Journal of Ambient Intelligence and Humanized Computing
https://doi.org/10.1007/s12652-020-02231-4

ORIGINAL RESEARCH

CID: a novel clustering-based database intrusion detection algorithm

Mohamad Reza Keyvanpour[1] · Mehrnoush Barani Shirzad[2] · Samaneh Mehmandoost[1]

Received: 15 November 2019 / Accepted: 13 June 2020

Abstract
At the same time with the increase in the data volume, attacks against the database are also rising, therefore information security and confidentiality became a critical challenge. One promised solution against malicious attacks is the *intrusion detection* system. In this paper, anomaly detection concept is used to propose a method for distinguishing between normal and abnormal activities. For this purpose, a new density-based clustering intrusion detection (CID) method is proposed which clusters queries based on a similarity measure and labels them as normal or intrusion. The experiments are conducted on two standard datasets including TPC-C and TPC-E. The results show proposed model outperforms state-of-the-art algorithms as baselines in terms of FN, FP, Precision, Recall and F-score measures.

Keywords Intrusion · Intrusion detection · Database · Anomaly detection · Outlier detection · Density-based clustering

1 Introduction

Database systems which have been designed to manage the data in computer systems are dealing with safety challenge. A database management system in order to provide access to data, responses commands from general or specified application programs (Darwen 2009). Any unauthorized attempt to access, manipulate, modify and destroy information or to use a computer remotely to spam, hack or modify other computers in computer science is called an intrusion (Dua et al. 2016). To overcome intrusions, intrusion detection systems have been developed which raise an alarm when detect an intrusive activity. The intrusion detection is fulfilled in three levels: network, operating system and database (Santos et al. 2014). Since some attacks and intrusions in the database are not detectable in the network and operating systems levels and considering the existence of inner attacker who has access to data, several studies have focused on intrusion detection in relational databases.

There are two general approaches for intrusion detection, called anomaly detection and misuse detection (Dua et al. 2016). Misuse detection strategy detects the predetermined intrusions efficiently, while fail to find new and unknown ones. This strategy, leads to large false negative rate. In the anomaly detection approach, the normal activities are modeled and any activity with a difference behavior is recognized as an intrusion. Data mining technologies with the aim of detecting patterns for normal and abnormal activity have been applied to find abnormal activity or intrusions (Kamber 2011). Both supervised and unsupervised learning have been utilized for current task (Gogoi et al. 2011; Aggarwal 2013). Supervised learning deals with labeled data, whereas unsupervised learning works with unlabeled data. As a result of lack of labeled data in this domain unsupervised models such as clustering algorithms based on density perform well for intrusion detection (Du 2010).

In this research, anomaly detection concept is used to present a novel method for distinguishing between normal and abnormal activities in database level. For this purpose, we propose a new density-based clustering method called CID (Clustering-based Intrusion Detection) which clusters queries based on a similarity measure and label them. CID works in two parts including training and test phase. In training phase, which works on training data first features are extracted for input queries, then clustering is modeled. In

✉ Mohamad Reza Keyvanpour
keyvanpour@alzahra.ac.ir

Mehrnoush Barani Shirzad
Mehrshirzad@gmail.com

Samaneh Mehmandoost
samaneh.mehmandoost@yahoo.com

1 Department of Computer Engineering, Faculty of Engineering, Alzahra University, Tehran, Iran

2 Data Mining Laboratory, Department of Computer Engineering, Faculty of Engineering, Alzahra University, Tehran, Iran

clustering, first the queries distances are computed, then the Radius Neighbor is calculated. After that, the number of points for the purpose of clustering is determined. At the end the training queries by density-based clustering are labeled. Test phase includes two parts feature extraction and CID model, which applied on test data. Our method contributes to the issue in two parts:

1. Computing the distances between features, a new distance metric inspired by similarity measure is introduced. Since the queries are in form of strings, we developed a metric to measure the distances of queries' string.
2. Training query clustering, in which a density-based query clustering for Labeling the Training Queries is proposed.

To the best of our knowledge, there is no other study in domain of database intrusion detection which consider string based query distance metric. The CID is tested on TPC-C and TPC-E datasets. The results of CID are compared to articles using mentioned datasets. The results show that in comparison with three algorithms including: RBDDRM (Ronao et al. 2015) and WRBDDRM (Rao and Singh 2017), and RF (Ronao and Cho 2015) CID outperforms baselines in terms of true positive, false negative, Recall and F-score. Proposed model performs competitively in terms of other metrics.

The rest of the paper is organized as follows: Sect. 2 presents related work. In Sect. 3, the concepts and definitions are reviewed, and in Sect. 4 proposed intrusion detection system which is based on clustering method is introduced. Section 5 indicates the experiments. Section 6 concludes the paper and outlines the future directions.

2 Related work

A plenty of studies focused on utilizing data mining methods to implement an anomaly-based IDS. In Hu and Panda (2004) and Srivastava and Sural (2006) authors applied classification to mine dependencies among data in a relational database system. In Yu et al. (2018) authors introduced CTSIF_SVMs in which, SVMs is applied as their Classification model for Two-Side cross domain collaborative filtering problem. Inspired by this study also in Yu et al. (2019) proposed TSEUIF for a cross-domain collaborative filtering problem. In Hu and Panda (2004) dependency is though as the access correlations among data items whilst in Srivastava and Sural (2006) authors considered sensitivity of the attributes as weights, to find malicious modifications. In Doroudian and Shahriari (2014) another rule mining method has been proposed. Normal behavior is learned through finding patterns. They claim that hybridation of transaction and user task levels, bring advantages to their model. Rule mining is investigated by Moradi and Keyvanpour (2015).

Hidden Markov models (HMM) applied in Barbara et al. (2003) to detect the normal changes in database's transactions. Other works (Ramasubramanian and Kannan 2004) used of artificial neural networks (ANN). In Ramasubramanian and Kannan (2004) a framework which combined statistical anomaly detection and rule based misuse detection proposed to determine misusers. In Ramasubramanian and Kannan (2006) a framework for anomaly detection by a neuro-genetic forecasting model to predict attack by capturing previous observations had been proposed.

Support vector machines and multilayer perceptrons are used in Pinzón et al. (2010) which classify new query in order to identify SQL injection attacks. In Ronao and Cho (2015) and Kamra et al. (2008) applied standard of role-based access control (RBAC). In Kamra et al. (2008) naïve Bayes classifier is utilized to detect anomalous query access. In Ronao and Cho (2015) authors used random forest (RF) algorithm for anomaly detection.

