Focus Article

Data and syntax centric anomaly detection for relational databases

Asmaa Sallam, Daren Fadolalkarim, Elisa Bertino* and Qian Xiao

Recent studies show that insider attacks that aim at exfiltrating data are very common and that these attacks are performed according to specific patterns. Protecting against such threats requires complementing existing security techniques, such as access control and encryption, with tools able to detect anomalies in data accesses. In this paper, we present a technique specifically tailored for detecting anomalous database accesses. Our technique extracts users' access patterns based on both the syntax of the input queries and the amount of data in their output results. Our technique is based on mining SQL queries in database audit logs in order to form profiles of the normal users' access patterns. New queries are checked upon these profiles, and deviations from these profiles are considered anomalous and thus indicative of possible attempts to exfiltrate or misuse the data. Our technique works under two application scenarios. The first is when the database has role-based access control (RBAC) in place. Under an RBAC system, users belong to roles and privileges are associated with roles rather than individual users. For this scenario, we form profiles of roles which make our approach usable for database management systems (DBMSs) that have a large user population; in this scenario, we apply the naive Bayesian classifier which shows accurate results in practice. We also employ multilabeling classification to enhance accuracy when the access patterns are common to multiple roles. The second application scenario is when the DBMS does not apply RBAC. In this scenario, we apply the COBWEB clustering method. Experimental results indicate that our techniques are very effective. © 2016 John Wiley & Sons, Ltd

How to cite this article:
WIREs Data Mining Knowl Discov 2016. doi: 10.1002/widm.1195

INTRODUCTION

Data often encode privacy-sensitive information for which strong protection from both external and internal attackers aiming at stealing data is necessary. Attacks from insiders are particularly difficult to protect against because these insiders are, in most cases, individuals with authorization to access the data. Therefore, conventional access control techniques supported by database management systems (DBMSs) are not able to protect against such attacks. Consider, hypothetically, the case of a clerk within an organization, suppose that she typically accesses 10% of the records in a table in the database for her daily job activity. Access by this clerk when she retrieves all records from the table would not be blocked by the access control mechanism of the DBMS as this clerk would typically have access authorization to the table for her daily tasks. However, the access would certainly be anomalous with respect to the access pattern of this clerk.

Variations in data access patterns are often an indication of possible attempts to steal data.[1] An important technique for data protection against insiders is thus represented by database anomaly detection (AD). Such technique creates profiles of users' data access patterns and then uses these profiles to detect anomalies in new accesses. Approaches for AD for database systems can be categorized into: (1) syntax-based approaches[2–4] by which the access

*Correspondence to: bertino@purdue.edu

Computer Science Department, Purdue University, West Lafayette, IN, USA

Conflict of interest: The authors have declared no conflicts of interest for this article.

profiles consist of features extracted from the query syntax and (2) data-based approaches[5] by which the profiles are created based on the actual data (or sample of data) returned by the queries. These approaches have drawbacks, including being specific to only one DBMS (e.g., PostgreSQL), having overhead at run time for inspecting the returned data, or being unable to deal with applications that retrieve data according to a retrieval loop consisting of several fetch operations.

In this paper, we describe an AD technique that provides rich profiling capabilities by combining syntax-based and data-based profiling and addressing the shortcomings of previous approaches. We consider two application scenarios. In the first scenario, we assume that the database has a role-based access control (RBAC) model in place. Authorizations are specified with respect to roles, and one or more roles are assigned to each user. We aggregate logs of individual users that belong to the same role, and use them to create profiles of roles. The use of roles makes our approach usable for databases with large user populations because managing a few roles is much more efficient than managing many individual users. We model the problem of finding anomalies in this scenario as a classification problem for which we apply the naive Bayesian classifier (NBC) and the multilabeling classifier (MLC). In this paper, we provide a qualitative and quantitative comparison between the two classification algorithms. We also consider the case in which no roles are defined as not all organizations adopt RBAC. In this setting, we build clusters that group similar/close user behavior that we use as the users' profiles. After assessing different clustering algorithms, we used the COBWEB algorithm which showed very good results in AD.

