Dynamic Sensitivity-Based Access Control

Amir Harel, Asaf Shabtai, Lior Rokach, Yuval Elovici
Deutsche Telekom Laboratories at Ben-Gurion University,
Department of Information Systems Engineering,
Ben Gurion University of the Negev, Beer-Sheva, Israel
{harelam, shabtaia, liorrk, elovici}@bgu.ac.il

***Abstract*— In this paper we propose a new access control mechanism, *Dynamic Sensitivity-Based Access Control* (*DSBAC*), designed to regulate users' access to sensitive data stored in relational databases. The DSBAC is an extension of the basic mandatory access control (MAC) mechanism, and it uses the *M*-score (*M*isuseability score) measure in order to assign, dynamically, an access class to each set of tuples.**

Keywords- M-score, access control, data leakage

I. INTRODUCTION

Organizational database systems retain a vast amount of private and sensitive data (e.g., customer records, medical details and business intellectual property). This data is one of the organization's most valuable assets, thus exposing it to unauthorized entities might lead to severe financial damage or compromise the privacy of its customers. Preventing data leakage and data misuse is, therefore, essential.

In recent years, several methods have been proposed for mitigating data leakage and data misuse in database systems. These methods can generally be classified as *detective* or *preventive*. Detective solutions attempt to detect the leakage/misuse while it occurs. For example, previous works used anomaly detection in order to implement the detective approach ([1]-[3]). This is done by comparing a set of features representing the current user action (e.g., accessed tables, statistics extracted from the result-sets, etc.) to the user's profile. A significant deviation from the normal behavior indicates a possible leakage/misuse.

The preventive approach focuses on preventing the leakage/misuse from occurring, mostly by applying encryption and access control mechanisms. Data access control mechanisms are generally categorized as *discretionary access control* (DAC), *role-based access control* (RBAC) or *mandatory access control* (MAC). In discretionary access control, the data owner decides which users can access the data and what privileges they have. In the role-based access control different users are classified as belonging to one or more roles according to their job description. Access to data objects are then granted according to the user's role. The mandatory access control mechanism regulates user access to data according to predefined classifications of both the user (the *subject*) and the data (the *object*). The classification is based on partially ordered access classes (e.g., top secret, secret, confidential, unclassified).

Previous works extended the access control mechanisms to enable the definition of fine-grained access control rules. For example, by generalizing the RBAC mechanism and also consider behavioral attributes of the user, such as the time she normally comes to work or the customers she usually interacts with, in addition to the basic job description (role) [4]; or, by predicting the ability of the user to infer sensitive data based on data that the user already have [5].

However, none of the proposed methods consider the sensitivity level of the data to which the user may be exposed. Consequently, in Harel et al. [6] we presented the *M*-score (*M*isuseability score) measure. The *M*-score measure estimates the potential damage to the organization, in case the data is leaked or misused, by measuring the amount and sensitivity of the data that was exposed to the user.

In this paper we propose a new access control mechanism, *Dynamic Sensitivity-Based Access Control* (*DSBAC*), to regulate users' access to sensitive data stored in relational databases. The DSBAC is an extension of the basic mandatory access control (MAC) mechanism, and it uses the *M*-score measure in order to dynamically set an access class to a given set of tuples.

II. THE *M*-SCORE MEASURE

The *M*-score is a measure used for estimating the extent of damage a user can cause an organization using the data she encounters in the course of her work [6]. This is done by ranking the sensitivity level of the data to which the user is exposed to. Using this information, the organization can then take appropriate steps to prevent this damage or to minimize its impact. In [7] we show that the *M*-score fulfill its goal of assigning a sensitivity score to tables of data that is not only accurately but also consistent with domain expert intentions.

The *M*-score measure is tailored for tabular datasets (i.e., result-sets of relational database queries), and aimed at assigning a sensitivity score to a given set of tuples. It incorporates the following three factors -

- *Quality of the information*- the importance of the information to the organization.
- *Quantity of the information*- how much information is exposed.
- *Distinguishing factor*- the amount of effort required in order to identify the specific entities in the tuples.