Study of (Bockermann et al. 2009) proposed to apply tree kernels to model SQL commands to detect malicious behavior. In Choi and Cho (2017) a combination of reinforcement learning and evolutionary learning have been proposed which claimed that is be resistant to insider misuse.

In Bu and Cho (2017) a hybrid system of convolutional neural network and genetic algorithm is proposed, in which CNN classified the queries through learning normal behaviors and genetic algorithm find new rules to detect abnormal behaviors. In Felemban et al. (2018) a data Partitioning-based Intrusion Management System has been proposed. Authors defined the concept of Intrusion Boundary (IB) and formulate it as an optimization problem and apply a Mixed Integer Non-Linear Programming model as solution.

In Subudhi and Panigrahi (2019) OPTICS clustering on the transaction attributes for building user behavioral profiles is applied. In Wee and Nayak (2019) reinforcement learning is applied for intrusion detection. In Sallam and Bertino (2019a) proposed anomaly detection techniques designed to detect data aggregation and attempts to track data updates.

Other works like (Ronao and Cho 2014) made comparison between several data mining based algorithms for anomaly detections. In Sallam and Bertino (2019b) reviewed the prominent techniques and systems for anomaly detection in database systems. Study of (Pourkazemi and Keyvanpour 2017) proposed a method for community detection.

3 Concepts and definitions

An intrusion detection system (IDS) is a system which analyzes the database system and user's activity, assesses the systems and data integrity, finds intrusion's pattern, reacts

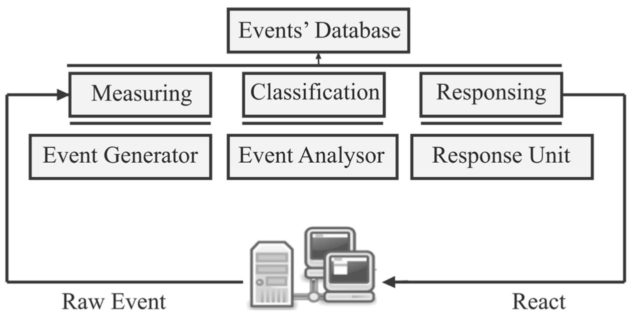

Fig. 1 Intrusion detection system architecture

to intrusions and reports its detections as outputs. IDSs are developed to overcome the short come of network and operating system in intrusion detection and their lack of capacity to detect inner intrusions (Corona et al. 2013). The act of intrusion detection is divided into three steps including measurement, classification and response. Figure 1 shows the IDS architecture.

An IDS is categorized according to: the detection approach, react to intrusion, the location of IDS, the time and architecture (Mordai 2014). Considering detection approach, IDS categorized into misuse/signature-based detection, and anomaly/profile detection. From the react to intrusion point of view, IDSs classified into active and passive systems. Based on IDS location, they are categorized into Network-based and Host-based systems. Considering time, IDS are continuously or discretely. Based on the architecture IDS are central or distributed. Two types of attacks are defined in a database (Corona et al. 2013) inner and outer attacks. Attacks can be classified into Inference, passive, active, and SQL injection (Corona et al. 2013). Several works studied system security such as (Zandian and Keyvanpour 2017) which assessed fraud detection models.

Anomaly detection is a widely used approach in detecting intrusion. Clustering is a solution to anomaly based intrusion detection. Clustering is divided into four types including; connectivity based, centroid based, distribution based and density based clustering (Pirrone et al. 2018).

In density based clustering points which are not belonging to any of clusters considered as noises (outlier or anomaly). In this clustering, dense areas are separated from each other by sparser areas in data space (Ester et al. 1996). In following definitions for anomaly based intrusion detection by density base clustering issue is provided.

Definition 1: (Distance) the distance between two query points q_i and q_j in a database D is denoted by DA(q_i,q_j) is the result of applying a distance function:

$$DA\left(q_i, q_j\right) : \mathrm{D} \times \mathrm{D} \rightarrow [0, \infty)$$

where [0, ∞) indicate that the result is a non-negative real number. For all members of D, the following conditions are satisfied:

1. $DA\left(q_i, q_j\right) \geq 0$
2. $DA\left(q_i, q_j\right) = 0 \leftrightarrow q_i = q_j$
3. $DA\left(q_i, q_j\right) = DA\left(q_j, q_i\right)$

First indicated the distance of two instances is a positive real number. Second expresses the instances whiteout distance are the same. Third one demonstrated the symmetric property of distance in which the distance between two instances is equal in every direction.

Generally, in clustering instances tends to make clusters with their nearest neighbors. Neighbors are determined according to their distances from each other.

Definition 2: (Eps-neighborhood of a query) The Eps-neighborhood of a query q, shown by $\mathrm{N}_{\mathrm{Eps}}(q_i)$, is defined as (Corona et al. 2013):

$$N_{Eps}(q_i) = \{q_i \in D | dist\left(q_j, q_i\right) \leq E_{ps}\}$$

For density based clustering for each q_i instance in a cluster C, there exists n number samples in Esp-neighborhood of q which belong to the same cluster C. The number of instances in Esp-neighborhood of q are at list MinPts. This parameter should set for clustering purpose.

Definition 3: (directly density-reachable) A query q_i is directly density-reachable from a point q_j wrt. Eps, MinPts if:

1. $q_i \in N_{Eps}(q)$
2. $|N_{Eps}(q)| \geq MinPts(\text{corepointcondition})$

Directly density-reachable concept enjoys the symmetric properties for core pairs of instances. Core instance refers to instances which are not on the border of clusters.

Definition 4: (cluster) for D database a cluster C wrt. Eps and MinPts is a non-empty subset of D satisfying the following conditions:

1. $\forall q_i, q_j$: if $q_j \in C$ and q_j is density-reachable from q_i wrt. Eps and MinPts, then $q_i \in C$.
2. $\forall q_i, q_j$: q_i is density-connected to q_j wrt. Eps and MinPts.

A cluster in density based clustering addresses a set of instances which are density connected to each other.

Definition 5: (non-overlapping clusters) this refers to clustering in which clusters do not share any member, are defined as follows (Grossi et al. 2015),

$$\forall C_i \in C : C_i \subseteq C, \cup \mathrm{C} = \mathrm{D}, \cap \mathrm{C} = \emptyset.$$

Clusters can share their instances between each other or not. In other words, an instance can be a member of more than one clusters. Here we considered no- overlapping cluster.

