Chapter 6
Data Stream Clustering for Real-Time Anomaly Detection: An Application to Insider Threats

Diana Haidar and Mohamed Medhat Gaber

6.1 Introduction

A data stream is a continuous acquisition of data generated from various source(s) in a dynamic environment, typically in a high velocity, leading to accumulation of large volumes of data. This characterisation leads to a typical Big Data computational problem. The dynamic nature of a data stream imposes a change in the data over time. In real-time data streaming, a change refers to an anomalous data that deviates from the normal baseline (e.g. credit card fraud, network intrusion, cancer, etc.). The ultimate aim of such stream mining problems is to detect anomalous data in real-time.

The absence of prior knowledge (no historical database) is often entangled to a real-time stream mining problem. The anomaly detection system is required to employ unsupervised learning to construct an adaptive model that continuously (1) updates with new acquired data, and (2) detects anomalous data in real-time. The system usually acquires data as segments and identifies the *outliers* in the segment as anomalous. An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism [23]. To detect outliers in a stream mining problem, several approaches have been proposed, nevertheless, unsupervised clustering has been successfully applied to identify the patterns in the data and spot outliers [3]. In this work, we select the insider threat problem as a real-world application to detect malicious insider threats (outliers) in real-time. With the absence of labelled data (no previously logged activities executed by users in an organisation), the insider threat problem poses a challenging stream mining problem.

D. Haidar (✉) · M. M. Gaber
Birmingham City University, Birmingham, UK
e-mail: diana.haidar@bcu.ac.uk; mohamed.gaber@bcu.ac.uk

O. Nasraoui, C.-E. Ben N'Cir (eds.), *Clustering Methods for Big Data Analytics*,
Unsupervised and Semi-Supervised Learning,
https://doi.org/10.1007/978-3-319-97864-2_6

Insider threat detection is an emergent concern for academia, industries, and governments due to the growing number of insider incidents in recent years. An insider is a current or former employee, contractor, or business partner of an organisation who has authorised access to the network, system, or data (e.g. trade secrets, organisation plans, and intellectual property) [35]. Malicious insider threats are attributed to insiders who exploit their privileges with the intention to compromise the confidentiality, integrity, or availability of the system or data. According to the 2011 Cybersecurity Watch Survey [10], 21% of attacks are attributed to insiders in 2011, while 58% are attributed to outsiders. However, 33% of the respondents inspect the insider attacks to be more costly, compared to 51% in 2010. For instance, Harold Martin, a former top-security contractor at the National Security Agency (NSA), was recently convicted for stealing around 500 million pages of national defence information over the course of 20 years [38]. Earlier in 2013, Snowden's attack was reported as the biggest intelligence leakage in the USA [42]. Edward Snowden, a former contractor to the NSA, disclosed 1.7 million classified documents to the mass media.

The challenge of the insider threat detection problem lies in the variety of malicious insider threats in the data sets. Each malicious insider threat is devised of a complex pattern of anomalous behaviours carried out by a malicious insider, thus making it difficult to detect **all** anomalous behaviours per threat. Analytically, some anomalous instances (behaviours) which exist in a dense area of normal instances have a high similarity to normal instances. These anomalous instances are difficult to detect and may be missed by the detection system.

Based on the challenge of the problem, we formulate this work with the aim to detect **any-behaviour-all-threat**; it is sufficient to detect **any** anomalous behaviour in **all** malicious insider threats. In other words, we can hunt a malicious insider threat by at least detecting one anomalous behaviour among the anomalous behaviours associated with this threat. We call this approach *threat hunting*. The design of the proposed approach with such a relaxing condition contributes in reducing the frequent false alarms.