In order to calculate the measure, three, non-intersecting, types of attributes are defined: *identifier* (or quasi-identifier) attributes; *sensitive* attributes; and *other* attributes, which are of no importance to our discussion. The *M*-score measure is derived using the following formula:

$$\boldsymbol{MScore} = r^{1/x} \times \max_{0 \le i \le r} \left(\frac{RRS_i}{D_i} \right) = r^{1/x} \times RS$$

where,

- r - the number of tuples in the published set, representing the quantity factor of the *M*-score;
- RRS_i (*Raw Record Score*) - the sensitivity rank of the tuple *i*. This rank is calculated based on a sensitivity-score function that is defined according to the domain expert knowledge. In [7] we have proposed and evaluated different methods to acquire this function and proved their efficiency. The RRS_i component represents the quality factor;
- D_i - Tuple *i* distinguishing factor, that is calculated by counting the number of entities with identical identifiers that exist in the organization's database;
- RS (*Records Score*) - the maximal value of RRS_i/D_i; and
- x - a settable parameter that defines the quality vs. quantity tradeoff. The domain expert needs to define how important is the published set size (*r*, which is an unbounded positive integer), compared to the sensitivity of the data in it (*RS*, which is a real number in the range of 0 to 1).

To demonstrate the calculation of the *M*-score, we use Table 1, which presents a published set of tuples containing customers' data. Each tuple contains identifier attribute (customer name) and sensitive attribute (account type).

TABLE I. A PUBLICATION CONTAINING CUSTOMER RECORDS

Customer Name	Account Type
Anton Richter	Bronze
Otto Hecht	Gold

In this example, we assume that the sensitivity score functions of AccountType[Gold] and AcountType[Bronze] are 0.8 and 0.3 respectively. In addition, the company database contains only one customer with the name "Anton Richter", but 300 different customers with the name "Otto Hecht". Assuming we use x=1 (i.e., every tuple leaking is highly sensitive) then the *M*-score measure of the publication is calculated as follows:

$$\boldsymbol{MScore(Table\ 1)} = 2 \times \max_{0 \le i \le 2} \left(\frac{0.3}{1}, \frac{0.8}{300} \right) = 0.6$$

III. FROM MAC TO DYNAMIC SENSITIVITY-BASED ACCESS CONTROL (DSBAC)

In this paper, we propose using the *M*-score as the basis for a new mandatory access control mechanism for relational databases. Basic MAC implementations for relational databases partition the set of tuples in the database to sub-sets, where each sub-set holds all tuples with the same access class. According to the proposed method, the *M*-score measure is used for assigning, dynamically, an "access class" to a given set of tuples.

The DSBAC is enforced as follows: Each user is assigned with a "sensitivity clearance"; i.e., the maximal *M*-score that the subject is eligible to access. Then, for each query that a user submits, the *M*-score of the returned result-set is calculated. The derived *M*-score, which represents the *dynamic access class* of that result-set, is compared with the sensitivity clearance of the subject, in order to decide whether she is entitled to access this data. Note that the DSBAC, as the MAC does, can be enforced in addition to existing access control layers such as RBAC or DAC.

This approach presents several advantages over the basic MAC mechanism. First, as opposed to the finite number of access classes in MAC, in DSBAC there can be an infinite number of dynamic access classes, allowing more flexibility and fine-grained access control enforcement. Second, while manual labeling of tuples is required in MAC, in DSBAC, once the sensitivity score function is acquired (as presented in [7]), every result-set can be labeled automatically. Third, the dynamic approach enables the access control mechanism to derive a context-based access label; for example, the amount of tuples that were exposed, or the data that the subject already possess. Last, while in the basic MAC, subjects are only permitted to write to objects with access class higher or equal to their own (to prevent exposure of data to unauthorized subjects), in DSBAC the access class is assigned dynamically and therefore subjects are not limited in writing.

