Database Anomalous Activities: Detection and Quantification

Elisa Costante[1], Sokratis Vavilis[1], Sandro Etalle[1], Jerry den Hartog[1], Milan Petković[1,2] and Nicola Zannone[1]

[1] *Eindhoven University of Technology, Den Dolech 2, Eindhoven, The Netherlands*

[2] *Philips Research Europe, High Tech Campus 34, Eindhoven, The Netherlands*

{*e.costante, s.vavilis, s.etalle, j.d.hartog, m.petkovic, n.zannone*}*@tue.nl*

Keywords: Data Misuse, Data Leakage, Database Activity Monitoring, Data Leakage Quantification.

Abstract: The disclosure of sensitive data to unauthorized entities is a critical issue for organizations. Timely detection of data leakage is crucial to reduce possible damages. Therefore, breaches should be detected as early as possible, e.g., when data are leaving the database. In this paper, we focus on data leakage detection by monitoring database activities. We present a framework that automatically learns *normal* user behavior, in terms of database activities, and detects anomalies as deviation from such behavior. In addition, our approach explicitly indicates the root cause of an anomaly. Finally, the framework assesses the severity of data leakages based on the sensitivity of the disclosed data.

1 INTRODUCTION

Data leakage, i.e. the (un)intentional distribution of sensitive data to an unauthorized entity (Shabtai et al., 2012), has a significant impact on businesses in terms of money, reputation or legal issues. Data and information represent a core asset for many organizations and thus they should be protected from unauthorized disclosure.

Timely detection of data leakage is important to reduce costs and comply with legislation. For instance, the new EU data protection regulation imposes strict breach notification policies, requiring organizations to notify data protection authorities of a breach within 24 hours (Information Age, 2012).

In this paper we focus on data leakage detection by monitoring database activities in terms of SQL transactions. Observing SQL activities enables for a prompt reaction by allowing early detection of a leakage, i.e. when the information leaves the database.

In this paper we tackle three main problems. First, despite the considerable amount of research in the area (Fonseca et al., 2007; Kamra et al., 2007; Roichman and Gudes, 2008; Wu et al., 2009; Bockermann et al., 2009; Mathew and Petropoulos, 2010; Gafny et al., 2010; Santos et al., 2012), there are still several problems with current SQL-based solutions for database leakage detection. The main problem of such solutions is their black-box approach: when a potential leakage occurs, an alarm with a *deviation degree*, an *anomaly score*, or a *probability value* is raised. Unfortunately, these values are not sufficient to explain the root cause of a violation and offer little help to seal the leakage. Second, current solutions have a high false positive rate and their validation has been limited to controlled environments: in a real system (with thousands of transactions per minute) even a 5% of false positive rate is unacceptable. Finally, existing solutions are not able to distinguish leakages based on the sensitivity of the data leaked.

The contribution of our solution is twofold:

- We propose a data-leakage detection model that i) automatically learns *normal behavior* by observing database activities and flags deviations from such behavior as anomalies; and ii) explicitly indicates the root cause of an anomaly, e.g. whether the anomaly depends on the tables, data values or data quantity accessed. The monitoring is done by tapping into the communication with the database server to observe which (SQL) queries users submit, and what data go back and forward.

- Once an anomaly is detected, we propose to quantify the leakage based on the sensitivity of the data involved. We use a data model to represent the semantics of information and the relations between different pieces of information. To quantify data leakage, concepts and instances in the model are annotated with sensitivity values representing the cost of their unauthorized disclosure.

2 MOTIVATING SCENARIO

To protect data from unauthorized disclosure, access control solutions are usually employed. Despite this, incidents leaking sensitive information still happen. An example is the data breach happened at the Samaritan Hospital, where two nurses illegitimately accessed and modified medical records[1]. Access control is not enough to protect from data leakage either because in some domains such as healthcare, certain actions cannot be blocked (e.g., accessing patient's medical records in case of emergency), or because it is not possible to specify fine-grained access control policies (e.g., not allow users to access more than a certain amount of data records).

