A Comparison of Data Mining Techniques for Anomaly Detection in Relational Databases

Charissa Ann Ronao, Sung-Bae Cho
Computer Science Department
Yonsei University
Seoul, South Korea
cvronao@sclab.yonsei.ac.kr, sbcho@cs.yonsei.ac.kr

***Abstract*—Data mining has gained a lot of attention in recent years especially with the advent of big data. In line with this, relational database management systems (RDBMS) have also become the ultimate layer in preventing malicious data access. However, despite the presence of traditional database security mechanisms, it is apparent that database intrusions still occur. Thus, there is an imminent need in developing a robust and efficient intrusion detection system (IDS) especially tailored for databases. Among the few studies that have been published with regards to the problem at hand, most researchers have proposed the use of data mining techniques to detect database anomalous behavior. However, up to this date, there has been no work aimed to objectively compare these various data mining techniques as applied to the field of database IDS. In this paper, we evaluate the state-of-the-art feature selection and data mining algorithms in the context of database IDS and provide a clear performance comparison of these techniques under common grounds. Experiments show that principal components analysis produces a reasonably compact and meaningful subset of features while graphical models like decision trees, random forest, and Bayesian networks yield a consistently high performance in detecting anomalies in databases.**

Keywords-intrusion detection; anomaly detection; database security; data mining; analysis.

I. INTRODUCTION

In today's information revolution era, data has become more and more indispensable to individuals, companies and organizations. This paved the way to developing relational database management systems (RDBMS), which can organize, contain, and protect these data from malicious threats. However, despite access controls and firewalls that are widely incorporated in these systems, it has been found that they are inadequate in defending against anomalous attacks. Moreover, network-based and host-based intrusion detection systems (IDS), although having been extensively researched and implemented in recent years, are awfully insufficient and unsuitable in detecting attacks specifically targeted to databases [1]. In particular, insider threats are as much of a concern as outsider threats, i.e., privileged users, if corrupt, can potentially cause more damage than average users. While many works have focused on how data can be protected from external attacks, there have been very few researches regarding the problem of protecting data from insider threats [2]. Because of this, there has been a growing awareness that a strong and effective IDS especially tailored for databases needs to be developed.

An efficient and robust intrusion detection mechanism is crucial in building a strong database security framework. In line with this, a number of data mining techniques have been proposed to perform this task [3]. Although previous works have integrated data mining algorithms in their IDS framework, to the best of our knowledge, none of these works have performed an in-depth evaluation and performance comparison of data mining algorithms in the context of database intrusion detection. To address this, this paper provides a clear comparison and parallel evaluation of state-of-the-art data mining methods in the application of database IDS. We mine SQL query logs and exploit the presence of role-based access control (RBAC) mechanism, which has already been adopted in various commercial RDBMS products [4], to detect anomalies. We model normal access behavior through these queries along with their corresponding role annotations, and detect anomalies by tagging queries that deviate from these normal access behaviors.

The rest of the paper is organized as follows: Section II describes the related work, followed by the discussion of system architecture, feature extraction, state-of-the-art feature selection methods, and data mining techniques in Section III. Section IV presents our experiment results, and finally, we draw our conclusion in Section V.

II. RELATED WORK

IDS's are generally divided into two main categories: signature-based and anomaly-based. Signature-based or misuse-based systems make use of explicitly defined attack signatures and detect intrusions by blacklisting. This kind of system is ineffective in the face of new types of attacks, which, in turn, makes it susceptible to evasion methods that take advantage of the expressiveness of the SQL language [5]. On the other hand, anomaly-based systems model normal behavior in the form of intrusion-free logs and marks deviations from this normal behavior as anomalies [6]. Unlike the former, these systems are clearly more robust to unknown attacks and to malicious users who may keep on evolving their attack strategy.