The proposed DSBAC mechanism can operate in one of the following modes: *binary* or s*ubset disclosure*. In the *binary mode*, if the subject's sensitivity clearance is lower than the result-set *M*-score, no data will be presented at all. In the *subset disclosure mode*, a subset of the result-set might be presented. The subset of tuples can be selected using an algorithm that removes parts of the set (i.e., tuples or attributes) until the *M*-score of the subset is lower than or equal to the subject's sensitivity clearance. For example, the *Remove Most Sensitive* heuristic, presented in Fig. 1, relies on the fact that the *M*-score is highly effected by the tuple with the maximal *RS*. Therefore, it iteratively removes the most sensitive tuple (i.e., the tuple with the highest *RS*) and by that reduces the *M*-score of the remaining subset of tuples. This heuristic also counts the number of tuples that were removed and returns it to the user.

```
Remove Most Sensitive (result-set T, sensitivity clearance SC)
1.  START
2.      removed ← 0
3.      WHILE M-score(T) > SC
4.          t ← select tuple with maximal RS value in T
5.          remove t from T
6.          removed ← removed + 1
7.      END WHILE
8.      return <T , removed>
9.  END
```

Figure 1. The *Remove Most Sensitive* heuristic

IV. EXTENDING THE *M*-SCORE: MULTIPLE PUBLICATIONS

So far, we have used the *M*-score as a sensitivity measure of a single publication. However, in order to avoid detection while obtaining a large amount of sensitive data, a user can try to get the data piece by piece, small portions at a time. In this section we extend the *M*-score measure to consider a set of publications. By doing so the DSBAC would be able to prevent

access to data that, together with data the user already has, violates her sensitivity clearance. We focus on the case where the user can uniquely identify each entity (e.g., customer) in the result-set; i.e., the distinguishing factor equals to 1 (D_i=1). We leave the case of publications with quasi-identifiers (where D_i>1) to future work.

Fig. 2 depicts nine optional cases resulting from two fully identifiable sequential publications. Each case is determined by the relation (equal, overlapping or distinct) between the two publications with respect to the publications' sensitive attributes (marked in shades of green) and the exposed entities which are the distinct identifiers values (marked in red). For example, in case 1 on Fig. 2, the publications share the same schema (i.e., holds the same attributes in all tuples), but have no common entities, while case 6 presents two publications that share some of the entities, but each publication holds different attributes on them.

Based on these nine possible cases we introduce the *Construct Publication Ensemble* procedure (Fig. 3) that constructs an ensemble set E on which the M-score should be calculated, where $<T_1, \ldots, T_{n-1}>$ are the previous publications, T_n is the current (new) publication and F is the time frame in which we still consider previous publication. By calculating the M-score of the ensemble set E, we actually consider the relevant prior knowledge the user has so far.

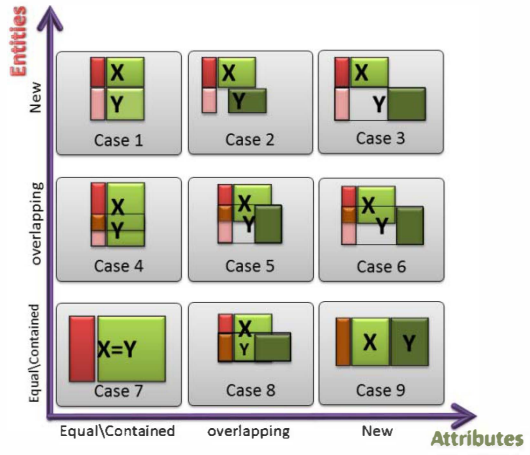

Figure 2. Nine cases resulting from two fully identifiable publications – X and Y

The *Construct Publication Ensemble* procedure is recursive. For each new publication the procedure first creates an ensemble set X of all the previous publications that are within the time frame F (lines 5 to 7). Then, the procedure checks which case in Fig. 2 fits to the current publications and acts accordingly (lines 8 to 16). Finally, on line 17 the resulted ensemble set is returned.