An important issue in the design of the system is how to model and extract information for characterizing the data retrieved by queries. A critical requirement is that the query has to be analyzed before being submitted to the monitored database. Therefore, an approach based on analyzing the data returned by the query[5] would not be suitable. To address this issue, we rely on the query selectivity as estimated by the PostgreSQL DBMS optimizer. Such estimates are included as part of the access profiles.

The paper is organized as follows. We first introduce the system architecture and the syntactic and semantic features extracted from the input SQL queries. We then show how the classification and clustering tasks are performed. We then report experimental results showing the effectiveness of our techniques and comparing the different classifiers and the clustering algorithm. We conclude by further discussing our solution and related work.

METHODOLOGY AND SYSTEM ARCHITECTURE

The system's architecture[4] consists of five main components: target DBMS (T-DBMS), AD engine (ADE), query interceptor (QI), mediator, and the response engine. The T-DBMS is a conventional DBMS that manages the monitored database and handles the query evaluation process. Our AD system operates in two phases: training and detection phases. During the training phase, the ADE builds roles' or users' profiles based on past logs of users' activities. During the detection phase, the ADE compares input queries against these profiles. Profiles consist of query features extracted from the queries that are considered normal for the role. Features of a query are grouped into a record which we call query *quadruplet*. To construct the quadruplet of a query, the feature selector uses an SQL parser and optimizer to extract the query features. It is the mediator's task to load and update the parser and optimizer with information on the schema and statistics of the data stored at the T-DBMS.

During detection, queries sent by the user to the T-DBMS are intercepted for analysis by the QI. The QI then forwards the query to the mediator which then sends the query to the anomaly detector along with other contextual information such as the username of the user who issued the query and the role(s) activated by the user at the time the query is sent. The anomaly detector checks the input query upon roles' or users' profiles and sends the detection result and the query contextual information to the response engine which finds the appropriate response action for the query based on the response policies created by the security administrator (Figure 1).

QUERY REPRESENTATION

Our approach is based on extracting features that represent both the syntax of the query and the data in the query result. We consider SELECT, UPDATE, and DELETE SQL queries. In the case of SELECT queries, the query structure has the following format:

```
SELECT [DISTINCT]            {TARGET_LIST}
FROM                         {RELATION_LIST}
WHERE                        {QUALIFICATION}
```