Definition 6: (Intrusion) Let Ct … Ck be the clusters of the database D wrt. parameters Epsi and MinPts i, i = 1 … k. Intrusion is defined as the set of queries in the database D not belonging to any cluster Ci, i.e.

$$\text{Intrusion} = \{q \in D | \forall i : q \in C_i\}$$

Noise or outlier or intrusion can be defined as instances which are not member of any clusters (Ester et al. 1996). Clusters contain the pattern of major queries, whilst the intrusions deny to follow the common trend.

4 CID: clustering-based intrusion detection method

Input to this system is an unlabeled relational database query. First in training phase, features are extracted for each query. Learning contains training clusters, in which feature vectors are received from previous step and a clustering model is learned. In test phase, first features are extracted for each new query, then the label is determined by learned CID model considering its similarity with training query. Figure 2 shows the scheme of CID.

4.1 Training phase

4.1.1 Feature extraction

This phase consists of feature extraction, in which for all input queries features vectors are extracted. Ten relational database queries including select, select into, insert, insert into, update, delete, alter, drop (Kamra et al. 2008) is considered in this paper. These commands belong to seven command type.

Each query has several features such as command, attributes, read_tables, write_tables, values, where_conditions, having_conditions, order by, group by. Command expresses each command type. Attribute relates to fields (columns) used in a query. Read_table include the name of tables have been read in a query. Write_table include the name of tables have been written in a query. Values depict the value assigned to each field. Where_condition include the conditionals expressions in where command while having_condition include the conditionals expressions in having command. Order by shows the attribute in order by command which determines the outputs are indicated by which attributes and group by indicates the attribute in group by command which determine the outputs are grouped and indicated by which attributes. Each of aforementioned commands possesses several of these features. Table 1 illustrated the commands and their possible features. Some commands do not have any features, which are shown with dash in the table.

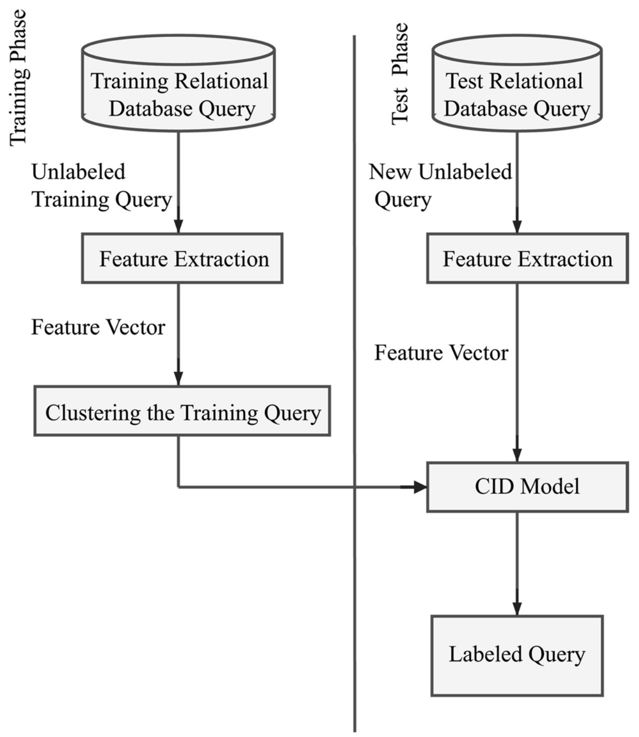

Fig. 2 Clustering-based intrusion detection scheme

The input to this feature extraction algorithm is an array of strings including training queries. The output is feature vectors of queries. For queries have not some of the above feature null is set as feature value. For all queries query is split to words and the first word for command is checked. Then in order to string normalization, unnecessary characters and words are eliminated and important features according to relevant command are extracted.

4.1.2 Training query clustering

This stage starts with calculating the distance between queries. Distance is defined inspiring by the similarity measure. In Shirzad and Keyvanpour (2017) distance is applied to find similarity. Here, the result is a symmetric matrix applying triangular matrix for saving in memory. Then, eps is calculated which is the threshold for radius neighbor of each query (Ester et al. 1996). Next the minPts which determines the minimum number of points for constructing a cluster is computed. The procedure is followed by clustering based on density. Figure 3 illustrates four components of clustering phase.

Table 1 Commands' features

	Command	Attribute	Read table	Write table	Values	Where condition	Having condition	Order by	Group by
Select	1	✓	✓	–	–	✓	✓	✓	✓
Select into	1	✓	✓	✓		✓	✓	✓	✓
Insert into	2	✓		✓	✓	–	–	–	–
Insert into table	2	✓	✓	✓		✓	✓	✓	✓
Update	3	–	–	✓	✓	✓	–	–	–
Delete	4	–	–	✓	–	✓	–	–	–
Alter	5	✓	–	✓	–	–	–	–	–
Drop/truncate table	6	–	–	✓	–	–	–	–	–
Create table	7	✓	–	✓	–	–	–	–	–
Create table as … select	7	✓	✓	✓	–	✓	✓	✓	✓

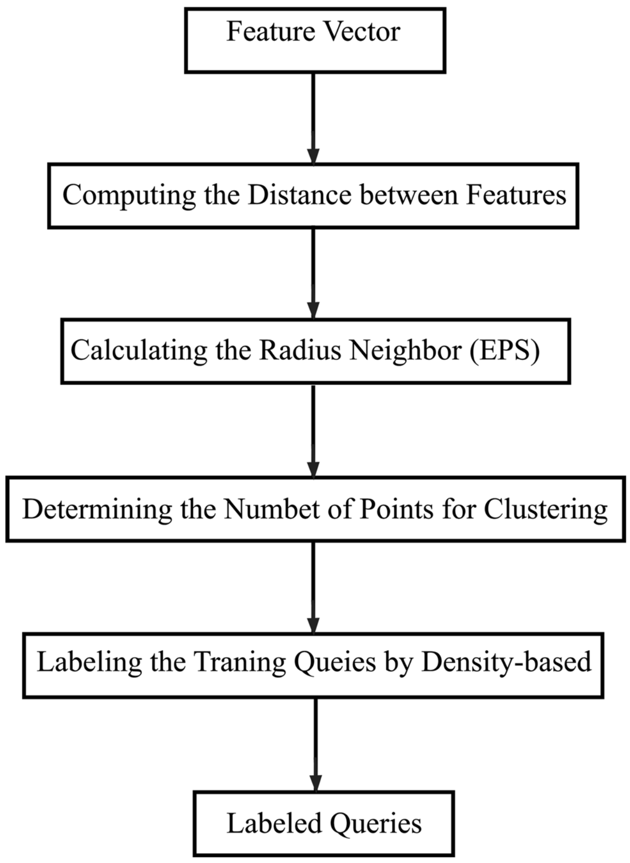
Feature Vector
Computing the Distance between Features
Calculating the Radius Neighbor (EPS)
Determining the Numbet of Points for Clustering
Labeling the Traning Queies by Density-based
Labeled Queries

