The Monetary Value of Information: A Leakage-Resistant Data Valuation

Stefan Barthel, Eike Schallehn
Institute of Technical and Business Information Systems,
Otto-von-Guericke-University Magdeburg, Germany,
{stefan.barthel, eike.schallehn}@ovgu.de

Abstract: The importance of information as a main asset of a company or organization is widely acknowledged nowadays. The loss of or the unauthorized access to sensitive information are critical and can possibly send a company into bankruptcy. Furthermore, the risk of information larceny is most often not caused by a direct attack of unauthorized outsiders, but by authorized extractions by malicious or unaware insiders passing data to unauthorized outsiders. Unfortunately, this problem cannot be solved by the typically used role-based authentication. The detection of malicious accesses based on typical access characteristics, which has inspired some research, is limited in its potential. Therefore, we present a conceptual approach based on the valuation of information, i.e., using a description of the actual worth of data items within database systems. This allows to rate potential losses on the fly as well as preventing valuable extractions done by insiders. In detail, we describe a mechanism called leakage-resistant data valuation that calculates a monetary value for every query and takes according action if the cumulated monetary value exceeds a threshold (per query or per time span).

1 Introduction

In current, worldwide distributed information systems, sensitive data needs to be stored securely and well protected to prevent malicious use [VBF^{+}04]. One example is the misuse of bank account information, which we will focus on to illustrate our proposal. In the area of financial institutes, highly sensitive information are stored compactly. Authorized users are querying sensitive data every day, while they enjoy an enormous trust from the management. Other fields with highly sensitive data include health care, law, government, industrial research, and military applications. Because sensitive data are so valuable, several organizations, individuals, and even governments are interested in data extracts, in some cases regardless whether they are obtained with legal or illegal methods. For example, countries that assume tax evasion by their inhabitants in countries with high bank secrecy, are interested in getting additional evidence of tax evaders, because it can lead to millions of additional tax income. Therefore, buying stolen data is feasible, even it is prohibited in the country of origin. In the case of Germany, which bought several extracts of tax evaders in the recent past, the stolen data was almost always sold on a CD, probably extracted and copied by a member of staff or someone who has access to the system. This scenario is generally referred to as the *insider threat*. It is generally accepted that the cost of damages caused by insider threats exceeds those of outsider threats [HSRE10]. That means, even if several security restrictions and authorization based access controls are active, at least one person has the ability to access all the data – in most cases the system administrator. Therefore, from our point of view, the conventional way of access authorization has to be thoroughly rethought for sensitive data.

In this paper, we introduce our approach based on an actual monetary value of the data any user is allowed to work with. We choose the mapping of monetary value and information, because it is intuitively accessible when speaking about a certain valuation of information. By specifying two thresholds and a monetary value for every attribute, the system is enabled to easily react in two ways:

- alerting and logging, if single queries or cumulated accesses of one user exceed a certain threshold, or
- truncating the result set, i.e., limiting the output of sensitive data, if a further and more rigid threshold is exceeded.

For example, a bank clerk will be able to access personal and bank information of individual customers, but as there is no need to extract many records at once or in a short period of time, this can be restricted by our approach. Nevertheless, the database management system will be able to backup data and have the ability to read and write data according to fixed application purposes.

2 Preliminary Considerations

Security mechanisms for conventional databases are divided into three layers: physical security, operating system security, and database management system security [BS05]. Because the database management system (DBMS) is the primary line of defense in constraining access to the data stored in a database, we focus on access control models of the DBMS. Established access control models can be grouped into three classes: Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role Based Access Control (RBAC) [SS94]. They are widely used, well investigated, and have been proven to be useful, as long as the data is accessed using mechanisms of the DBMS. Especially RBAC is an effective way to scale access control in large systems [PG06]. However, access control is useless if an attacker gains direct access to the database or at least the hard disk drive. An intruder could mine a database footprint on disk and simply copy the plain text. Encryption can solve those issues and protect the data against unauthorized, physical reads and writes [BG09]. In addition, encryption can solve the problem of data backup theft [SVEG10].