Figure 6.1 illustrates a continuous data stream of behaviours (instances) including normal behaviours and anomalous behaviours. Each arrow denotes a behaviour (instance) X^t executed by a user at a specific period of time. Let the blue arrows represent the normal behaviours. Let the green arrows and the red arrows represent

Fig. 6.1 Continuous data stream of behaviours

the anomalous behaviours which belong to the malicious insider threats T_1 and T_2, respectively. To detect a malicious insider threat T_1, it is required to detect **any** green behaviour X^t. Hence, it is essential to detect **any** of the anomalous behaviours per malicious insider threat; the aim to detect any-behaviour-all-threat. Note that the proposed approach may detect **more than one** behaviour which belong to a malicious insider threat, nevertheless, the ultimate aim is to detect one anomalous behaviour per threat.

Based on the above argument, the feature space is defined as a set of features which describe the user's behaviour. Each feature is extracted from the data set logs to represent a user's behaviour related to a particular activity. A feature vector (i.e. instance, behaviour) represents a set of feature values (i.e. actions, commands) evaluated at a period of time. A more detailed description about the feature space is provided in Sect. 6.3.

Several machine learning approaches have been suggested to address the insider threat problem. However, these approaches still suffer from a high number of false alarms. A recent real-time anomaly detection system, named RADISH, based on k nearest neighbours (k-NN) is proposed by Bose et al. [5]. The experimental results showed that 92% of the alarms flagged for malicious behaviour are actually benign [false positives (FPs)].

To address the shortcoming of the high number of false alarms, we propose a streaming anomaly detection approach, namely Ensemble of Random subspace Anomaly detectors In Data Streams (E-RAIDS). We presented a preliminary version of E-RAIDS (coined RandSubOut) in [22]. E-RAIDS is built on top of the established outlier detection techniques [Micro-cluster-based Continuous Outlier Detection (*MCOD*) or Anytime Outlier Detection (*AnyOut*)] which employ clustering over continuous data streams. The merit of E-RAIDS is its capability to detect malicious insider threats in real-time (*based on the definition of real-time in terms of window iterations as discussed in Sect.* 6.4).

E-RAIDS learns an ensemble of p outlier detection techniques (either MCOD or AnyOut), such that each model of the p models learns on a random feature subspace. The acquired data is accumulated in a temporary *buffer* of pre-defined capacity (equals to a fixed window size). So that, at each window iteration, each of the p models in the ensemble is updated with the acquired data, and local outliers are identified in the corresponding feature subspace.

Let $p = n + 1$ denote the number of feature subspaces selected randomly, such that n is set to the number of features (dimension) of the feature space F in the community data set. n represents the number of feature pairs (i.e. two features per subspace), and 1 represents the whole feature space (i.e. all features). In this way, E-RAIDS is capable to detect *local outliers* in the n feature pairs, as well as *global outliers* in the whole feature space. Hence, E-RAIDS employs the idea of random feature subspaces to detect local outlier(s) (anomalous behaviour(s)) which might not be detected over the whole feature space. These anomalous behaviour(s) might refer to malicious insider threat(s).

E-RAIDS introduces two factors: (1) a survival factor vf_s, which confirms whether a **s**ubspace votes for an alarm, if outlier(s) survive for a vf_s number of

window iterations; and (2) a vote factor vf_e, which confirms whether an alarm should be flagged, if a vf_e number of subspaces in the **e**nsemble vote for an alarm. E-RAIDS employs an aggregate component that aggregates the results from the p feature subspaces, in order to decide whether to generate an alarm. The rationale behind this is to reduce the number of false alarms.

The main contributions of this chapter are summarised as follows:

- an ensemble approach on random feature subspaces to detect local outliers (malicious insider threats), which would not be detected over the whole feature space;
- a survival factor that confirms whether outlier(s) on a feature subspace survive for a number of window iterations, to control the vote of a feature subspace;
- an aggregate component with a vote factor to confirm whether to generate an alarm, to address the shortcoming of high number of FPs;
- a thorough performance evaluation of E-RAIDS-MCOD and E-RAIDS-AnyOut, validating the effectiveness of voting feature subspaces, and the capability to detect (more than one)-behaviour-all-threat detection (Hypothesis 2) in real-time (Hypothesis 1).