Fig. 3 Clustering phase

a. Computing the distances between features

Since here the queries are not digital data and are string, for computing the distances the metrics which determine the similarity or difference between two strings are required (Gomaa and Fahmy 2013). In this research the fields' name and tables' name are saved as string and compared with each other. Features such as attributes, read_tables, write_tables and values each word shows a value. In order to compute distance of each of these features of two queries, the similar values are counted. Whilst for features such as where condition and having_conditions, since the value of them are sentences other measure is needed. To this end, a string metric called Levenshtein distance is applied. Levenshtein distance between of two strings is computed by summation of the minimum number of operations such as insert, delete and substitution of letters applied in order to transform one string to the other string (Miller et al. 2009) Levenshtein is defined as follows.

$$Lev_{a,b}(i,j)\begin{cases} \max(i,j) if \min(i,j)=0 \\ min\begin{cases} Lev_{a,b}(i-1,j)+1 \\ Lev_{a,b}(i,j-1)+1 \\ Lev_{a,b}(i-1,j-1)+I(a!=b) \end{cases} & otherwise. \end{cases} \tag{1}$$

where $\mathrm{Lev}_{a,b}$ (i,j) is the distance between word i of the first string and word j of the second string. I(a! = b) is an indicator function which equals to zero if a_i and b_j are equal otherwise set to 1. In above equation first expression in min relates to delete, second relate to insert operation and the third one relates to equality or inequality of letters (Miller et al. 2009). Herein, two example for Levenshtein distance between two words is indicated (Rani and Sing 2018).

Levenshtein (Right, fight) = 1 (substitution operation of 'f' for 'r').

Levenshtein (book, books) = 1 (insert operation of s).

In following some examples for Levenshtein distance method on queries are depicted. For followings three queries A, B and C, Levenshtein are computed.

Query A = select 1, stock.rowid, i_price, i_name, i_data, s_dist_%02d, s_data, s_quantity from item, stock where i_id = :11 and s_w_id = :31 and s_i_id = i_id.

Query B = select 10, stock.rowid, i_price, i_name, i_data, s_dist_%02d, s_data, s_quantity from item, stock where i_id = :22 and s_w_id = :3 and s_i_id = i_id.

Query C = select 1, stock.rowid, i_price, i_name, i_data, s_dist_%02d, s_data, s_quantity from item, stock where i_id = :11 and s_w_id = :31 and s_i_id = i_id UNION ALL.

Levenshtein (A, B) = 4 indicates the number of operations including insert, substitution and delete, which are needed to transform query A to query B. Levenshtein (A, C) = 10 shows the number of required insert operations for converting query A to query C including insertion of characters and spaces. Differences in B and C queries from query A are also highlighted in example queries. Obviously, the Levenshtein distance for two identical queries is equal to zero. The distance between two queries is achieved by summation of difference of all features. In distance computation the command feature is not considered. In following the pseudocode for distance algorithm is shown. Input is array of feature vectors of queries, and output is an upper triangular matrix including distances between queries.

```
Algorithm1 : compute the distance of queries
1   counter = 0
2   FOR i = 1 TO (number of queries-1) DO
3   //constructing an upper triangular matrix
4       FOR j = i+1 TO number of queries DO
5           split features attributes, read_tables,
6   write_tables, values into words
            compare words of every relevant
7   feature in Q[i] and Q[j], and find number of
            uncommon words
            use levenshteinDistance function for
8   features where_condition and
9   having_condition ( levenshteinDistance
10  (where_condition[i], where_condition[j]),
11   levenshteinDistance(having_condition[i],
12  having_condition[j]) )
13          TotalDistance = sum of the number of
    all uncommon words and levenshtein
    distances
            DA[counter] = TotalDistance
            counter++
        ENDFOR
    ENDFOR
    EN
```

b. Calculating the radius neighbor (EPS)

Study of (Ester et al. 1996) showed the distance function determines the clustering shape. Here a new 2-dimensional distance function is introduced. For calculating the radius neighbor, first the average and variance of distance for each query against others is computed, then the absolute of their differences is set as the query's radius neighbor. The average of distances from one query indicates the mean distance of other queries from it. Standard deviation (Bland and Altman 1996) is shown by δ and is a distribution identification which expresses the data's distance from mean value in average. Following equation defines the standard deviation.

$$\delta = \sqrt{\frac{1}{N}\left[(x_1 - \mu)^2 + (x_2 - \mu)^2 + \cdots + \left(x_2 - \mu\right)^2\right]}$$
$$\textit{where}\ \mu = \frac{1}{N}(x_1 + \cdots + x_N) \tag{2}$$

where x_i here is the queries' distance from the considered query, N is the number of queries, μ is the average distance. If the standard deviation value is near to zero it means the data are near to average and the has small distribution, whilst large variance expresses the considerable data distributions. Standard deviation equals to square of variance. Herein, the absolute difference between average and standard deviation is considered as radius neighbor. Following formula indicates the radius neighbor.

$$\text{epsi} = |\mu i - \delta i| \tag{3}$$

where eps_i is radius neighbor related to ith query, μi is average and δ_i is the standard deviation from i query. Algorithm 2 shows the process of computing EPS. Input is array of distances between queries (DA[]) and output is an array of queries' EPS.