V. Summary and Future Work

In this paper we present a new dynamic mandatory access control mechanism which regulates users' access to sensitive data stored in relational databases. The proposed mechanism assigns, dynamically, an access class to each result-set the user requests by calculating the M-score of that result-set. The access is granted only if the sensitivity clearance of the user is higher than the dynamic access class of the result-set. Two modes of operation were presented– the binary mode, in which no data is presented in case the user's sensitivity clearance is lower than the access class; and the subset disclosure mode, in which the user can be exposed to part of the data that she does have privileges to see. Last, we present an extension of the M-score that allows it to consider previous publications when ranking the current data sensitivity. In future work, we intent to implement and further evaluate the DSBAC mechanism; and extend the M-score measure to consider additional types of prior knowledge, such as publications with D_i>1.

Construct Publication Ensemble **(<T_1, … , T_{n-1}>, T_n, F)**

1. **START**
2. **IF** $n = 1$ [1]
3. **THEN** return T_n
4. **ELSE**
5. ***RemoveOldPublications***(<T_1, … , T_{n-1}>,F) [2]
6. X ← ***ConstructPublicationEnsemble***(<T_1, … , T_{n-2}>, T_{n-1},F)
7. Y ← T_n
8. **IF** X∩Y = Y **(case 7 in Fig. 2)**
9. **THEN** E ← Y
10. **ELSE**
11. **IF** Entities(X) ∩ Entities(Y) = ∅ [3] **(cases 1-3 in Fig. 2)**
12. **THEN** E ← X ∪ Y
13. **ELSE**
14. **IF** SensAttr(X) = SensAttr(Y) [4] **(case 4 in Fig. 2)**
15. **THEN** E ← X ∪ Y-X
16. **ELSE** E ← X JOIN Y **(cases 5,6,8 and 9 in Fig. 2)**
17. return E
18. **END**

(1) No previous publications exist.
(2) Removes previous publications that are out of the time frame and shouldn't be considered
(3) Entities(X) are all the *values* of the identifiers attributes in set X.
(4) SensAttr (X) are all the *sensitive attributes* of set X.

Figure 3. The *Construct Publication Ensemble* procedure

[1] E. Bertino and E. Terzi, "Intrusion detection in RBAC-administered databases," *in: Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC 05)*, 2005, pp. 170–182.

[2] A. Kamra, E. Terzi, and E. Bertino, "Detecting anomalous access patterns in relational databases," *VLDB J.*, vol. 17, no. 5, pp. 1063–1077, 2008.

[3] S. Mathew, M. Petropoulos, H. Q. Ngo, and S. Upadhyaya,"A data-centric approach to insider attack detection in database systems," in *Proceedings of the 13th international conference on Recent advances in intrusion detection* (RAID'10), Berlin, Heidelberg: Springer-Verlag, 2010, pp. 382–401.

[4] M. Bishop and C. Gates, "Defining the insider threat," in *Proceedings of the 4th annual workshop on Cyber security and information intelligence research* (*CSIIRW '08).* New York, NY, USA: ACM, 2008, pp. 1–3.

[5] Q. Yaseen and B. Panda, "Knowledge acquisition and insider threat prediction in relational database systems," in *IEEE International Conference on Computational Science and Engineering*, vol. 3, pp. 450–455, 2009.

[6] A. Harel, A. Shabtai, L. Rokach, and Y. 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* (WIT '10). New York, NY, USA: ACM, 2010, pp. 13–20.

[7] A. Harel, A. Shabtai, L. Rokach, and Y. Elovici, "Eliciting domain expert misuseability conceptions," in *Proceedings of the 6th International Conference on Knowledge Capture* (KCAP '11). Alberta, Canada, 2011