As an example, assume the information system of a hospital is only accessed by its employees. In addition, doctors and nurses can access Electronic Health Record (EHR) of all the patients of the hospital. Notice that, such an assumption is very common in hospitals where data availability is crucial to emergencies. Bob, a geriatrician at the hospital, could easily engage in behaviors that can be classified as threats ($\boldsymbol{T}$). For example, $\boldsymbol{T_1}$) Bob might look for EHR of his colleagues to check if someone has AIDS; or $\boldsymbol{T_2}$) he might know that a famous singer has been hospitalized for a minor surgery, and look for his room number to pay him a visit later; or $\boldsymbol{T_3}$) he might read the EHRs of teenagers while, being a geriatrician, he should only access EHRs of elderly patients.

Some threats, however, might be more serious than others. For example, let assume that a doctor usually accesses 20 EHRs per day. Suppose that one day $\boldsymbol{T_4}$) the doctor reads 500 records containing the *address* of patients at the hospital, or $\boldsymbol{T_5}$) 100 records containing the *disease* suffered by the patients (amongst which there are AIDS or mental disorder). Both situations represent a threat because of the unusual number of records that has been accessed. However, the second threat should be considered more severe because more sensitive data are at risk.

3 RELATED WORK

In this section we discuss the existing Data Leakage Prevention (DLP) solutions. First, we present an overview of solutions general to DLP, and then we dive into solutions specific to Database Activity Monitoring (DAM), the area where we place our contribution. Finally, we discuss the main limitations of the available DAM solutions.

[1]http://www.timesunion.com/local/article/Letters-warn-of-access-to-files-4340178.php

Table 1: General DLP Solutions Overview.

Data State	Location	Approach	Techniques
At Rest	Storage (Database, Data warehouse,...)	Rule-Based	Access Control
			Firewall DB
		Behavior-Based	Audit / Log
			DAM
		Content-Based	-
In Motion	Network (HTTP, FTP, SMTP,...)	Rule-Based	Firewall Rules
			Trusted computing
		Behavior-Based	Anomaly detection (network protocol based)
			Extrusion detection systems
			Honey Tokens / Honey Pots
		Content-Based	Keyword list
			Regular Expressions
			Fingerprinting
			Information Retrieval based
			Data Classification based
In Use	Endpoint Device (Workstation, Laptop, Phone,...)	Rule-Based	Access Control
			Multi-level Security
			Feature Blocking (Print Screen, Printing, USB, ..)
		Behavior-Based	User-Behavior anomaly (System-call based)
		Content-Based	Keyword list
			Regular Expressions
			Fingerprinting
			Data Classification based

General DLP Solutions Data leakage can happen at a different location within an organization, e.g. at a database level, at a network level or at workstation level. DLP systems differ according to which part of an organization perimeter they aim to protect from leakages. Usually, data inside an organization can be in three different states namely, *data-in-use*, *data-in-motion*, and *data-at-rest*. Data-in-use are data currently in process, located at network endpoints such as workstations, laptops and other devices; data-in-motion are data moving through the network that may leave the organization perimeter (e.g., data transmitted over FTP or e-mail); data-at-rest are data residing in repositories such as database servers. Table 1 presents an overview of the existing DLP solutions categorized with respect to data states. For example, in (Carvalho and Cohen, 2007) the focus is essentially on data-in-motion and specifically on the e-mail protocol.

DLP solutions can also differ for the approach used to detect a leakage. The approaches that are usually adopted are *rule-based*, *behavior-based* and *content-based*. In the rule-based approach a set of predefined policies are used to define which operations are allowed or not. Such an approach is used, e.g., in access control mechanisms and firewalls, where a set of rules regulates the access to resources and data. In the behavior-based approach the permitted usage of data is defined by observing users' behavior: DAM technologies apply this approach to detect leakage of data-at-rest. With respect to data-in-motion, network behavior monitoring, such as anomaly detection and extrusion detection systems (Koch, 2011), or Honeypots (Spitzner,

Table 2: Specific DAM Solutions Overview.