Algorithm 2: compute EPS

```
int sum = 0;
  FOR i = 1 TO number of queries DO
    FOR j = 1 TO number of queries DO
      int m;
      IF (i == j) THEN
        continue;
      IF (j < i) THEN
        m = (j*n)-(((1+j)*j)/2)+(i-(j+1));
      ELSE
        m = (i*n)-(((1+i)*i)/2)+(j-(i+1));
       sum += DA[m];  //sum the distances
between query i and all the other queries
    ENDFOR
        compute average of distances
        compute standard deviation of
distances
        eps[i] = absolute value of average-
deviation;
        sum = 0;
  ENDFOR
END
```

Radius neighbor is computed for each query, this lead to have difference density for clusters. In other words, radius neighbor for each query depends on points' distance from it. This method leads to change in density based on query which made the algorithm dynamic.

c. Determining the number of points for clustering (minPts)

The minimum number of points is set to find all instances member of the same cluster (Ester et al. 1996). Queries' distances are applied here to gain minPts. To this aim, first for each query the number of queries which their distance values are lower than or equal to eps is gained. Then average is computed. The following equation shows the minPts.

$$\mathrm{minPts} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\mathrm{Distance}_{i,j}}{n}\ \text{Such that } \mathrm{Distance}_{i,j} \le \mathrm{eps}_i \tag{4}$$

this value is identical for all queries.

d. Labeling the training queries by density-based clustering

The main strategy in clustering based on query is to model the clusters according to dense area in space to separate them from sparse area. Clustering methods are able to clustering arbitrary [7]. In this research the center of a cluster is selected considering the distance matrix of queries opting for queries near to each other. Then, the number of near queries for each query is achieved. After that, the query's points are sorted in descending order according to their distances. So that, there exists a set of points can be chosen as cluster centers. First the number of each label is assigned as its label. In Hassanzadeh and Keyvanpour (2013) sequence labeling is addressed. First point which has the largest number of points with lowest distances is considered as cluster center. Then, the list of query candidate for cluster center are investigated. If the query's distance from the first selected center cluster is lower than or equal to eps, then it gains the label of first query. Otherwise, it keeps its previous label. After tracing the list and labeling its members, the queries which own the shortest distance with center are assessed. If their distance with current point is lower than or equal to eps then receive the current query's label, else keeps its own label. This process continues for all queries; in cases which query had been labeled before its label would not change anymore. The clusters are constructed as a result of aforementioned steps. The number of minimum points for cluster should satisfy the minPts. A query is labeled as intrusion in following conditions.

$$\mathrm{Intrusion}(q_i) = \{\text{if } q_i \in C_i \text{ and } |C_i| \le \mathrm{MinPts}, \text{or } q_i \notin C\}$$

If the number of a cluster's member is lower than minPts, all quires belong to that cluster are considered as outlier or intrusion. A query is considered as intrusion if it is not a member of any cluster, its label is unique. Input to this algorithm is array of unlabeled queries, array of EPSes, minPts and output is the labeled queries as normal or intrusion.

Algorithm 3: density based clustering (PMMD)

```
FOR i = 1 TO number of queries DO
  find the nearest query to query[i]
ENDFOR
keep the queries that are nearest to others in an array and sort them decently (NQ[])
FOR i = 1 TO number of queries DO
  label[i] = i; // default label of each query
  count_label[i] = 1; //number of each label is 1
ENDFOR
FOR i = 1 TO number of nearest queries in sorted array DO
  FOR j = i+1 TO number of nearest queries in sorted array DO
    IF (NQ[j] is not visited)
      IF(dist(i,j) <= eps[NQ[i])
          count_label[j]--;
          label[j] = label[i];
          count_label[i]]++;
          visited[j] = true;
      ENDIF
    ENDIF
  ENDFOR
  FOR x = 1 TO number of queries that NQ[i] is nearest to them DO
    IF (distance(x,i) <= eps[i] and order of x is not higher than i) THEN
      count_label[x]--;
      label[x] = label[i];
      count_label[i]++;
    ENDIF
  ENDFOR
ENDFOR
FOR i = 1 TO number of queries DO
  IF (count_label[label[i]] < minPts) THEN
    label[i] = a number bigger than the number of training queries; //to show that a query is not normal
  ENDIF
ENDFOR
END
```

4.2 Test phase

Test phase is the last part of every learning based model (James 2013). This step includes the new data which are different from training data are applied in order to evaluate the final model. Herein, this part works in two segments including feature extraction in which queries are transformed into vectors of features and CID in which a label assigned to query and the intrusion detection is happened.

4.2.1 Feature extraction

The input of this part is a new query and the output is feature vector. To label new instance, features introduced in (Kamra et al. 2008) are extracted for this new query. Nine features including command, attributes, read_tables, write_tables, values, where_conditions, having_conditions, order by, group by similar to training feature extraction section are extracted for test data. The extracted feature vectors are delivered to test phase as input.

4.2.2 CID model

The aim of density based clustering is to separate the dense area of instances from the spars space. In the clustering based intrusion detection issue the instances which are not belong to any clusters are considered as intrusion query. In this part the new query represented by a feature vector is received to CID learned from training phase and the proper label is assigned to it. Hans-Peter et al. (2011) defined the dense cluster as areas of higher density than the other area of the data space. Therefore, in order to assign a label to a new test data which is an unlabeled query, the closest query to the test query is found then assess if their distance is lower than or equal to the threshold i.e. eps value (Ester et al. 1996). If the condition is satisfied, the test query obtains the nearest query's label otherwise it is labeled as intrusion. This relation is formulated as following formula:

$$\text{Label}_{\text{new-query}} = \begin{cases} \text{Label}_{\text{nearest-query}} & \text{if } D(\text{new query.nearest query}) \leq \text{eps}_{\text{nearest-query}} \\ & \text{intrusive else.} \end{cases} \tag{5}$$

where $\text{Label}_{\text{new-query}}$ shows the test query, D(new-query-nearest-query) indicates the distance between test query and the nearest query to it. $\text{eps}_{\text{nearest-query}}$ expresses the radius

Table 2 Databases' attributes

Database		No. fields	Min no. fields' table	Max no. fields' table	Data type	No. primary keys	No. foreign keys	No. constraint	Referential integrity rule
TPC-E	33	188	2	24	UID, CHAR, NUM, DATE, BOOL, LOB	33	50	22	Y
TPC-C	9	92	3	21	UID, CHAR, NUM, DATE	8	9	0	N

neighbor for nearest query. Therefore, here the issue is transformed to finding nearest query to the test query. For this purpose, the distances are computed applying distance algorithm and finding the minimum value representing the nearest query. In following algorithm 4 is presented the test process.

```
Algorithm 4: Labeling Test Query
    find the minimum distance between new query and training queries
    keep the distance(d) and label of nearest query
        IF (d <= eps (nearest query)) THEN
            Label [new query] = label [nearest query]
        ELSE
            Label [new query] = "intrusive"
        ENDIF
END
```

5 Experiments

5.1 Experimental setting

Datasets. The experiments are conducted on two databases from TPC benchmark similar to other valid studies (Rao and Singh 2017; Ronao and Cho 2014; Kundu et al. 2010). This benchmark consists of several datasets.[1] We use TPC-C (online transaction processing benchmark)[2] and TPC-E simulates the online transaction processing (OLTP)[3] database. Transactions of both datasets are standard and have ACID properties. Table 2 lists the features for each dataset. We have used 3120 normal queries and 220 abnormal queries on TPC-E dataset. In order to test the algorithm on TPC-C dataset, 622 normal queries and 158 abnormal queries have been employed.