One of the most common way of implementing an anomaly-based IDS is by exploiting data mining algorithms as the detection mechanism. In the past decade, a number of data mining techniques have been proposed for the purpose of detecting intrusions in databases. Among these are the use

 ISBN: 978-1-61208-381-0

of data dependency and association rules [7][8][9]. Such methods, however, require the manual assignment of attribute weights; they also cannot be scaled easily to typical database sizes [3]. Another technique was proposed by Barbara et al. [10], who made use of hidden Markov models (HMM) to capture the change in database normal behavior over time. This too, however, is impractical to implement in large databases with many tables and attributes. Consequently, Ramasubramanian et al. integrated artificial neural networks (ANN) into their proposed IDS framework [11], while Pinzon et al. made use of support vector machines (SVM) and multilayer perceptrons (MLP) to detect outsider attacks [12]. These papers have focused mainly on the structure development of the database IDS framework, and did not sufficiently evaluate the underlying core mechanism, which is the data mining technique. Furthermore, Kamra et al. proposed an IDS which exploits a naïve Bayes (NB) classifier to detect abnormal behavior [4]. The latter had based their approach on the RBAC model, a standardized access control mechanism, building a profile for each role, and checking the behavior of each role with respect to the profile [14]. The main idea is to assign one or more roles to each user, and assign privileges to roles. This effectively minimizes the number of profiles to maintain, which makes it scalable to a large database user population, and is a much more efficient method compared to managing a profile for each individual user. We adopt the same rationale and build normal profiles through roles and SQL query access.

We stress, however, that all mentioned works lack the necessary evaluation step of analyzing the features they have extracted and comparing their proposed data mining approach to other state-of-the-art techniques. We believe that merely applying an algorithm to the problem and showing its satisfactory results are not enough to prove the effectiveness and efficiency of the system—a clear comparison and parallel evaluation must be made to know how these algorithms perform in detecting intrusions under common grounds, most especially, in the data mining perspective.

III. DATA MINING FOR DATABASE INTRUSION DETECTION

We exploit the existence of the RBAC mechanism and model normal access behavior profiles through roles. Normal access behavior is represented by intrusion-free SQL queries, and they are used to train a data mining algorithm to produce normal profile models. We define an anomaly as an access behavior that deviates from these normal profiles. Given these profiles, clearly, we have a standard classification problem.

A. Intrusion Detection System

Figure 1 shows the intrusion detection process. Every time a query is issued, the profile logs are updated. During the training phase, normal access behavior, in the form of SQL queries grouped into profiles, are fed to the feature extractor, feature selector, and finally, the data mining algorithm or classifier; the classifier then produces a trained model out of normal access behavior. During the detection phase, each new query goes through the feature extractor and selector, and is evaluated by the trained classifier. An alarm is raised if the query deviates from normal profiles. We emphasize that role profiles should be regularly updated and classifier training periodically done, so as to be able to update the normal profile models and minimize false alarms.

Given this setup, there are three main problems: (1) how to extract and represent features, (2) which of these features to use, and (3) which data mining technique to employ. We discuss the solutions to these three problems in the following section.

B. SQL Query Parsing and Feature Extraction

One SQL query corresponds to an entry in the database log file, which follows the SQL language syntax. For simplicity, we illustrate the SQL grammar with the SELECT command:

```
SELECT      <Projection attribute clause>
FROM        <Projection relation clause>
WHERE       <Selection attribute clause>
ORDER BY    <ORDER BY clause>
GROUP BY    <GROUP BY clause>
```

We parse queries in this manner, line-by-line, and extract features accordingly in order to transform SQL log entries into feature vectors that can be understood and processed by data mining classifiers.

We gather proposed features from database IDS literature and combine them to form a more complete feature set [4][12][15]. We represent a query as a feature vector Q with seven fields: Q(`SQL-CMD[], PROJ-REL-DEC[], PROJ-ATTR-DEC[], SEL-ATTR-DEC[], ORDBY-ATTR-DEC[], GRPBY-ATTR-DEC[], VALUE-CTR[]`), as seen in Table I. Query mode, c, represents the query commands SIUD: if the query command is `SELECT`, it is represented by integer 1; if `INSERT`, integer 2; if `UPDATE`, integer 3; and if `DELETE`, integer 4. Query length, Q_L, is denoted by the number of characters in the whole query, including spaces. The number of string values, S_V, and numeric values, N_V, indicate how many times these values appear in the selection clause. The same logic is applied with the number of JOINs, J, and ANDs/ORs, AO.