		Commercial Solutions						Literature				Our Proposal
		Guardium (IBM)	Database Firewall (Oracle)	Database Security (McAfee)	FortiDB (Fortinet)	Power Broker (Beyond Trust)	SecureSphere (Imperva)	Kamra et al.	Fonseca et al.	Mathew et al.	Gafny et al.	
Approach												
Blacklisting (Signature/Policy Based)		✓	✓	✓		✓	✓					
Whitelisting (Rule Based)		✓	✓		✓							
Whitelisting (Behavior Based)					✓		✓	✓	✓	✓	✓	✓
Features												
Query Based	SQL command (e.g. select, insert)	✓	✓	✓	✓	✓	✓**	✓	✓		✓	✓
	Tables	✓	✓		✓	✓	✓**	✓	✓			✓
	Columns					✓		✓	✓			✓
	Where Clause							✓	✓			✓
Context Based	Application level (e.g. userid, role)	✓						✓			✓	✓
	Contextual info (e.g. time, IP)		✓		✓	✓	✓*				✓	✓
Result-Set Based	#of records or bytes retrieved				✓*		✓*				✓	✓
	Specific values										✓	✓
	Statistics over data values (e.g. avg, std)									✓		✓

*blacklisting is used e.g. to say that accessing time out of working hours is malicious
** behavior-based whitelisting is limited to this feature

2003) can be used to detect unusual behavior. Finally, the content-based approach to DLP directly focuses on data values. Such an approach includes the use of keywords, regular expressions, text classification (Hart et al., 2011), and information retrieval (Gessiou et al., 2011; Gómez-Hidalgo et al., 2010) to detect the presence of sensitive data leaving the organization perimeter.

Commercial DLP systems usually offer a more holistic solution by integrating several of the aforementioned approaches. For instance, MyDLP (http://www.mydlp.com/), focuses on both data-in-use and data-in-motion. In particular, MyDLP monitors the network traffic (data-in-motion) to detect data leakage using content-based techniques. In addition, it regulates the usage of data-in-use by controlling input/output interfaces of workstations.

Specific DAM Solutions DAM provides a solution to detect unusual and unauthorized access to database records at a very early stage. Several database activity monitoring tools are available both in the academic and in the business sectors. Table 2 shows a comparison between currently available solutions. The comparison is done over two dimensions: the *approach* and *features* used to detect data leakages.

The *approach* to detect leakages can be of three types: *blacklisting*, *rule-based whitelisting* and *behavior-based whitelisting*. The blacklisting approach detect leakages based on *unauthorized* actions defined on the *shape* of well-known threats or undesired behavior. In contrast, the whitelisting approach detect leakages as deviations from *authorized* actions. The ways authorized actions are defined differs between rule-based and behavior-based whitelisting. In the first approach authorized actions have to be manually specified in rules, while with the latter approach the authorized actions are automatically learned by observing user database activities.

Features can be defined as the aspects of database transactions taken into account while defining rules and policies (blacklisting and rule-based whitelisting), or while learning normal behavior profiles (behavior-based whitelisting). Features can be classified into *query-based* (related to the syntax of the SQL query), *context-based* (related to contextual information, e.g. where, from whom and when the SQL query has been submitted), and *result-set based* (related to the data retrieved by a SQL query).

As shown in Table 2, most of the commercial solutions (with the exclusion of FortiDB and SecureSphere) use rules (black or white listing) to define unauthorized behavior that may indicate data leakage. In contrast, solutions originating from the academia mostly use a behavior-based whitelisting. With this approach three main steps are necessary to build a detection system: i) define the features that represent a normal behavior profile; ii) learn profiles by observing database activities; and iii) detect anomalies by catching deviations from such profiles. Existing so-

lutions applying this approach mainly differ for the feature set used to profile users, for the algorithms used to learn normal behavior, and for the accuracy (in terms of false positive rates) of the detection model. For example, in (Kamra et al., 2007) *normal profiles* are built using purely query-based features, i.e. the *table* and *columns* accessed by a user *role*; when a new query arrives, it is matched against the profiles. In case a mismatch between the predicted role and the actual role of the user performing the query exist, an alarm is raised. Another query-based approach is the one in (Fonseca et al., 2007), where *normal behavior* profiles are built based on a graph of valid sequence of queries. With this model if an attacker executes valid queries but in an incorrect order (not in the paths of the graph) an anomaly flag is raised.