Evaluation Measures. Popular metrics applied for intrusion detection assessments include: true negative, false positive, false negative, true positive, precision, recall and F-score. True positive rate indicates the number of intrusions detected correctly. True negative rate shows the rate of normal activities that determined as normal. False positive rate depicts the number of normal activities that incorrectly indicates as intrusion. False negative rate measures the number of intrusions that detected as normal (Zhang et al. 2008). Precision and Recall are defined based on these measures. Precision addresses the percentage of relevant results, whilst Recall indicates the percentage of total relevant results correctly detected. Following equation formulates the Precisions.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6}$$

Following equation formulates the Recall measure.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{7}$$

To balance these metric F-score (Sasaki 2007) has been proposed. F-score is shown in following equation.

$$\text{F} - \text{score} = \left(\frac{2{*}\text{Precision}{*}\text{Recall}}{\text{Precision} + \text{Recall}} \right) \tag{8}$$

These measures are widely used, studies such as (Ronao and Cho 2014; Pourkazemi and Keyvanpour 2017; Hassanzadeh and Keyvanpour 2013; Kundu et al. 2010) applied these metrics to evaluate their work.

5.2 Experimental Results

5.2.1 Experiments 1. Test on TPC-C dataset

For TPC-C dataset we compare CID with WRBDDRM (Rao and Singh 2017) and RBDDM (Ronao et al. 2015) methods. In the study by Rao and Singh (2017) 600 normal query and 100 intrusive query had been generated. The results reported here are average values. The results indicated in

[1] Available at: https://www.tpc.org.

[2] TPC, TPC Benchmark C, Standard Specification, Ver. 5.1, available at: https://www.tpc.org/ tpcc, July 11, 2016.

[3] Transaction Processing Performance Council (TPC): TPC benchmark E, Standard specification, Version 1.13.0, 2014.

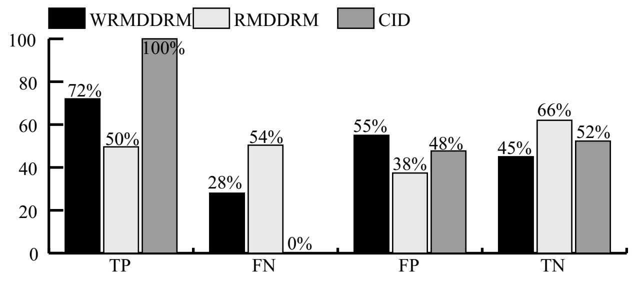

Fig. 4 Results on TPC-C dataset based on TP, FN, FP, TN

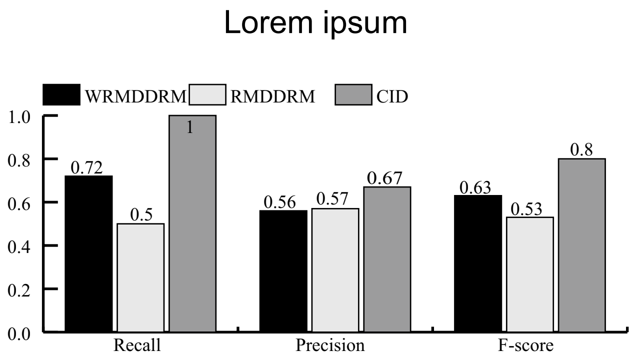

Fig. 5 Results on TPC-C database in terms of precision, recall and F-score

Fig. 4 express that CID outperforms the baselines in terms of TP and TN. Based on TP metric CID gained 100% and 28% higher than what WRBDDRM obtained which illustrates its power in finding all the normal queries. RBDDRM has been gained the least amount for this metric. Also, according to TN, RBDDRM reaches the highest value, it shows 62%. CID gained the second highest amount and shows a 7% improvement against WRBDDRM. Based on FN our method gained zero value, which indicates the robustness of our method, which have not detect any intrusions as normal in average and outperforms the two baselines. Whilst according to FP measure CID find a rate of about 48% of normal queries as intrusions. In comparison to WRBDDRM, CID experienced a decrease of about 8% whilst it reaches a higher value rather than RBDDRM gained. On the whole proposed method did better on TPC-C database against WRBDDRM in all metrics, and in terms of TP and FN outperforms RBDDRM.

Figure 5 depicts the result of CID and baselines WRBDDRM (Rao and Singh 2017) and RBDDRM (Ronao et al. 2015) on TPC-C Database in Terms of Precision, Recall and F-score. As it can be seen, based on recall CID obtained 1 which is the highest possible amount. It reaches an improvement of 28% against WRBDDRM and 50% against RBDDRM. This result rooted in TP values, which CID outperforms baselines in terms of TP values. Turning to precision value CID emphasizes its merit with a 11% and

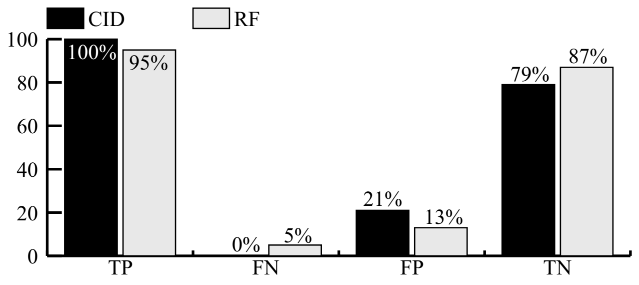

Fig. 6 Results on TPC-E dataset based on TP, FN, FP, TN

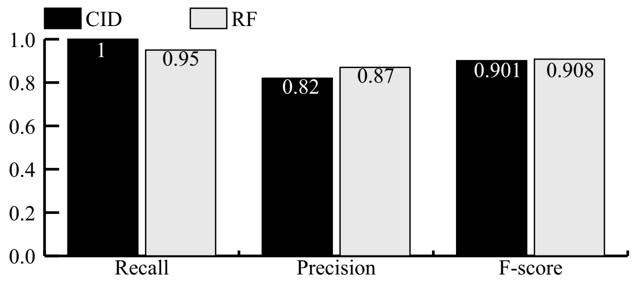

Fig. 7 Results on TPC-E database in terms of precision, recall and F-score

10% improvement in comparison to baselines. According F-score CID witnessed a significant improvement of 27% and 37% against the basedlines and gained the highest value.