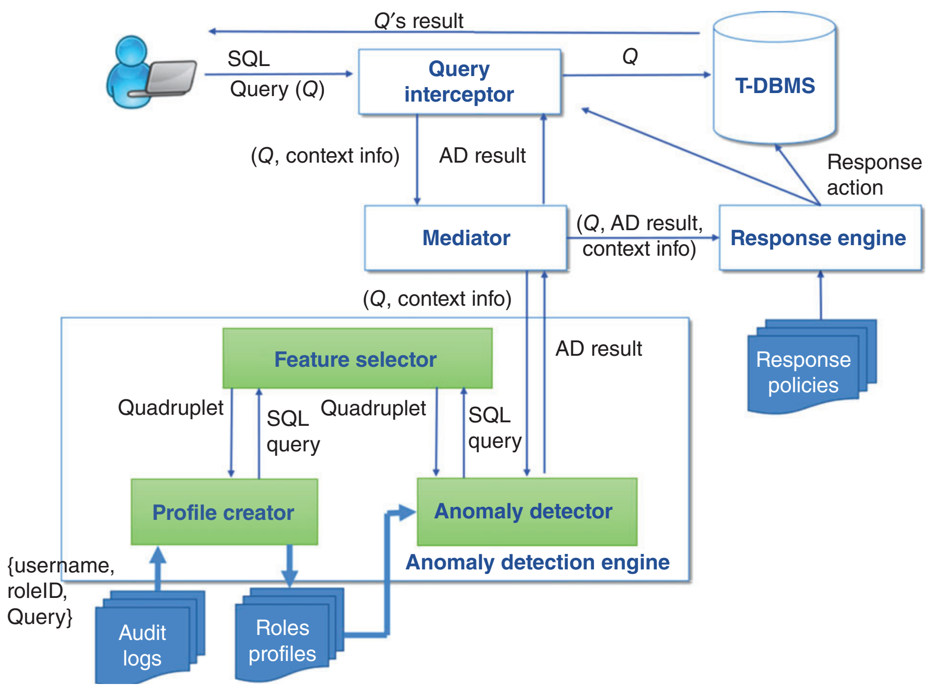

FIGURE 1 | System architecture.

We represent a query by a *quadruplet* which is a 4-ary relation in the form $\mathcal{Q}(c, \mathcal{P}_\mathcal{R}, \mathcal{P}_\mathcal{A}, \mathcal{S}_\mathcal{R})$. $\mathcal{S}_\mathcal{R}$ contains information about the output of the query, while the others are a representation of the query syntax as follows. c corresponds to the type of the command issued by the user. $\mathcal{P}_\mathcal{R}$ indicates the tables accessed by the query and whose data appear in the query result. It is a binary array whose length is equal to the number of tables in the database that stores the projection relation information. The entries in the array corresponding to tables accessed in the query will have the value 1, whereas the others will be 0. $\mathcal{P}_\mathcal{A}$ contains the projection attribute information in the TARGET_LIST of the query. $\mathcal{P}_\mathcal{A}$ is also an array of length equal to the number of tables in the database where each entry corresponds to a table in $\mathcal{P}_\mathcal{R}$ and is itself a binary array of length equal to the number of attributes of the table it corresponds to. If an attribute is accessed in the projection list of the query, its corresponding entry will have the value 1, otherwise it will be 0. $\mathcal{S}_\mathcal{R}$ contains information about the selectivities of the query's range tables. $\mathcal{S}_\mathcal{R}$ is an array of length equal to that of $\mathcal{P}_\mathcal{R}$ as well. If a table is accessed in the query, its corresponding entry in $\mathcal{S}_\mathcal{R}$ will have one of the following values: small (s), medium (m), or large (l), which indicate the portion (selectivity) of the table used to compose the final query result. Small represents a selectivity in the range [0, 0.33[, medium represents a selectivity in the range [0.33, 0.67[, and large represents a selectivity in the range [0.67, 1]. The selectivity of a table is null if the table is not referenced in the query. Methods to estimate the selectivities of tables in an SQL query are described in Ref 4. An example database and the quadruplet representation of example queries on it are shown in Figure 2.

ANOMALY DETECTION

In this section, we show how to apply classification and clustering techniques to find anomalies in users' accesses. *Role-Based Anomaly Detection* section details on the use of the NBC and the MLC when users have associated role information; while *Unsupervised Anomaly Detection* section shows the case when role information is not present.

Role-Based Anomaly Detection

In case the users' accesses have associated role information, we address the AD problem as a classification problem for which we used two different types of classifiers: (1) the naive Bayesian classifier (NBC) and (2) the multilabeling classifier (MLC).

The Naive Bayesian Classifier

The NBC directly applies to our AD framework by considering the set of roles in the system as classes

(a)

Clients

c_ID	c_name
1	c1
2	c2
3	c3
4	c4

Products

p_ID	p_name	Price
1	p1	1
2	p2	3
3	p3	5
4	p4	11
5	p5	13

Clients_products

p_ID	c_ID	Qnty	Total
1	1	10	10
5	1	1	13
2	1	1	3
2	3	2	26

Examples of database

(b)