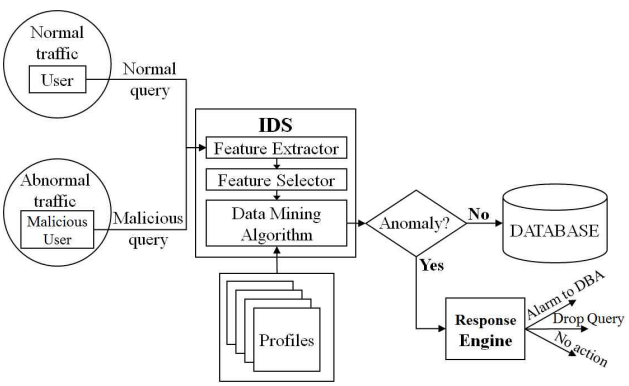

Figure 1. Flow of the IDS process.

TABLE I. LIST OF EXTRACTED FEATURES

Vector field	Description	Feature elements
SQL-CMD[]	Command features	query mode, *c* query length, Q_L
PROJ-REL-DEC[]	Projection relation features	Number of projected relations, P_R Position of projected relations, P_{RID}
PROJ-ATTR-DEC[]	Projection attribute features	(P_A, P_A[], P_{AID}[])[a]
SEL-ATTR-DEC[]	Selection attribute features	(S_A, S_A[], S_{AID}[])[a]
ORDBY-ATTR-DEC[]	ORDER BY clause features	(O_A, O_A[], O_{AID}[])[a]
GRPBY-ATTR-DEC[]	GROUP BY clause features	(G_A, G_A[], G_{AID}[])[a]
VALUE-CTR[]	Value counter features	Number of string values, S_V Length of string values, S_L Number of numeric values, N_V Number of JOINs, *J* Number of ANDs and ORs, *AO*

a. Convention (N_A, N_A[], N_{AID}[]):
N_A – number of attributes in a particular clause
N_A[] – number of attributes in a particular clause counted per table
N_{AID}[] – position of the attributes present in a particular clause, represented in decimal

In addition, the number of relations, P_R, indicates how many tables are present in a specific clause. The position of relations, P_{RID}, is represented by a binary string, wherein each bit stands for a table in the database schema. If a table is present in the query, its bit representation is 1; if it is absent, it is represented by bit 0. Wu et al. stated that different input encoding schemes (binary or decimal) result to different algorithm performance results [16]. Decimal encoding was found to be more robust to noise and decreases computational complexity; thus, to get the final value of the ID feature, we convert the binary string into its decimal form. The same logic is applied to the mapping of the positions of attributes given a specific clause. Thus, all ID features are represented by a single decimal value.

Extending the parsing and feature extraction method to other commands such as `INSERT`, `UPDATE`, and `DELETE` is clearly straightforward. A total of 21 main features are extracted for every query in the SQL log, with some features (e.g., ID features) branching out to sub-features that depend on the number of tables and attributes in the database schema. For example, for a schema consisting of 2 relations with 4 attributes each, the resulting number of features will be 45.

C. Feature Selection Methods

Selecting good feature sets improves performance, eliminates noise, and enables faster and more accurate detection [17]. We use five feature selection methods to evaluate the extracted features, and they are categorized into two groups: ranking methods and filter methods.

Ranking methods output the complete feature set sorted from highest to lowest according to a certain evaluation measure. Since the top variables are considered to be the most discriminant features, a certain threshold should be determined to cut off features that are considered to have little or no contribution to the classification process. One of the most common evaluation measure when ranking features is information gain (IG). It is the expected reduction in entropy caused by partitioning a query data set according to a certain feature. Given a query data set *S* with *K* different roles/classes, entropy is given by:

$$I(S) = -\sum_{k=1}^{K} \frac{|s_k|}{|s|} \log \frac{|s_k|}{|s|}, \quad (1)$$

where *s* is the total number of queries in the data set and s_k is the number of queries in class *k*. We get the IG of feature *Y* which can partition *S* into *M* subsets by,

$$IG(Y) = I(S) - \sum_{m=1}^{M} \frac{|s_m|}{|s|} I(S_m), \quad (2)$$

where the second term is the conditional entropy, $I(S|Y)$, and s_m is the number of queries in subset *m*.