On the whole CID outperform WRBDDRM and RBDDRM in terms of several metrics. Also, it gained the highest possible value based on true positive and Precision. The improvement in the result of CID can be the result of applying clustering which is not sensitive to the uncorrected labeling. Morevoer, in suggested algorithm the conditional expressions in queries are considered, whilst the WRBDDRM only considered the attributes and tables. Two baselines depend on support and confidence for each role to create rules. RBDDRM applied the same support and confident whereas WRBDDRM utilized different values for supprot and confident. Obviously, assigning and investigateing supprot and confident need extra operation and exceed the reqiured time.

5.2.2 Experiments 2. Test on TPC-E dataset

Figure 6 shows the result of CID and the base line RF (Ronao and Cho 2015) method in terms of TP, FN, FP, TN on TPC-E dataset. The RF applied 11,000 labeled query which 30% of them are intrusions. In this paper tenfold cross validation is utilized. Similar to TPC-C dataset CID gained 100% in terms of TP and outperforms the baseline. Here CID has 5% improve against the RF. According to FN our method gained 0 zero. Whereas based of FP suggested model gained higher amount in comparison to RF. It shows a growth of 8%. Considering the TN measure our method experienced a decrease of 8% against the RF. Appling various Principal Component Analysis in RF can be the reason of its merit.

Turning to the precision, recall and f-score as Fig. 7 shows CID has competitive results against RF. Clearly CID gained 100% in terms of recall and outperform baseline. Same as TP measure CID has a growth of 5% against its baseline. According to precision evaluation measure, our model reached 82% which is 5% lower than what RF gained. These near values for precision and recall lead to almost equal result based on f-score for both CID and RF.

Generally, CID shows competitive performance with its base line RF. Whilst in terms of TP, FN and recall CID outperforms the RF according to other measure such as FP, TN and precision it received weaker results. In worst state our model just has 8% difference with the baseline. Considering high values of TN and precision, and low amount of FP gained here CID showed acceptable performance. The advantage of CID against other clustering model is its capability in detection of unknown attack or apparently normal intrusion. In addition, RF required constructing trees, clearly RF used 30 trees which is a time consuming process. Order of complexity for constructing each tree in RF is O (v*nlog(n)), which n shows the number of instances and v indicates the number of variables. In CID clustering is fulfilled offline and order of our model in the worst case is O(n^2) in labeling part.

Considering results based on evaluation metrics, CID noticeably performs the better than baselines. In addition, CID is not limited to role based databases whilst all baselines rely on roles in the database. Morevoer in suggested algorithm the conditional expressions in queries are considered whereas baseline merely focused on attributes and tables. Here clustering is applied which inherit sound properties such as fairly low complexity. CID is a density based algorithm which does not require to determine the number of clusters and find non-overlapping clusters led to high accuracy. This kind of clustering is robust against outliers, which the results on two datasets proved it.

6 Conclusions and future works

In this research, a new density-based clustering method called CID is proposed to intrusion detection problem. The results of CID compared to two articles using two datasets show that in comparison with WRBDDRM, FN and FP are improved. In comparison with RF, it has an improvement in FN, but FP higher than FP of RF, showing that it has a lower performance in this part. Generally, the results show that the proposed method, has a valuable improvement in FN. For future work developing a distance metric which based on semantic rather than words and letter is suggested. Improving the clustering performance by tuning eps and minPts parameter can be another future plan. Other open challenge is about supplying standard databases that provided specifically for the intrusion detection issue.

References

Aggarwal CC (2013) An introduction to outlier analysis. In Outlier analysis. Springer, New York, pp 1–40

Barbara D, Goel R, Jajodia S (2003) Mining malicious corruption of data with hidden markov models. In: Gudes E, Shenoi S (eds) Research directions in data and applications security, IFIP, vol 128. Springer, Berlin, pp 175–189

Bland JM, Altman DG (1996) Statistics notes: measurement error. BMJ 312(7047):1654

Bockermann C, Apel M, Meier M (2009) Learning SQL for database intrusion detection using context-sensitive modellingl. In: Flegel U, Bruschi D (eds) DIMVA 2009. LNCS, vol 5587. Springer, Heidelberg, pp 196–205

Bu SJ, Cho SB (2017) A hybrid system of deep learning and learning classifier system for database intrusion detection. In: Martínez de Pisón F, Urraca R, Quintián H, Corchado E (Eds.). Hybrid artificial intelligent systems. HAIS 2017. Lecture notes in computer science, Vol. 10334. Springer, Cham

Choi SG, Cho S-B (2017) Adaptive database intrusion detection using evolutionary reinforcement learning. In: Perez Garcia H, Alfonso-Cendon J, Sanchez Gonzalez L, Corchado E, Quintian H (Eds.). International joint conference SOCO'17- CISIS'17-ICEUTE'17, Proceedings (pp. 547–556). Advances in intelligent systems and computing; Vol. 649. Springer Verlag

Corona I, Giacinto G, Roli F (2013) Adversarial attacks against intrusion detection systems: taxonomy, solutions and open issues. Inf Sci 239:201–225

Darwen H (2009) An introduction to relational database theory, 3rd edn. Bookboon

Doroudian M, Shahriari HR (2014) Database intrusion detection system for detecting malicious behaviors in transaction and inter-transaction levels., 7th international symposium on telecommunications (IST'2014), pp. 809–814

Du H (2010) Data mining techniques and applications: an introduction. Cengage Learning, Boston

Dua S, Du X (2016) Data mining and machine learning in cybersecurity. CRC Press, Boca Raton