E-RAIDS extends the preliminary version RandSubOut [22], where it upgrades the selection of the set of outliers (anomalous behaviours) identified at a window iteration over a feature subspace to improve the detection performance of malicious insider threats over the whole ensemble. This is later described in Sect. 6.3.3.1. Unlike RandSubOut, we evaluate E-RAIDS on community data sets such that each community is richer with malicious insider threats which map to a variety of scenarios. Moreover, E-RAIDS is evaluated, not only in terms of the number of detected threats and FP Alarms, however, in terms of (1) F1 measure, (2) voting feature subspaces, (3) real-time anomaly detection, and (4) the detection of (more than One)-behaviour-all-threat.

The rest of this chapter is organised as follows. Section 6.2 reviews the techniques which utilised clustering to detect outliers, and the stream mining approaches proposed for insider threat detection. Section 6.3 presents the proposed streaming anomaly detection approach, namely E-RAIDS, for insider threat detection. Experiments and results are discussed in Sect. 6.4. Finally, Sect. 6.5 summarises the chapter and suggests future work.

6.2 Related Work

The absence of labelled data (i.e. low data maturity) reveals that an organisation has no previously logged activities executed by users (no historical database). We address the absence of prior knowledge using unsupervised streaming anomaly detection built on top of established outlier detection techniques.

Clustering has been successfully applied to identify the patterns of the data for outlier detection [3]. The continuous acquisition of data generated from various

sources defines the streaming environment of the insider threat problem. Several clustering methods have been proposed to handle the streaming environment. The state of the art presents two primitive clustering methods: Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [46], and CluStream [1].

BIRCH is an incremental hierarchical clustering method which was first proposed for very large data sets. BIRCH incrementally and dynamically clusters acquired data instances. It maintains a tree of cluster features (information about clusters) which is updated in an iterative fashion [21]. Thereafter, BIRCH was applied to data stream clustering.

BIRCH was the first to introduce the concepts of micro- and macro-clusters [26]. CluStream is a data stream clustering method that employed those two concepts for the clustering process: online micro-clustering and offline macro-clustering. In the online phase, CluStream scans the acquired data instances and creates micro-clusters in a single pass to handle the big (unbounded) data stream. In the offline phase, CluStream only utilises the micro-clusters and re-clusters into macro-clusters [21].

From BIRCH to CluStream, the concept of data stream clustering is applied, despite the fact that BIRCH includes incremental processing of data instances. *Incremental clustering* processes one data instance at a time and maintains a predefined data structure (i.e. model) that is incrementally updated without the need for model reconstruction [40]. In fact, many incremental methods predate the data stream mining methods. The intrinsic nature of data streams requires methods which implement incremental processing of data instances, in order to handle the resource limitations (i.e. time and memory) [40]. But unlike earlier incremental clustering methods, *data stream clustering* methods require a more efficient time complexity to cope with high data rates [28]. Indeed, the literature stresses the importance of considering the inherent time element in data streams [40]. For instance, a typical data stream clustering method exhibits the temporal aspect of cluster tracking [40]. The dynamic behaviour of cluster over a data stream manifests in the evolving of existing clusters over time; the emergence of new clusters; and the removal of clusters based on a time stamp and/or size. Those cluster updates must be performed on the data structure (i.e. model) very efficiently [28]. Hence, the data stream clustering methods are not only incremental, but are also fast in terms of the inherent temporal aspects.

In this work, data stream clustering is used to underpin the outlier detection techniques for real-time anomaly detection. In the insider threat problem, the temporal aspect is substantial to consider, in order to detect malicious insider threats in real-time (based on the definition of real-time in terms of window iterations as discussed in Sect. 6.4). Hence, data stream clustering methods are more adequate to be utilised in this work than typical incremental clustering methods.

In the following we review the techniques which utilised clustering to detect outliers. Thereafter, we shed light on two outlier detection techniques (MCOD and AnyOut) which employ *data stream clustering*. Those two techniques are later utilised in the proposed framework.