Nevertheless, access control as well as encryption are not restrictive and protective enough to guarantee that sensitive data can not be queried if an insider or an outsider, who gained access rights, extracts data. This threat is most often caused by current or former contractors, employees, or even business partners who have or had authorized access to an organization's network and want to take advantage of it [GSB+04]. To ensure a maximum of protection – even if a total protection will never exist – this insider threat should be reduced to a minimum. We will illustrate that the protection level can be significantly improved by introducing the relatively light-weight concept of a leakage-resistant data valuation.

3 Leakage-Resistant Data Valuation

Our proposed approach is a valuation-based extension of the conventional access control for the relational database model, which calculates for every query the total monetary value depending on queried tables and attributes as well as used operations. In detail, every attribute $A_i \in R$ of a base relation schema R is valuated by a certain monetary value ($mval(A_i) \in \mathbb{R}$). With these attribute valuations, we derive a monetary value for one tuple $t \in r(R)$ given by Equation (1), as well as the total monetary value of the relation $r(R)$ given by Equation (2), if data is extracted by a query.

$$mval(t \in r(R)) = \sum_{A_i \in R} mval(A_i) \tag{1}$$

$$mval(r(R)) = \sum_{t \in r(R)} mval(t) = |r(R)| * mval(t \in r(R)) \tag{2}$$

We are considering (1) the quality of information (monetary value of attributes) as well as (2) quantity of information (number of tuples). Based on this, query results derived by operations need to be considered and valuated depending on type and granularity. For these purposes we decide, that operations which enhance the information content should increase (e.g., joins), whereas coarsening operations (e.g., aggregation), which reduce the number of rows or columns, should decrease the total monetary value of the resulting data. However, the calculated monetary value of one tuple is less than of a tuple with the same number of attributes, but at least as of one aggregated value, because the information content increases due to aggregation. For the sake of simplicity, we propose to multiply the monetary value of an aggregated attribute by a certain factor (e.g., 2), regardless of the type of aggregation (e.g., AVG, SUM, MAX, etc.). In case of a joined tuple of several tables, all monetary values of included attributes are summarized. The cumulated total monetary value of a query is afterwards compared to two thresholds.

Suspicious threshold: $thr_{susp} \in \mathbb{R}$ is used to identify queries with a monetary value which is suspicious, i.e., could indicate malicious behavior, and these are written to an alert log and handled accordingly.

Truncate threshold: $thr_{trun} \in \mathbb{R}$ is used to identify malicious queries directly. Queries with a higher total monetary value are truncated when exceeding this boundary. In addition, truncated queries also need to be logged, therefore the truncate threshold is set to a higher value than the suspicious threshold ($thr_{susp} < thr_{trun}$).

Alert log: The *alert log* accordingly captures every exceedance and is written continuously to allow a tracing of violations over time. A maintenance job periodically analyses the captured discrepancies and informs responsible security guards – a number of persons that are delegated (at least four-eye-principle).

To prevent valuable extractions by one query and several queries within a short period of time, we do not only (1) truncate or log a single query that extracts to much information, but also (2) prevent a sequence of queries with the same intention. Hence, we cumulate the monetary values of previous non-suspicious queries ($mval(r(R_i)) < thr_{susp}$) per user, to measure how much information was extracted over a period of time. Because we are just interested in a certain time interval, the user valuation is reset periodically (e.g., hourly, daily, weekly, etc.). The mechanism of logging and truncating when reaching thresholds is also used for the time restricted user valuation. With this determination we are facing line by line extractions, e.g., by a batch script. Additionally, we recommend that the calculation of the total monetary value of a query is done before physically querying the result set. Due to the fact that monetary values of attributes and tuples as well as both thresholds are known, by pre-calculating, there is no overhead of joining tuples, which will not be displayed anyway. To do so, we calculate the monetary value of one result tuple including every attribute (shown in Eq. (1)) and we derive the maximum releasable tuples ($tpl_{max} \in \mathbb{N}$) for a certain result relation $r(R)$:

$$tpl_{max}(r(R)) = \left\lceil \frac{thr_{trun}}{mval(t \in r(R))} \right\rceil \tag{3}$$