Query	Quadruplet (c, P_R, P_A, S_R)
SELECT * FROM Products WHERE price < 10;	('SELECT' ,[0, 1, 0], [[0, 0], [1, 1, 1], [0, 0, 0, 0]], [null, 'm', null])
SELECT * FROM Clients, Clients_Products WHERE total >= 20 and Clients.c_ID = Clients_Products.c_ID;	('SELECT' ,[1, 0, 1], [[1, 1], [0, 0, 0], [1, 1, 1, 1]], ['s', null, 's'])
SELECT * FROM Clients, Clients_Products WHERE Clients.c_ID = Clients_Products.c_ID;	('SELECT' ,[1, 0, 1], [[1, 1], [0, 0, 0], [1, 1, 1, 1]], ['m', null, '1'])

Examples of quadruplets

FIGURE 2 | Query representation.

and the log-file quadruplets as the training observations. We chose to apply the Maximum-Aposteriori (MAP) decision rule which is most commonly used with the NBC and can be formulated as

$$r_{map} = \arg\max_{r_j \in R} \left(p(r_j)\, p(c|r_j) \prod_{i=1}^{N} p\left(\mathcal{P}_{\mathcal{R}}[i].\mathcal{P}_{\mathcal{A}}^{\mathcal{T}}[i] | r_j\right) .p\left(\mathcal{P}_{\mathcal{R}}[i].\mathcal{S}_{\mathcal{R}}^{\mathcal{T}}[i] | r_j\right) \right) \quad (1)$$

where r_{map} is the output of the classifier, that is, the expected role of the user submitting the query, R is the set of roles in the database and N is the number of tables in the database. After r_{map} is computed, it is compared to the actual role of the user; if they are identical, the query is considered normal, otherwise it is considered anomalous.

The Multilabeling Classifier

An important problem to consider when choosing the classifier to use is the case in which there is overlap between the access patterns of roles. Classifiers like the NBC do not produce accurate results in this case. MLCs are well suited to address this problem. The idea behind the MLC is to associate each training query instance with more than one class/role if the instance is common to multiple roles. During detection, an input query can be labeled with more than one role. If the role of the user is not one of the roles predicted by the classifier, the query is considered anomalous.

In our work, we used the BR[6] method where R binary SVM[7] classifiers are used when R roles exist in the database. Each classifier distinguishes between whether a role is ON or OFF. To classify a new input instance, BR outputs the aggregation of the positive labels (indicated ON) as predicted by all binary classifiers. We preferred using this method rather than creating a classifier for each pair of roles and using a voting scheme to find the possible source roles of the input query; the reason is that the voting approach does not guarantee correct result when a few roles are present in the database.

Unsupervised Anomaly Detection

We now consider the case when the DBMS or users do not use roles or when there is no information about roles available in the log files. In this case, the problem of forming user profiles is an unsupervised learning problem and thus we treat it as a clustering problem. The methodology we follow for AD is as follows. First, we partition the training data into clusters using a standard clustering algorithm. We maintain a mapping for every user and the clusters that her queries belong to. Note that each user can be mapped to more than one cluster depending on the nature of the training data. For every new query under observation, we use the same clustering algorithm to find the cluster that the query belongs to. We then check that the user submitting this query has some queries in this specific cluster using the user-cluster mapping discussed earlier; if so, the query is considered normal and anomalous otherwise.

In the process of choosing the clustering algorithm to use, we tried several ones, including k-means,[8] Expectation-Maximization (EM),[9] Farthest-

First,[10,11] and COBWEB[12,13] algorithms. We chose to use COBWEB as it proved to have good detection results in our problem setting as will be shown in the evaluation section. COBWEB is an example of conceptual clustering by which a classification tree is created based on the training observations.

PERFORMANCE EVALUATION

In this section, we report experimental results obtained using the OLTP Benchmark.[14] As it is typically hard to gain access to real datasets, we decided to use the OLTP Benchmark because it is the reference benchmark for evaluating the performance of DBMSs and contains diverse datasets and workloads. Because the benchmark does not have role definitions, we investigated the meaning and purpose of each scenario in each dataset in the benchmark and defined roles for the database for each workload. We defined roles for each of the following workloads: AuctionMark, Epinions, Seats, and TPCC (Table 1). Each of these workloads has two roles: Worker and Client roles. The AuctionMark and Epinions datasets have overlapping access patterns between Workers and Clients which would help us understand the behavior of the classifiers in this case.