At the best of our knowledge the work in (Mathew and Petropoulos, 2010) is the only one to build *normal profiles* by using result-set based features. Here, the *records* retrieved by SQL queries are modeled in such a way that it is possible to know what data values a user usually accesses. Note that also FortiDb and SecureSphere use result-set based features such as the *number of record* retrieved. However, in those solutions the values of such features have to be defined a-priori and are not learned by observing users' activities. Finally, Gafny et al. (2010) propose to add context-based features such as who submitted the query (*userid*), at what time (*timestamp*), and from which location (*IP address*) to build behavior profiles.

Limitations Existing approaches do not completely solve the problem of data leakage. For instance, blacklisting approaches are not able to detect certain types of leakage if they are not known in advance. Whitelisting partially solves this problem since, by defining the *normal behavior* of users, it allows the detection of both known and unknown types of leakage. However, in the rule-based whitelisting, an extensive knowledge of internal processes and authorizations is required to define policies of authorized actions. Behavior-based whitelisting does not require such a knowledge and it is more time-effective since authorized behavior is learned by observing users' interactions with the database. The main challenge with behavior-based whitelisting is to identify the features for profiling the *normal behavior* that leads to the best accuracy, i.e. reduces the number of false positives.

With regard to the features, a more extended feature set would intuitively enable the construction of more detailed user profiles, leading to better detection results. Indeed, a pure query-based approach may leave undetected situation where a malicious user performs a valid query, though accessing values for which he should have no interest. For example, with such approach the treats $\boldsymbol{T_1}$, $\boldsymbol{T_2}$, and $\boldsymbol{T_3}$ introduced in Section 2 would not be detected, since the doctor is performing a valid query for his role.

Using result-set features partially solves the previous problem, allowing the detection of situations where the SQL query is legitimate, but the data values accessed are anomalous. Therefore, it is able to detect those threats, which are missed by only considering the query-based feature. However, looking at data values in isolation is not sufficient. Indeed, it does not allows the detection and discrimination of threats like $\boldsymbol{T_4}$ and $\boldsymbol{T_5}$ where the anomaly involves not only the data values retrieved by a query, but also their amount and sensitivity.

4 APPROACH

Based on literature review in Section 3, whitelisting approaches are more suitable for database activity monitoring because they are able to detect both known and unknown data leakage types. In addition, especially in complex and dynamic environments, the definition of authorization rules might require a large amount of time and resources that the automatic self-learning behavior-based approach can save.

For this reason, we propose a framework that uses a behavior-based whitelisting approach to learn normal behavior profiles and to detect deviations from such profiles as anomalies. Intuitively, if behavior profiles are built by combining query, result-set and context based features, it is possible to detect a wider set of data leakage threats. To this end, our approach builds profiles that encompass these different types of features. In addition, our approach is white-box in the sense that it allows the identification of the root causes of a data leakage and thus the quantification of its severity based on the sensitivity of the leaked information. This makes it possible to prioritize alarms and therefore to reduce the impact of false positives. Furthermore, we intend to perform the detection of anomalies *per set of* and not per single query. This way, we can detect situations where a single query is not malicious, but n queries together are (e.g., $\boldsymbol{T_4}$ might be carried out by several small queries instead of a single one which might be more suspicious).