To inform the requester of a query, that the result set was shortened because the truncate threshold was exceeded, an information is displayed. The truncation only applies to commands which create displayed results. Others, like backup processes, data modifications, etc., could for instance be privileged by a whitelist. We see our approach as an extended access control mechanism and classify it as active access control regarding to [PG06], because it provides continuous and dynamic access control beyond initial access control decisions. For implementation purposes, monetary values for distinct attribute may differ and need to be set manually while defining the table schema. Moreover, monetary values of tuples are derived from attributes and do not need to be set at all. The suspicious threshold as well as the truncate threshold are disabled and set by default ($thr_{susp} = thr_{trun} = \infty$), but can be easily activated.

To be able to define monetary values for attributes and thresholds, we propose an extension of SQL by adding thresholds as system variables and monetary values of attributes as extra options within the table definition. An example of syntax to define several monetary values for attributes while creating a table and enabling both thresholds are shown below:

```
suspicious_valuation=2000
truncate_valuation=4000

CREATE TABLE table_1
(
  attribute_1 INT PRIMARY KEY MVAL 0.1,
  attribute_2 UNIQUE COMMENT 'important' MVAL 10,
  attribute_3 DATE
);
```

Within this code example attribute 1 and 2 are directly set by definition, while attribute number 3 is set to the default value ($mval(A_3) = 0$) by the system. With this determination, we can ensure, that a DBMS with an implemented leakage-resistant data valuation, but without any set valuations, will exactly work like a conventional, non-modified DBMS. Consequently, it is also possible to modify those valuations by using an ALTER-command of SQL. All data valuations are stored in the data dictionary, because it is a centralized repository of information about data such as meaning, relationships to other data, origin, usage, and format [IBM93]. However, at runtime all valuations should be held in main memory to ensure a maximum of performance.

4 Related Work

Conventional database management systems mostly use access control models to face unauthorized access on data. However, these are insufficient when an authorized individual extracts data regardless whether she is the owner or has stolen that account. Several methods were conceived to eliminate those weaknesses. We refer to Park and Giordano [PG06], who give an overview of requirements needed to address the insider threat.

Authorization views partially achieve those crucial goals of an extended access control and have been proposed several times. For example, Rizvi et al. [RMSR04] as well as Rosenthal et al. [RS00] use authorization-transparent views. In detail, incoming user queries are only admitted, if they can be answered using information contained in authorization views. Contrary to this, we do not prohibit a query in its entirety. Another approach based on views was introduced by Motro [Mot89]. Motro handles only conjunctive queries and answers a query only with a part of the result set, but without any indication why it is partial. We do handle information enhancing (e.g., joins), as well as coarsening operations (e.g., aggregation) and we do display a user notification. All authorization view approaches require an explicit definition of a view for each possible access need, which also imposes the burden of knowing and directly querying these views. In contrast, the monetary values of attributes are set while defining the tables and the user can query the tables or views she is used to. Comparison methods are also not addressing the problem of how to keep up to date, because new operators and constraint constructs are repeatedly evolving. Moreover, the equivalence test of general relational queries is undecidable and equivalence for conjunctive queries is known to be NP complete [CM77]. Therefore, the leakage-resistant data valuation is more applicable, because it does not have to face those challenges.