TABLE 1 | Training Time (Min) and Size of OLTP Benchmark Datasets

Dataset	NBC	MLC	COBWEB	Size (recs)
Auctionmark	1.04	1.52	1.19	11.3 k
Epinions	2.48	3.41	3.5	26.6 k
Seats	1.83	3.35	4.32	18.1 k
TPCC	0.85	1.13	1.01	7.4 k

To perform our experiments, we executed the table creation, role definition, and table insertion scripts for each workload individually. Then, we started the training of the ADE. Afterwards, we turned on the detection phase at the ADE. For this set of experiments, we computed the following metrics for the NBC, the MLC, and COBWEB clusterer for each of the four OLTP datasets: number of true negatives (TN), number of false positives (FP), number of false negatives (FN), number of true positives (TP), true positive rate (TPR), false positive rate (FPR), true negative rate (TNR), false negative rate (FNR), precision, accuracy, and F1-score. Based on the results for these metrics, we computed the area under the ROC curve (AUC) for the classifiers and the clusterer. To compute the false positives, we used 80% of the workload logs queries to train the classifiers and the clusterer, and checked the remaining 20% of the queries during the detection phase of AD; an anomaly in this case is considered a false positive. To compute the rate of false negatives, we sent the training log queries with role information (user information in case of clustering) negated (remember that each workload contains only two roles/users); because all of these accesses should be deemed anomalous, if the result of a query indicates a nonanomalous access, this is considered a false negative. The results are shown in Tables 2–4.

It can be noted from the detection results that the MLC has the best accuracy when compared to the NBC and COBWEB. The NBC has very high false positives for the AuctionMark and Epinions datasets; the reason is that these datasets have overlapping queries between the roles and the NBC is not able to handle well such case as mentioned earlier.

COBWEB, however, performs reasonably well on AuctionMark dataset. However, it has high false

TABLE 2 | Results for the NBC

	Confusion Matrix										
	TN	FP	FN	TP	TPR	FPR	TNR	FNR	Prec	Acc	F1
Auction Mark	470	125	151	1817	92.33	21.01	78.99	7.67	93.56	89.23	92.94
Epinions	9531	4736	570	3537	86.12	33.2	66.8	13.88	42.75	71.12	57.14
Seats	2581	95	259	7047	96.45	3.55	96.45	3.55	98.67	96.45	97.55
TPCC	3636	81	16	738	97.88	2.18	97.82	2.12	90.11	97.83	93.83
				Avg	93.195	14.985	85.015	6.805	81.2725	88.6575	85.365
				Std	5.27	14.86	14.86	5.27	25.92	12.28	18.92
				Conf	5.16	14.56	14.56	5.16	25.4	12.03	18.54
											AUC = 88.96

TABLE 3 | Results for the MLC

	Confusion Matrix										
	TN	FP	FN	TP	TPR	FPR	TNR	FNR	Prec	Acc	F1
Auction Mark	595	0	134	1834	93.19	0	100	6.81	100	94.77	96.47
Epinions	14,267	0	0	4107	100	0	100	0	100	100	100
Seats	2676	0	0	7306	100	0	100	0	100	100	100
TPCC	3674	43	2	752	99.73	1.16	98.84	0.27	94.59	98.99	97.09
				Avg	98.23	0.29	99.71	1.77	98.6475	98.44	98.39
				Std	3.36	0.58	0.58	3.36	2.71	2.49	1.88
				Conf	3.29	0.57	0.57	3.29	2.66	2.44	1.84
											AUC = 99.87