An improved variant of IG is gain ratio (GR), which overcomes the bias of the former towards features that can have a large number of possible values. GR applies a kind of normalization to IG by using the information value corresponding to *M* outcomes on feature *Y*, i.e.,

$$IV(S \mid Y) = -\sum_{m=1}^{M} \frac{|s_m|}{|s|} \log \frac{|s_m|}{|s|}. \quad (3)$$

Dividing (1) by (3) gives the GR of feature *Y*.

Principal components analysis (PCA) is an unsupervised ranking feature selection technique which can transform query data set *S* into a new coordinate system and produce a set of components $p \in P$ wherein the top components, called principal components (PCs), represent the greater part of the variance of *S*. With this, we can easily eliminate the tailing *p*'s (those that does not contain much of the variance of the *S*). Scaling and standardizing is often applied before PCA to simplify the latter's calculation.

In contrast, filter methods automatically output a set of chosen features based on a certain evaluation measure. One of these methods, best first search (BFS), is a combination of forward selection and backward elimination which can greedily search through the query feature space. In the case when performance starts to drop, it can backtrack previous feature subsets (those with good enough performance) and start again from there. However, for a high dimensional query data set *S* (which depends on how big the database schema is), BFS can be computationally expensive.

Genetic algorithm (GA) is another filter method based on the principle of natural selection, which randomly creates a population *N* of possible feature subsets *n* (any combination of fields from *Q*) and evaluates each one by a certain measure (e.g. correlation). GA runs for several generations, each time creating a new *N* by performing crossover and mutation. This method has been proven to be very effective in practice [11].

 ISBN: 978-1-61208-381-0

D. Data Mining Algorithms

We consider the following state-of-the-art classifiers which have been successfully applied in the intrusion detection domain, namely: naïve Bayes, K-nearest neighbors, artificial neural networks and multilayer perceptrons, support vector machines, Bayesian networks, J48 decision trees, and random forest [4][11][12][13][19][20][21].

Naïve Bayes (NB) is a simple classifier with strong feature independence assumptions. Given a new query $q \in Q$ with a set of features $Y = \{y_1,\ldots,y_d\}$, and k roles/classes, we compute the posterior probability of class membership, i.e., the probability that Y belongs to role r_k, by,

$$p(r_k \mid Y) \propto p(r_k)\prod_{i=1}^{d} p(y_i \mid r_k). \quad (4)$$

Using (4), we can classify q into a role r_k that achieves the highest posterior probability.

Another method based on Bayes theorem is Bayesian network (BN), a probabilistic graphical model represented by a directed acyclic graph, wherein nodes signify the query features and edges represent the dependencies among them. A BN is learned by obtaining the log-likelihood, which is the probability of the data given the network, i.e.,

$$\log L(\Theta \mid Q) = \sum_{e}\sum_{d} \log p(q_{yi} \mid \pi_i, \theta_i), \quad (5)$$

where e is the number of queries in Q, q_{yi} is a feature instance of q_e, π_i is the set of parent nodes of node y_i, and $\theta_i \in \Theta$ is $p(y_i \mid \pi_i)$.

Artificial neural network (ANN) is a computational model based on the concept of human biological neurons. Weights between the so-called neurons, or nodes, are learned based on the query feature inputs; learning is done with the use of gradient descent and backpropagation algorithm. Multilayer perceptron (MLP) is a feedforward variant of ANN.

Support vector machines (SVM) are based on the concept of maximum margin hyperplanes that define a decision boundary between two classes/roles. They benefit from high dimensional feature spaces; high dimensionality means that there are more possible configurations that can be done in the feature space, which can produce more accurate results.