However, none of these methods does consider the sensitivity level of data that is extracted by an authorized user. In the field of *privacy-preserving data publishing* (PPDP), on the contrary, several methods are provided for publishing useful information while preserving data privacy. In detail, multiple security-related measures (e.g., k-anonymity [Swe02], l-Diversity [MKGV07]) have been proposed, which aggregate information within a data extract in a way that they can not lead to an identification of a single individual. We refer to Fung et al. [FWCY10], who give a detailed overview of recent developments in methods and tools of PPDP. However, these mechanisms are mainly used for privacy-preserving tasks and are not in use when an insider accesses data. Moreover, privacy preserving mechanisms are not applicable for our scenario, because they do not consider a line by line extraction over time as well as the information loss by aggregating attributes of the result set, that the user is allowed to see.

To the best of our knowledge, there is only the approach of Harel et al. ([HSRE10], [HSRE11], [HSRE12]) that is comparable to our data valuation to prevent suspicious, authorized data extractions. Harel et al. introduce the *Misuseability Weight (M-score)* that describes the sensitivity level of the data exposed to the user. Hence, Harel et al. focus on the protection of the quality of information, whereas our approach predominantly preserves the extraction of a collection of data (quantity of information). Harel et al. also do not consider extractions over time, logging of malicious requester and the backup process. In addition, mapping attributes to a certain monetary value is much more applicable and intuitional, than mapping to a artificial M-score. We also suppose a better performance for our approach, because we can calculate the maximal size of a result set before physically querying.

Our extended authorization control does not limit the system to a simple query-authorization control without any protection against the insider threat, rather we allow a query to be executed whenever the information carried by the query is legitimate according to the specified authorizations and thresholds.

5 Conclusion and Future Work

We presented the leakage-resistant data valuation as an additional layer in a security defense model for raw data. Within this security defense model each layer functions as a separate firewall and all of the layers are ordered like onion skins with the raw data as base (see Fig. 1). This principle is called defense in depth, because attackers must get through layer after layer of defense to reach the base – the raw data. The data valuation layer is inserted below the conventional access control

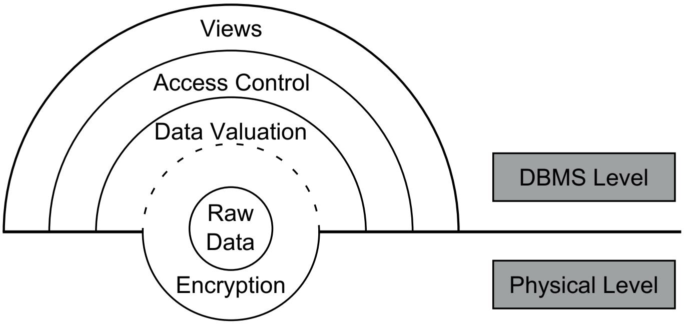

Figure 1: Security defense model on DBMS and physical level

layer and therefore more powerful and restrictive. On the top of access control methods, it is common practice to define views which give users access to a specific portion of data without having direct access rights. Encryption on the contrary, is situated below our data valuation layer, because it uses DBMS-, file-, application- or client side encryption to protect the raw data [SVEG10]. In future work we will implement our leakage-resistant data valuation approach and evaluate performance as well as usability issues. There will also be research on modifying our quantity-cutting mechanism to a quality-cutting one, where data will be masked instead of truncated. Furthermore, we will publish a more detailed description of how to treat various operations (e.g., different joins, set operation, etc.), procedures, and functions. It needs to be investigated how to set the monetary values and thresholds effectively, regarding to various usage scenarios. Another field of application is a self-protecting, leakage-resistant data valuation, where the system will set various parameters by itself, collect user profiles, and compare them to a malicious behavior that was identified earlier. With this extension, we will also be able to establish an anomaly detection or at least enhance an existing anomaly detection system.

Acknowledgments: This research has been funded in part by the German Federal Ministry of Education and Science (BMBF) through the Research Program under Contract FKZ: 13N10817. We thank Ingolf Geist for his support.