TABLE 4 | Results for COBWEB

	Confusion Matrix										
	TN	FP	FN	TP	TPR	FPR	TNR	FNR	Prec	Acc	F1
Auction Mark	595	0	226	1742	88.52	0	100	11.48	100	91.18	93.91
Epinions	14,267	0	988	3119	75.94	0	100	24.06	100	94.62	86.32
Seats	2676	0	107	7199	98.54	0	100	1.46	100	98.93	99.26
TPCC	3717	0	7	747	99.07	0	100	0.93	100	99.84	99.53
				Avg	90.5175	0	100	9.4825	100	96.1425	94.755
				Std	10.86	0	0	10.86	0	4.02	6.19
				Conf	10.64	0	0	10.64	0	3.94	6.07
											AUC = 99.53

negative rate for Epinions dataset. The problem is that COBWEB cannot distinguish between the following two queries:

(1) SELECT avg(rating) FROM trust WHERE …
(2) SELECT avg(rating) FROM trust, review WHERE …

These two queries differ in one feature in the quadruplet which is the selectivity of the table review which would be null in the first query and ‘s’ in the second one. When such cases do not arise, COBWEB performs pretty well even when the roles have overlapping access patterns.

Table 1 shows the time required (in minutes) to train the NBC, the MLC, and the COBWEB clusterer for each dataset in the benchmark and the size of each workload. For all our time-measurement experiments, we used a virtual machine running Ubuntu Linux that has 3 cores and 6GB RAM. It is clear that in general as the dataset size increases, the training time increases too. For the largest dataset, which is Epinions, that contains 26.6 k records, the training time does not exceed 3.5 min. Since training is done offline, the training rates that we observed from our experiments are acceptable. The average time required to do detection is 12 millisecond for the NBC, 0.89 second for the MLC, and 17 millisecond for the clusterer. In all cases, the detection time is less than a second and thus acceptable as part of the response time for a query.

RELATED WORK

AD has been widely used for detecting attacks. Many papers have surveyed AD methods and their applications[15–20]. These include databases, web applications, networks, medical and public health, and many other domains.

The work presented in this paper classifies as AD in database systems. In this domain, several AD applications have been proposed which can be classified into two categories: (1) detection of attacks performed by sending explicit SQL commands to the DBMS and (2) detection of attacks performed through the use of applications. Our approach is in the first category, in that we consider attacks by which insiders try to exfiltrate the data stored in the database through direct SQL commands.

Approaches to AD in database systems that assume that users interact with the database through SQL commands can be grouped into two categories: syntax centric and data centric approaches. Syntax centric AD relies on the SQL query syntax to construct user profiles. Previous work by Kamra, Bertino et al. proposed a role and user-based syntax centric AD approach.[2,3,21] Even though the methods proposed are computationally fast as they only need to parse the query in order to perform AD, some attributes of the query cannot be captured by relying upon the query syntax only, such as the size of the query result set and the characteristics of the portions of accessed tables that will show in the result. The approach presented in this paper is a major extension of our previous work[4] to take into consideration the common access patterns between roles.

Data centric AD relies on analyzing the results of the query. Mathew et al.[5] proposed a data centric AD system that consists of extracting features from the query results. Specifically, certain statistics are computed for each attribute in the result depending on the type of the attribute. This method has the problem of having to evaluate the query to perform AD. The system thus cannot prevent a malicious query from performing changes to the database. Costante et al.[22] propose the use of semantic features of the query extracted from the query execution result in addition to some query contextual information for query representation. This approach has the same problem of the need for query execution to detect anomalies.

Some systems have been proposed to protect database systems from exfiltration through applications. The use of applications to access the database imposes a different kind of restriction than the use of direct commands, that is the control and data flow of the program that enforce a specific order of the commands sent to the DBMS. Rafiul et al.[23] propose a system to detect data exfiltration attacks from insiders performed through applications. Such system creates an expected program execution flow graph and monitors at run time, the actual execution to detect anomalies with respect to the expected execution. DIDAFIT[24] is also a system that operates at the database application level and summarizes SQL queries in the application programs into compact regular expressions called fingerprints. New accesses are compared against these fingerprints and mismatching indicates anomalies. DIDAFIT, however, does not take into account the order of commands in the program. Solutions for attacks performed through web applications have been widely investigated.[25–29] These systems, as mentioned earlier, are addressing different application architectures and contexts than the ones assumed in this paper.

