Proceedings of the 10th International Conference on Security and Cryptography*, pages 603–608. SciTePress, 2013.

[13] Data Segmentation for Privacy (DS4P) initiative. `http://wiki.siframework.org/Data+Segmentation+for+Privacy+Homepage`. (accessed July 27, 2015).

[14] C. Doulaverakis, G. Nikolaidis, A. Kleontas, I. Kompatsiaris, et al. GalenOWL: Ontology based drug recommendations discovery. *J. Biomedical Semantics*, 3:14, 2012.

[15] F. Farahmand, S. B. Navathe, P. H. Enslow, and G. P. Sharp. Managing vulnerabilities of information systems to security incidents. In *Proceedings of International Conference on Electronic Commerce*, pages 348–354. ACM, 2003.

[16] A. Garg, J. Curtis, and H. Halper. Quantifying the financial impact of it security breaches. *Information Management & Computer Security*, 11(2):74–83, 2003.

[17] E. Gessiou, Q. H. Vu, and S. Ioannidis. IRILD: an Information Retrieval based method for Information Leak Detection. In *Proceedings of European Conference on Computer Network Defense*, pages 33–40. IEEE, 2011.

[18] J. Gómez-Hidalgo, J. Martın-Abreu, J. Nieves, I. Santos, F. Brezo, and P. Bringas. Data leak prevention through named entity recognition. In *Proceedings of International Conference on Social Computing*, pages 1129–1134. IEEE, 2010.

[19] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies*, 43(5):907–928, 1995.

[20] A. Harel, A. Shabtai, L. Rokach, and Y. Elovici. M-score: A misuseability weight measure. *IEEE Transactions on Dependable and Secure Computing*, 9(3):414–428, 2012.

[21] M. Hart, P. Manadhata, and R. Johnson. Text classification for data loss prevention. In *Privacy Enhancing Technologies*, LNCS 6794, pages 18–37. Springer, 2011.

[22] Health Level Seven (HL7). Healthcare Privacy and Security Classification System (HCS), Release 1, 2013.

[23] HL7 Fast Health Interoperable Resources (FHIR). `http://hl7.org/fhir`. (accessed July 27, 2015).

[24] HL7 Security and Privacy Ontology Project). `http://wiki.hl7.org/index.php?title=Security_and_Privacy_Ontology`. (accessed July 27, 2015).

[25] Information Age. New EU data laws to include 24hr breach notification, 2012.

[26] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Trans. Database Syst.*, 26(2):214–260, 2001.

[27] A. Kamra, E. Terzi, and E. Bertino. Detecting anomalous access patterns in relational databases. *VLDB Journal*, 17(5):1063–1077, 2007.

[28] R. Koch. Towards next-generation intrusion detection. In *Proceedings of International Conference on Cyber Conflicts*, pages 1–18. IEEE, 2011.

[29] N. Li, T. Li, and S. Venkatasubramanian. t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In *Proceedings of the International Conference on Data Engineering*, pages 106–115. IEEE, 2007.

[30] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy Beyond K-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), 2007.

[31] F. Massacci, J. Mylopoulos, and N. Zannone. Hierarchical hippocratic databases with minimal disclosure for virtual organizations. *VLDB Journal*, 15(4):370–387, 2006.

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

[33] I. Matteucci, P. Mori, and M. Petrocchi. Prioritized Execution of Privacy Policies. In *Data Privacy Management and Autonomous Spontaneous Security*, LNCS 7731, pages 133–145. Springer, 2012.

[34] Open Clinical: Ontologies. `http://www.openclinical.org/ontologies.html`. (accessed January 12, 2015).

[35] OpenGALEN. `http://www.opengalen.org/`. (accessed January 12, 2015).

[36] F. Paci and N. Zannone. Preventing information inference in access control. In *Proceedings of Symposium on Access Control Models and Technologies*, pages 87–97. ACM, 2015.

[37] Ponemon Institute. Third annual benchmark study on patient privacy & data security. Research Report, 2012.

[38] Ponemon Institute. 2014 Cost of Data Breach Study: Global Analysis. Research Report, 2014.

[39] M. B. Salem, S. Hershkop, and S. J. Stolfo. A survey of insider attack detection research. In *Insider Attack and Cyber Security*, Adv. Inf. Secur. 39, pages 69–90. Springer, 2008.

[40] G. Smith. On the foundations of quantitative information flow. In *Foundations of Software Science and Computation Structures*, LNCS 5504, pages 288–302. Springer, 2009.

[41] SNOMED - CT. `http://www.ihtsdo.org/snomed-ct/`. (accessed January 12, 2015).

[42] L. Sweeney. K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, 2002.

[43] T. Takebayashi, H. Tsuda, T. Hasebe, and R. Masuoka. Data loss prevention technologies. *Fujitsu Scientific and Technical Journal*, 46(1):47–55, 2010.

[44] Test Level 7. `http://tl7.intelliware.ca`. (accessed July 27, 2015).

[45] The Gene ontology. `http://www.geneontology.org/`. (accessed January 12, 2015).

[46] The Open Biological and Biomedical Ontologies Foundry. `http://www.obofoundry.org/`. (accessed January 12, 2015).

[47] S. Vavilis, A. Egner, M. Petković, and N. Zannone. An anomaly analysis framework for database systems. *Computers & Security*, 53:156–173, 2015.

[48] S. Vavilis, M. Petković, and N. Zannone. Data Leakage Quantification. In *Data and Applications Security and Privacy XXVIII*, LNCS 8566, pages 98–113. Springer, 2014.

[49] Y. Yang. An evaluation of statistical approaches to text categorization. *Inf. Retr.*, 1(1-2):69–90, 1999.