References

[BG09] Luc Bouganim and Yanli Guo. Database Encryption. In *Encyclopedia of Cryptography and Security*, pages 1–9. Springer, 2009.

[BS05] Elisa Bertino and Ravi Sandhu. Database Security - Concepts, Approaches, and Challenges. *IEEE Dependable and Secure Comp.*, 2(1):2 –19, March 2005.

[CM77] Ashok K. Chandra and Philip M. Merlin. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In *Proceedings of the ninth annual ACM symposium on Theory of computing*, STOC'77, pages 77–90. ACM, 1977.

[FWCY10] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-Preserving Data Publishing: A Survey of Recent Developments. *ACM Comput. Surv.*, 42(4):14:1–14:53, June 2010.

[GSB$^+$04] Richard Guida, Robert Stahl, Thomas Bunt, Gary Secrest, and Joseph Moorcones. Deploying and Using Public Key Technology: Lessons Learned in Real Life. *IEEE Security Privacy*, 2(4):67 – 71, August 2004.

[HSRE10] Amir Harel, Asaf Shabtai, Lior Rokach, and Yuval Elovici. M-score: Estimating the Potential Damage of Data Leakage Incident by Assigning Misuseability Weight. In *Proceedings of the 2010 ACM workshop on Insider threats*, Insider Threats'10, pages 13–20. ACM, 2010.

[HSRE11] Amir Harel, Asaf Shabtai, Lior Rokach, and Yuval Elovici. Eliciting Domain Expert Misuseability Conceptions. In *Proceedings of the sixth international conference on Knowledge capture*, K-CAP'11, pages 193–194. ACM, 2011.

[HSRE12] Amir Harel, Asaf Shabtai, Lior Rokach, and Yuval Elovici. M-Score: A Misuseability Weight Measure. *IEEE Trans. Dependable Secur. Comput.*, 9(3):414–428, May 2012.

[IBM93] Corporation IBM. *IBM Dictionary of Computing*. McGraw-Hill, Inc., New York, NY, USA, 10th edition, 1993.

[MKGV07] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-Diversity: Privacy Beyond k-Anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1):1–50, March 2007.

[Mot89] Amihai Motro. An Access Authorization Model for Relational Databases Based on Algebraic Manipulation of View Definitions. In *Proceedings of the Fifth International Conference on Data Engineering*, pages 339–347. IEEE Computer Society, 1989.

[PG06] Joon S. Park and Joseph Giordano. Access Control Requirements for Preventing Insider Threats. In *Proceedings of the 4th IEEE international conference on Intelligence and Security Informatics*, ISI'06, pages 529–534. Springer, 2006.

[RMSR04] Shariq Rizvi, Alberto Mendelzon, S. Sudarshan, and Prasan Roy. Extending Query Rewriting Techniques for Fine-Grained Access Control. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD'04, pages 551–562. ACM, 2004.

[RS00] Arnon Rosenthal and Edward Sciore. View Security as the Basis for Data Warehouse Security. In *CAiSE Workshop on Design and Management of Data Warehouses*, DMDW'2000, pages 5–6. CEUR-WS, 2000.

[SS94] Ravi S. Sandhu and Pierangela Samarati. Access Control: Principle and Practice. *IEEE Communications Magazine*, 32(9):40–48, September 1994.

[SVEG10] Erez Shmueli, Ronen Vaisenberg, Yuval Elovici, and Chanan Glezer. Database Encryption: An Overview of Contemporary Challenges and Design Considerations. *SIGMOD Rec.*, 38(3):29–34, December 2010.

[Swe02] Latanya Sweeney. k-Anonymity: A Model For Protecting Privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, October 2002.

[VBF$^+$04] Vassilios S. Verykios, Elisa Bertino, Igor N. Fovino, Loredana P. Provenza, Yucel Saygin, and Yannis Theodoridis. State-of-the-Art in Privacy Preserving Data Mining. *SIGMOD Rec.*, 33(1):50–57, March 2004.