Figure 1 presents a general overview of our framework. Our solution is divided in two distinct phases: learning and detection. During the learning phase, users' activities, in terms of SQL queries to the database, are monitored for a certain period. Such period should be long enough to capture normal database access patterns of each user. The data col-

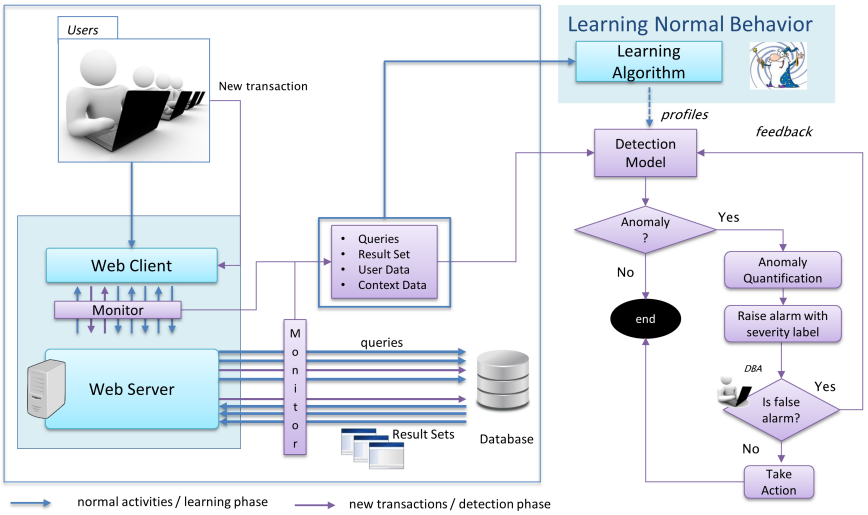

Figure 1: Detection Framework Description.

lected during this period are used to learn *normal behavior profiles*. Such profiles are fine-grain and user-based in the sense that they are able to model information such as which user is accessing which tables and columns, which values is retrieving, at what time and from which location. Profiles are generated with a learning algorithm that is able to deal with flows of large amount of data. Note that in this phase anomalous behavior might be already part of the data collected, e.g., a malicious user is already operating. To deal with such situation we apply outlier detection techniques to highlight *anomalous* transactions before the learning, to analyze and possibly discard them from the learning.

Once profiles have been defined, they are incorporated in a detection model and the detection phase can take place. During detection, each new transaction of a user u is analyzed and matched against his profile. If the transaction does not match the user's profile, an anomaly alarm is raised. The alarm clearly states why the transaction does not match the user profile, e.g. because he is accessing an unusual table, or at unusual time, or accessing unusual amount of data.

Detected anomalies are then further analyzed and labeled with a severity degree. The severity degree depends on the sensitivity of the data involved in the transaction. To measure the sensitivity we employ a data model that is specific to the application domain. The data model provides semantic information about the relationships (e.g., data hierarchy, inference) between pieces of information. Finally, the data model is annotated with the sensitivity, necessary for the quantification.

As an example of our quantification approach we use threat $\boldsymbol{T_4}$ (Section 2). Figure 2 shows an example of data model for healthcare domains. Blue nodes represent the type of information (e.g., demographic or medical data), while green nodes represent different instances of a certain type of information (e.g., personal e-mail or paracetamol medication). We make this distinction, as the sensitivity is defined both in terms of the information type and the value.

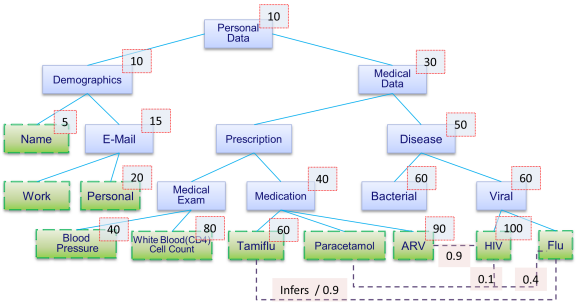

Figure 2: Data Model Example.