J48 decision trees are one of the most common techniques in data mining that have been successfully used in various fields. It makes use of tree-like graph decisions, selecting query features for every node based on (2). Although prone to overfitting and feature bias, it can achieve high performance with very little effort.

Accordingly, random forest (RF) is an ensemble model based on decision trees. It exploits bagging and random feature selection to create numerous simple trees to vote for the most popular class/role, and is considered to be better in performance and speed than plain decision trees.

Lastly, K-nearest neighbors (KNN) is an unsupervised classifier that groups new queries based on a distance function. Given a new query q, KNN will find the K nearest query data points with respect to q, the most popular class of the nearest neighbors being the inferred role of q.

IV. EXPERIMENTS

A. Benchmark Database

We have adopted the TPC-E benchmark database schema structure and its transactions for all our experiments. TPC-E is a database that simulates the online transaction processing (OLTP) workload of a brokerage firm [18]. Customers, brokers, and the market initiate read/write and read-only transactions against the database, which consists of 33 tables, an overall count of 191 attributes, and 11 standard transactions.

B. Synthetic Data Set Generation

We treat one TPC-E transaction as one role, and we set privileges of a role based on which tables and attributes the transactions are authorized to access, with the corresponding number of times they appear in the transaction. We emphasize, however, that depending on the context, one role may contain several transactions at once.

We employed the transaction database footprint and pseudo-code found in [18]. Each role has a set of specific tables T (and its corresponding attributes A) that it is allowed to access, as well as a set of commands C that it is allowed to execute. We specify the following probabilities for each role: (1) the probability of using a command $c \in C$ given a role r, $p(c|r)$, (2) the probability of projecting a table $t \in T$ given a command c and a role r, $p(P_t|c,r)$, (3) the probability of selecting a table t given a set of projected tables P_T, command c, and role r, $p(S_t|P_T,c,r)$, (4) the probability of projecting an attribute $a \in A$ given a projected table P_t, command c, and role r, $p(P_a|P_t,c,r)$, (5) the probability of selecting an attribute a given a selected table S_t, command c, and role r, $p(S_a|S_t,c,r)$, (6) the probability of including a random string or numeric value $v \in V$ in the selection clause given a command c and role r, $p(v_{sn}|c,r)$, (7) the probability of including a JOIN J given a command c and role r, $p(J|c,r)$, and (8) the probability of including an AND or OR given a command c and role r, $p(AO|c,r)$.

Note that probabilities 2 to 6 are uniformly distributed among a role's corresponding set of tables T, projected tables P_T, projected table P_t, selected table S_t, and list of random strings and numeric V, respectively. Probability 1 is based on a set of commands C (may compose of any combinations of SIUD) that a role/transaction is allowed to issue. For a certain role, probability 7 means that a query can contain a JOIN or not, while probability 8 may contain a combination of AND and OR, AND only, OR only, or none at all [18].

We generate 1,000 queries for each role, labeling each query with its corresponding class, for a total of 11,000 queries or training samples for our normal query log data set. Since we create the models with insider threats in mind, anomalous queries are generated using the same probability distribution as that with normal queries, only with role information negated. That is, if the role annotation for a

 ISBN: 978-1-61208-381-0

certain normal query is class 1, we change it to any other role other than class 1, effectively making it anomalous [4].

C. Results

From this point on, we will refer to features describing the number of elements present in a query as counting features, and those that represent the position of elements as ID features.

The number of features generated largely depends on the number of tables and attributes in the schema. In the case of TPC-E, a total of 277 features were extracted.

Figure 2 shows the average merit based on IG and GR measures. The line indicates the threshold we adopted to get the feature subset for IG and GR. IG produced 12 features while GR produced 144 features. Observing the variables chosen by both measures, IG preferred counting features (those having more possible values), while GR produced a more spread-out merit graph, noticeably preferring pairs of counting by table and ID features while removing string features (S_V and S_L).