Existing AD methods can also be categorized into online and offline methods. In online methods, the detection is done on an input basis and the AD result is ready before the query result is returned to the user. Therefore, online methods support rigorous query response actions[30] such as the action of dropping the query, and thus not returning the query answer to the user or not applying the user input changes depending on the query type. The approach presented in this paper is an online approach because it does not require the execution of the query to perform AD. While the approach presented by Mathew et al.[5] is considered an offline approach, because the detection is executed after the query execution is completed. Spalka et al.[31] compared an offline and an online approaches. The offline approach computes reference values on monitored attributes and, periodically, the values of these attributes are checked against their reference values. An anomaly is raised if the difference exceeds a threshold set by the user. The online approach stores Δ-relations that keep the track of changes performed on the database tables. These changes are periodically applied on artificial tables that represent the monitored ones and then the offline approach is applied on these artificial tables to detect anomalies. Both the online and offline approaches have the problem that they focus on update commands only and cannot detect malicious read commands.

CONCLUSIONS

In this paper, we have described an approach that complements existing security techniques for databases in order to protect highly sensitive data from insiders. Our approach relies on AD techniques specifically designed for DBMSs and is capable of detecting changes in access patterns based on the syntax of the input queries and the amount of data in the queries' results. We evaluated the performance of the NBC, MLC, and COBWEB clustering on datasets from OLTP benchmark. Experimental results show that our methods are very effective. Our work shows that there is no single data mining technique that works for all types of access patterns, including direct access by users and role-mediated access. Therefore, the development of an AD system that supports multiple data mining techniques and is, automatically, able to select which one to use for which users and data would be critical. In addition, discussions with system users have shown that information about query temporal patterns and support for access patterns evolutions are critical for effective AD systems.