Each node in the model is annotated with a sensitivity value which represents the cost of its disclosure. Note that in cases where sensitivity is not explicitly defined, it is propagated downwards through the data hierarchy. For instance, *prescription* inherits the sensitivity of *medical data*. Dashed lines represent inference relationships between the instances, along with the probability that the disclosure of a piece of information makes it possible to infer another piece of information. As an example, a patient treated with *Anti-RetroViral(ARV)* is most probably infected by the *HIV* virus. When the sensitivity of an instance is not explicitly defined, then it is calculated as the weighted average of the sensitivities of all the inferred instances. For example, the sensitivity of information

about *Paracetamol* is calculated as 34, while the sensitivity of *Tamiflu* is explicitly set to 60. By utilizing such annotated data model we i) map the data values accessed by the anomalous query to the data model; and ii) measure the severity of the anomaly.

Once an anomaly is detected and evaluated, it has to be analyzed by a security officer to determine whether it is a real or a false alarm. In the first case, he can take the necessary actions, while in case of a false alarm a feedback is sent back to the detection model, so that the system can automatically learn from its own mistakes and thus reduce the false positive rate.

Note that the proposed solution is not only useful for leakage detection, but also for accountability purposes. Moreover, the monitoring and analysis of how data is actually accessed can help in discovering access patterns that are allowed but undesired; in this case security officers might act on the access policies to solve the problem.

5 CONCLUSION

Prompt detection of data leakages is essential to reduce the damages they can cause to an organization. Data leakages can be detected by observing anomalies in the way data is accessed within the organization perimeter. Focusing the analysis on database activities improves the chances of detecting the leakage at a very early stage. In this paper we proposed a DAM solution to detect anomalous database activities and to quantify them according to the sensitivity of the data leaked.

Our aim is to further develop the proposed framework, and to validate it by the means of extensive tests on real data, coming from different operational environments. The main goal of the validation phase is to show that the system is able to detect a wide range of data leakage attacks, while keeping a low rate of false positive. We are testing our approach in collaboration with an industrial partner in the area of service management. Preliminary results confirm that our approach improves performance, in terms of false positive, if compared with other solutions on the same dataset.

ACKNOWLEDGEMENTS

This work has been partially funded by the THeCS project in the Dutch National COMMIT program.

REFERENCES

Bockermann, C., Apel, M., and Meier, M. (2009). Learning SQL for database intrusion detection using context-sensitive modelling. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, LNCS 5587, pages 196–205. Springer.

Carvalho, V. R. and Cohen, W. W. (2007). Preventing information leaks in email. In *Proceedings of SIAM International Conference on Data Mining*.

Fonseca, J., Vieira, M., and Madeira, H. (2007). Integrated intrusion detection in databases. In *Dependable Computing*, LNCS 4746, pages 198–211. Springer.

Gafny, M., Shabtai, A., Rokach, L., and Elovici, Y. (2010). Detecting data misuse by applying context-based data linkage. In *Proceedings of Workshop on Insider Threats*, pages 3–12. ACM.

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

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

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

Information Age (2012). New EU data laws to include 24hr breach notification.

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

Koch, R. (2011). Towards next-generation intrusion detection. In *Proceedings of International Conference on Cyber Conflict*, pages 1–18. IEEE.

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

Roichman, A. and Gudes, E. (2008). DIWeDa - Detecting Intrusions in Web Databases. In *Data and Applications Security XXII*, LNCS 5094, pages 313–329. Springer.

Santos, R., Bernardino, J., Vieira, M., and Rasteiro, D. (2012). Securing Data Warehouses from Web-Based Intrusions. In *Web Information Systems Engineering*, LNCS 7651, pages 681–688. Springer.

Shabtai, A., Elovici, Y., and Rokach, L. (2012). *A Survey of Data Leakage Detection and Prevention Solutions*. SpringerBriefs in Computer Science. Springer.

Spitzner, L. (2003). Honeypots: Catching the insider threat. In *Proceedings of Computer Security Applications Conference*, pages 170–179. IEEE.

Wu, G., Osborn, S., and Jin, X. (2009). Database intrusion detection using role profiling with role hierarchy. In *Secure Data Management*, LNCS 5776, pages 33–48. Springer.