We determine the threshold for PCA by plotting the eigenvalues, as shown in Figure 3. Optimal coordinates method produced a subset of 13 features (PCA3), while parallel analysis yielded 63 features (PCA2) [22]. We obtain an additional subset by getting 99% of the variance of the data (113 features, PCA1) for comparison purposes.

For the filter methods, BFS yielded 19 features using correlation as the evaluation measure. Consequently, GA was run for 20 generations with a population size of 20 individuals, crossover rate of 0.6, mutation rate of 0.033 and correlation as evaluation measure. GA chose a total of 68 features, which are noticeably more diverse than the ones chosen by IG, GR, and BFS.

The performances of classifiers in terms of false positive (FP) and false negative (FN) error rates are shown in Table II. False positives are those queries that should have been classified as normal but tagged as abnormal, while false negatives are those that should have been identified as anomalous but were categorized as normal. The Weka toolkit was used in all our experiments and all parameters were left to their default settings [23].

Based on the resulting feature subsets, it can be observed that counting features are vital to obtain a satisfactory classification performance (as seen in the performance of the IG subset). However, they are not enough on their own. PCA came out to be the best feature selection technique among the ones employed—from 277 features, it reduced the data set to 113 features (threshold of 99% variance, PCA1), yielding the overall best average performance. Halving PCA1 to form PCA2 does not have any significant effect on the FP and FN rates, and even when only one-third of PCA2 is retained (PCA3), it still yielded above average performance. This proves that PCA effectively eliminates most of the noise in

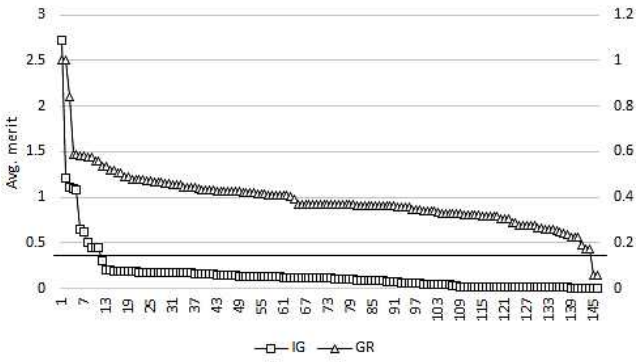

Figure 2. IG and GR values in terms of average merit (y-axis); features (x-axis).

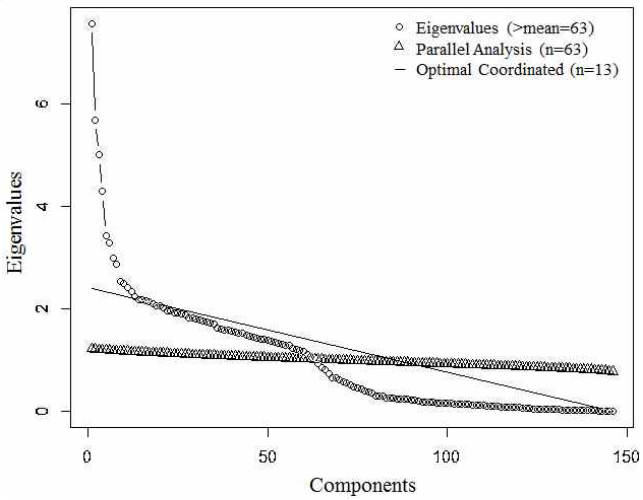

Figure 3. Eigenvalues, parallel analysis, and optimal coordinates plot.