REFERENCES

1. Silowash G, Cappelli D, Moore A, Trzeciak R, Shimeall T, Flynn L. Common sense guide to mitigating insider threats. Technical Report CMU/SEI-2012-TR-012, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2012. Available at: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=34017. (Accessed August 22, 2016)
2. Bertino E, Kamra A, Terzi E, Vakali A. Intrusion detection in RBAC-administered databases. In: *Proceedings of the 21st Annual Computer Security Applications Conference*, ACSAC '05, IEEE Computer Society, Washington, DC, USA, 5-9 Dec. 2005, 170–182.
3. Kamra A, Terzi E, Bertino E. Detecting anomalous access patterns in relational databases. *VLDB J* 2008, 17:1063–1077.
4. Sallam A, Bertino E, Hussain SR, Landers D, Lefler M, Steiner D. DBSAFE—an anomaly detection system to protect databases from exfiltration attempts. *IEEE Syst J* 2015, 99:1–11.
5. Mathew S, Petropoulos M, Ngo HQ, Upadhyaya S. 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, Springer-Verlag, Ottawa, Ontario, Canada, September 15 - 17 2010, 382–401.
6. Read J, Pfahringer B, Holmes G, Frank E. Classifier chains for multi-label classification. *Mach Learn* 2011, 85:333–359.
7. Cortes C, Vapnik V. Support-vector networks. *Mach Learn* 1995, 20:273–297.
8. Lloyd SP. Least squares quantization in PCM. *IEEE Trans Inf Theory* 1982, 28:129–136.
9. Dempster AP, Laird NM, Rdin DB. Maximum likelihood from incomplete data via the EM algorithm. *J R Stat Soc Series B* 1977, 39:1–38.
10. Dasgupta S. Performance guarantees for hierarchical clustering. In: *15th Annual Conference on Computational Learning Theory*. Springer, Sydney, Australia, July 8-10 2002, 351–363.
11. Hochbaum DS, Shmoys DB. A best possible heuristic for the k-center problem. *Math Oper Res* 1985, 10:180–184.
12. Fisher D. Knowledge acquisition via incremental conceptual clustering. *Mach Learn* 1987, 2:139–172.
13. Gennari J, Langley P, Fisher D. Models of incremental concept formation. *Artif Intell* 1990, 40:11–61.
14. Eddine Difallah D, Pavlo A, Curino C, Cudre-Mauroux P. Oltp-bench: an extensible test bed for benchmarking relational databases. *Proc VLDB Endow* 2013, 7:277–288.
15. Chandola V, Banerjee A, Kumar V. Anomaly detection: a survey. *ACM Comput Surv* 2009, 41:15:1–15:58.
16. Chandola V, Banerjee A, Kumar V. Anomaly detection for discrete sequences: a survey. *IEEE Trans Knowl Data Eng* 2012, 24:823–839.
17. Estevez-Tapiador JM, Garcia-Teodoro P, Diaz-Verdejo JE. Anomaly detection methods in wired networks: a survey and taxonomy. *Comput Commun* 2004, 27:1569–1584.
18. Victoria JH, Austin J. A survey of outlier detection methodologies. *Artif Intell Rev* 2004, 22:85–126.
19. Patcha A, Park J-M. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Comput Network* 2007, 51:3448–3470.
20. Xie M, Han S, Tian B, Parvin S. Anomaly detection in wireless sensor networks: a survey. *J Netw Comput Appl* 2011, 34:1302–1325. Advanced Topics in Cloud Computing.
21. Shebaro B, Sallam A, Kamra A, Bertino E. PostgreSQL anomalous query detector. In: *Proceedings of the 16th International Conference on Extending Database Technology*, EDBT '13. ACM, Genoa, Italy, March 18-22 2013, 741–744.
22. Costante E, Vavilis S, Etalle S, den Hartog J, Petkovic M, Zannone N. Database anomalous activities detection and quantification. In: *International Conference on Security and Cryptography* (*SECRYPT*), Reykjavík, Iceland, 29-31 July, 2013, 1–6.
23. Rafiul Hussain S, Sallam AM, Bertino E. Detanom: detecting anomalous database transactions by insiders. In: *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, CODASPY '15, ACM, San Antonio, TX, USA, March 2-4 2015, 25–35.
24. Lup Low W, Lee J, Teoh P. Didafit: detecting intrusions in databases through fingerprint transactions. In: *Proceedings of the 4th International Conference on Enterprise Information Systems, Ciudal*, Ciudad Real, Spain, 3-6 April 2002, 2–6.
25. Cova M, Balzarotti D, Felmetsger V, Vigna G. *Swaddler: An Approach for the Anomaly-Based Detection of State Violations in Web Applications*. Berlin and Heidelberg: Springer; 2007, pp. 63–86.

26. Felmetsger V, Cavedon L, Kruegel C, Vigna G. Toward automated detection of logic vulnerabilities in web applications. In: *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, USENIX Association, Washington, DC, USA, August 11-13 2010, 10–10.
27. Ficco M, Coppolino L, Romano L. A weight-based symptom correlation approach to SQL injection attacks. In: *Proceedings of the 2009 Fourth Latin-American Symposium on Dependable Computing*, LADC '09, IEEE Computer Society, João Pessoa, Brazil, September 01-04 2009, 9–16.
28. Huang Y-W, Huang S-K, Lin T-P, Tsai C-H. Web application security assessment by fault injection and behavior monitoring. In: *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, ACM, New York, NY, USA, 2003, 148–159.
29. Valeur F, Mutz D, Vigna G. A learning-based approach to the detection of SQL attacks. In: *Proceedings of the Second International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA'05, Springer-Verlag, Vienna, Austria, July 7-8 2005, 123–140.
30. Kamra A, Bertino E. Design and implementation of an intrusion response system for relational databases. *IEEE Trans Knowl Data Eng* 2011, 23:875–888.
31. Spalka A, Lehnhardt J. A comprehensive approach to anomaly detection in relational databases. In: *Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, DBSec'05, Springer-Verlag, Storrs, CT, USA, August 7-10 2005, 207–221.