Ester M, Peter H, Jörg S et al. (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Simoudis E, Han J, Fayyad UM (Eds.). Proceedings of the second international conference on knowledge discovery and data mining (KDD-96). AAAI Press. pp. 226–231

Felemban M, Javeed Y, Kobes J et al. (2018) Design and evaluation of a data partitioning-based intrusion management architecture for database systems. arXiv:1810.02061

Gogoi P, Borah B, Bhattacharyyac D (2011) Supervised anomaly detection using clustering based normal behaviour modeling. Int J Adv Eng Sci 1(1):12–17

Gomaa WH, Fahmy AA (2013) A survey of text similarity approaches. Int J Comput Appl 68:13

Grossi V, Monreale A, Nanni M et al (2015) Clustering formulation using constraint optimization. In: Bianculli D, Calinescu R, Rumpe B (eds) Software engineering and formal methods. SEFM

2015. Lecture notes in computer science, vol 9509. Springer, Berlin

Hans-Peter K, Peer K, Jörg S et al (2011) Density-based Clustering. WIREs Data Min Knowl Discov 1(3):231–240 **(J.M.P. Martinez)**

Hassanzadeh H, Keyvanpour M (2013) A two-phase hybrid of semi-supervised and active learning approach for sequence labeling. Intell Data Anal 17(2):251–270

Hu Y, Panda B (2004) A data mining approach for database intrusion detection. ACM symposium on applied computing, pp. 711–716

James G (2013) An introduction to statistical learning: with applications in R. Springer, Berlin, p 176

Kamber M, Pei J (2011) Data mining: concepts and techniques. Morgan Kaufmann, Burlington

Kamra A, Terzi E, Bertino E (2008) Detecting anomalous access patterns in relational databases. VLDB J 17(5):1063–1077

Kundu A, Sural S, Majumdar AK (2010) Database intrusion detection using sequence alignment. Int J Inf Secur 9(3):179–191

Miller FP, Vandome AF, Mc Brewster J (2009) Levenshtein distance: information theory, computer science, string (computer science), string metric, Damerau? Levenshtein distance, spell checker, hamming distance. Alpha Press

Moradi M, Keyvanpour M (2015) An analytical review of XML association rules mining. Artif Intell Rev 43(2):277–300

Mordai F (2014) Improving community detection methods for network data analysis. Phd thesis

Pinzón C, Herrero A, De Paz JF et al (2010) CBRid4SQL: a CBR intrusion detector for SQL injection attacks. In: Corchado E, Graña Romay M, Manhaes Savio A (eds) HAIS 2010, Part II. LNCS, vol 6077. Springer, Heidelberg, pp 510–519

Pirrone R, Cannella V, Giordano G et al. (2018) Linear density-based clustering with a discrete density model. arXiv:1807.08158v

Pourkazemi M, Keyvanpour M (2017) Community detection in social network by using a multi-objective evolutionary algorithm. Intell Data Anal 21(2):385409

Ramasubramanian P, Kannan A (2004) Intelligent multi-agent based database hybrid intrusion prevention system. In: Benczúr AA, Demetrovics J, Gottlob G (eds) ADBIS 2004. LNCS, vol 3255. Springer, Heidelberg, pp 393–408

Ramasubramanian P, Kannan A (2006) A genetic algorithm based neural network shortterm forecasting framework for database intrusion prediction system. Soft Comput 10(8):699–714

Rani S, Singh J (2018) Enhancing Levenshtein's edit distance algorithm for evaluating document similarity. In: Sharma R, Mantri A, Dua S (eds) Computing, analytics and networks. ICAN 2017. Communications in computer and information science, vol 805. Springer, Singapore

Rao UP, Singh NK (2017) Weighted role based data dependency approach for intrusion detection in database. Int J Netw Secur 19(3):358–370

Ronao CA, Cho SB (2014) A comparison of data mining techniques for anomaly detection in relational databases. Int Conf on Digital Society (ICDS), pp. 11–16

Ronao CA, Cho SB (2015) Mining SQL queries to detect anomalous database access using random forest and PCA. In International conference on industrial, engineering and other applications of applied intelligent systems, Vol. 9101, pp. 151160. Springer, Cham

Sallam A, Bertino E (2019a) Result-based detection of insider threats to relational databases. Proceedings of the ninth ACM conference on data and application security and privacy, pp. 133–143

Sallam A, Bertino E (2019b) Techniques and systems for anomaly detection in database systems. In: Calo S, Bertino E, Verma D (eds) Policy-based autonomic data governance. Lecture notes in computer science, vol 11550. Springer, Cham

Santos RJ, Bernardino J, Vieira M (2014) Approaches and challenges in database intrusion detection. ACM SIGMOD Rec 43(3):36–47

Sasaki Y (2007) The truth of the F-measure. https://www.toyota-ti.ac.jp/Lab/Denshi/COIN/people/yutaka.sasaki/F-measure-YS-26Oct07.pdf. Accessed 5 June 2019

Shirzad MB, Keyvanpour M (2017) Weighted similarity: a new similarity measure for document ranking features. In: Silhavy R, Senkerik R, Kominkova Oplatkova Z, Prokopova Z, Silhavy P (eds) Artificial intelligence trends in intelligent systems. CSOC 2017. Advances in intelligent systems and computing, vol 573. Springer, Cham, pp 273–280

Srivastava A, Sural S, Majumdar AK (2006) Database intrusion detection using weighted sequence mining. J Comput 1(4):8–17

Subudhi S, Panigrahi S (2019) Application of OPTICS and ensemble learning for database intrusion detection. J King Saud Univ Comput Inf Sci. https://doi.org/10.1016/j.jksuci.2019.05.001

Wee CK, Nayak R (2019) A novel machine learning approach for database exploitation detection and privilege control. J Inf Telecommun 3:308–325

Yu X, Chu Y, Jiang F et al (2018) SVMs classification based two-side cross domain collaborative filtering by inferring intrinsic user and item features. Knowl-Based Syst 141:80–91

Yu X, Jiang F, Du J et al (2019) A cross-domain collaborative filtering algorithm with expanding user and item features via the latent factor space of auxiliary domains. Pattern Recogn 94:96–109

Zandian ZK, Keyvanpour M (2017) Systematic identification and analysis of different fraud detection approaches based on the strategy ahead. KES J 21(2):123–134

Zhang J, Zulkernine M, Haque A (2008) Random-forests-based network intrusion detection systems. Syst Man Cybern 38(5):649–659