TABLE II. PERFORMANCE OF CLASSIFIERS AND CORRESPONDING NUMBER OF FEATURES IN DECREASING ORDER

No. of features	GR		PCA1		GA		PCA2		BFS		PCA3		IG		Avg.	
	144		113		68		63		19		13		12			
Classifiers	***FP***	***FN***	***FP***	***FN***	***FP***	***FN***	***FP***	***FN***	***FP***	***FN***	***FP***	***FN***	***FP***	***FN***	***FP***	***FN***
NB	0.153	0.008	0.227	0.014	0.244	0.011	0.248	0.012	0.274	0.017	0.224	0.012	0.389	0.024	0.251	0.014
KNN	0.119	0.006	0.141	0.008	0.128	0.006	0.126	0.007	0.140	0.009	0.107	0.004	0.242	0.012	0.143	0.007
MLP	0.143	0.007	0.079	0.003	0.128	0.007	0.082	0.004	0.166	0.008	0.104	0.007	0.232	0.013	0.133	0.007
SVM	0.574	0.051	0.095	0.004	0.449	0.034	0.103	0.005	0.455	0.034	0.103	0.004	0.485	0.039	0.323	0.024
BN	0.064	**0.002**	0.131	0.007	0.083	0.004	0.168	0.010	0.089	0.005	0.160	0.009	0.097	0.004	0.113	0.006
J48	0.067	0.003	0.113	0.005	0.086	0.003	0.118	0.006	0.092	0.004	0.117	0.006	0.091	0.004	0.098	0.0044
RF	**0.055**	0.003	0.075	0.004	0.079	0.003	0.078	0.005	0.086	0.003	0.079	0.004	0.126	0.006	**0.083**	**0.0039**
Avg.	0.168	0.012	**0.123**	**0.006**	0.171	0.010	0.132	0.007	0.186	0.011	0.128	0.007	0.237	0.015		

the data, at the same time reducing its dimensions significantly. GA is second in line to PCA (in terms of performance and number of features used), followed closely by BFS.

Among the classifiers that we have evaluated, graphical models like J48, RF, and BN noticeably performed better than the other algorithms. This may be due to the fact that SQL language syntax has an inherent tree-like structure—a simple attribute that these classifiers are most likely to exploit. Conversely, SVM yielded the worst performance, producing a satisfactory result only with the PCA feature subsets. It is clear that the application of special kernel methods is necessary to obtain acceptable results with SVM. Moreover, NB is the second worst performer, having yielded the highest FP and FN rates for all PCA subsets. In terms of algorithm speed, SVM and MLP are significantly and impractically slower in build time compared to other classifiers, while J48 and RF yielded the fastest detect times.

V. CONCLUSION AND FUTURE WORK

We have shown a clear, side-by-side comparison of data mining feature selection methods and classifiers as applied to the context of database IDS. PCA demonstrated exceptional performance in reducing noise and dimension in the data set, while graphical models, especially RF, came out to be the best suited classifiers for the intrusion detection task, exhibiting very reasonable FP and FN trade-offs and fast detection speed. We hope that these results will provide researchers with a more concrete direction towards designing a more efficient database IDS.

Although we have covered many algorithms in this work, there are still a lot of subjects to explore. Future works will include considering the sensitivity of the tables and attributes in the database. We are also considering on building an ensemble model to be able to develop a stronger classifier out of simple ones.

ACKNOWLEDGMENT

This work is supported by the National Strategic R&D Program for Industrial Technology (10044828), funded by the Ministry of Trade, Industry and Energy (MOTIE)..

REFERENCES

[1] X. Jin and S. L. Osborn, “Architecture for data collection in database intrusion detection systems,” Secure Data Management, VLDB Workshop, vol. 4721, Sept. 2007, pp. 96-107.

[2] C. Mouza et al., “Towards an automatic detection of sensitive information in a database,” Int’l Conf. on Advances in Databases, Knowledge, and Data Applications (DBKDA), Jan. 2010, pp. 247-252.

[3] I. J. Rajput and D. Shrivastava, “Data mining based database intrusion detection system: A survery,” Int’l Journal of Engineering Research and Applications (IJERA), vol. 2, no. 4, July 2012, pp. 1752-1755.

[4] A. Kamra, E. Terzi, and E. Bertino, “Detecting anomalous access patterns in relational databases,” The VLDB Journal, vol. 17, no. 5, Aug. 2008, pp. 1063-1077.

[5] F. S. Rietta, “Application layer intrusion detection for SQL injection,” ACM Southeast Regional Conference, Mar. 2006, pp. 531-536.

[6] A. Adebowale, Idowu S.A, and O. Oluwabukola, “An overview of database centred intrusion detection systems,” Int’l Journal of Eng’g. and Advanced Technology, vol. 3, no. 2, Dec. 2013, pp. 273-275.

[7] Y. Hu and B. Panda, “A data mining approach for database intrusion detection,” ACM Symposium on Applied Computing, 2004, pp. 711-716.

[8] A. Srivastava, S. Sural, and A. K. Majumdar, “Database intrusion detection using weighted sequence mining,” Journal of Computers, vol. 1, no. 4, July 2006, pp. 8-17.

[9] S. Hashemi, Y. Yang, D. Zabihzadeh, and M. Kangavari, “Detecting intrusion transactions in databases using data item dependencies and anomaly analysis,” Expert Systems, vol. 25, no. 5, Oct. 2008, pp. 460-473.

[10] D. Barbara, R. Goel, and S. Jajodia, “Mining malicious corruption of data with hidden Markov models,” Research Directions in Data and Applications Security, Int’l Federation for Information Processing (IFIP), vol. 128, 2003, pp. 175-189.

[11] P. Ramasubramanian and A. Kannan, “A genetic algorithm based neural network short-term forecasting framework for database intrusion prediction system,” Soft Computing, vol. 10, no. 8, June 2006, pp. 699-714.

[12] C. Pinzon, A. Herrero, J. F. De Paz, E. Corchado, and J. Bajo, “CBRid4SQL: A CBR intrusion detector for SQL injection attacks,” Hybrid Artificial Intelligence Systems, vol. 6077, 2010, pp. 510-519.

[13] R. M. Elbasiony, E. A. Sallam, T. E. Eltobely, and M. M. Fahmy, “A hybrid network intrusion detection framework based on random forests and weighted k-means,” Ain Shams Eng’g. Journal, vol. 4, no. 4, Dec. 2013, pp. 753-762.

[14] E. Bertino, E. Terzi, A. Kamra, and A. Vakali, “Intrusion detection in RBAC-administered databases,” Computer Security Applications Conf. (ACSAC) , Dec. 2005, pp. 170-182.

[15] F. Valeur, D. Mutz, and G. Vigna, “A learning-based approach to the detection of SQL attacks,” Detection of Intrusions and Malware, and Vulnerability Assessment, vol. 3548, 2005, pp. 123-140.

[16] S. X. Wu and W. Banzhaf, “The use of computational intelligence in intrusion detection systems: A review,” Technical Report, Dept. of Computer Science, Memorial University of Newfoundland, Nov. 2008.

[17] A. Zainal, M. A. Maarof, and S. M. Shamsuddin, “Feature selection using rough set in intrusion detection,” IEEE Region 10 Conf. (TENCON) , Nov. 2006, pp. 1-4.

[18] Transaction Processing Performance Council (TPC), TPC benchmark E, Standard specification, version 1.13.0, 2014.

[19] H. A. Nguyen and D. Choi, “Application of data mining to network intrusion detection: Classifier selection model,” Challenges for Next Generation Network Operations and Service Management, vol. 5297, 2008, pp. 399-408.

[20] L. M. Lima de Campos, R. C. Limao de Oliveira, and M. Roisenberg, “Network intrusion detection system using data mining,” Eng’g. Applications of Neural Networks, Communications in Computer and Information Science, vol. 311, 2012, pp. 104-113.

[21] J. Zhang and M. Zulkernine, “Anomaly based network intrusion detection with unsupervised outlier detection,” IEEE Int’l Conf. on Communications, 2006, pp. 2388-2393.

[22] S. B. Franklin, D. J. Gibson, P. A. Robertson, J. T. Pohlmann, and J. S. Fralish, “Parallel analysis: A method for determining significant principal components,” Journal of Vegetation Science, vol. 6, no. 1, 1995, pp. 99-106.

[23] M. Hall et al., “The WEKA data mining software: An update,” SIGKDD Explorations, vol. 11, no. 1, 2009, pp. 10-18.

 ISBN: 978-1-61208-381-0