We also review the streaming anomaly detection approaches proposed for insider threat detection.

6.2.1 Clustering for Outlier Detection

It is important to distinguish our objective—to optimise outlier detection—from that of a benchmark of clustering methods developed to optimise clustering (such as DBSCAN [11] and BIRCH [46]). The outliers in these methods [11, 46] are referred to as noise, where they are usually tolerated or ignored. However, in our work, we refer to outliers as anomalous (suspicious) behaviour(s) which may correlate to malicious insider threats. Therefore, clustering is utilised to optimise the detection of outliers.

On the other hand, a benchmark of clustering methods to optimise outlier detection (such as LOF [6] and CBLOF [24]) are developed. Breunig et al. [6] introduced the concept of Local Outlier Factor (LOF) to determine the degree of outlierness of an instance using density-based clustering. The locality of instance (local density) is estimated by the distance to its k nearest neighbours. Thus, the LOF of an instance located in a *dense* cluster is close to 1, while lower LOF is assigned for other instances. He et al. [24] present a measure, called Cluster-based Local Outlier Factor (CBLOF), to identify the physical significance of an outlier using similarity function. The CBLOF of an instance is determined by (1) the distance to its nearest cluster (if it belongs to a small cluster), or (2) the distance to its cluster (if it belongs to a large cluster).

However, the aforementioned methods do not tackle outlier detection in *continuous* data streams. MCOD [28] and AnyOut [3] are two established outlier detection techniques, which employ data stream clustering in *continuous* data streams. MCOD and AnyOut are utilised as building block clustering techniques for the proposed E-RAIDS approach. A detailed description and formalisation for these techniques is found in Sect. 6.3.2.

6.2.2 Streaming Anomaly Detection for Insider Threat Detection

Few approaches have utilised streaming anomaly detection to detect insider threats [36, 37, 41, 45] with no prior knowledge. We give a brief description for these approaches in the following.

Among the emerging interest in deep learning, Tuor et al. [41] present a preliminary effort to utilise deep learning in an online unsupervised approach to detect anomalous network activity in real-time. The authors presented two models: a deep neural network (DNN) model which is trained on each acquired instance only once, and a recurrent neural network (RNN) which learns an RNN per user such that the weights are shared among all users, however, the hidden states are trained per user.

Zargar et al. [45] introduce a Zero-Knowledge Anomaly-Based Behavioural Analysis Method, namely XABA, that learns each user's behaviour from raw logs and network traffic in real-time. XABA is implemented on a big-stream platform without pre-defined or pre-processed activity logs, to handle high rates of network sessions. The authors indicated that XABA reports a low number of FPs, when evaluated on a real traitor scenario.

One of the remarkable approaches is an ensemble of one class SVM (ocSVM), namely Ensemble-based Insider Threat (EIT), proposed by Parveen et al. [36]. The authors proposed an ensemble approach, based on static ocSVM learners, to model a continuous data stream of data chunks (i.e. daily logs). The EIT maintains an ensemble of a k pre-defined number of models M, where the ensemble is continuously updated at each day session upon the learning of a new model M_s. The EIT selects the best $k-1$ models from the k models, having the minimum prediction error over the data chunk C_s, and appends the new model M_s. The results show that ensemble-ocSVM outperforms ocSVM, where it reports a higher accuracy and almost half the number of FP. The authors extended their work in a future paper [37], where the ensemble approach is applied to unsupervised graph-based anomaly detection (GBAD). The results show that ensemble-ocSVM outperforms the ensemble-GBAD in terms of FP.

The above approaches have shown merit in addressing the insider threat detection problem, however, as aforementioned, they do suffer from high false alarms. In this book chapter, we utilise data stream clustering to detect outliers (malicious insider threats), while reducing the number of false alarms.

6.3 Anomaly Detection in Data Streams for Insider Threat Detection

This section identifies the feature space in the insider threat problem and the categories of the feature set extracted. It then describes and formalises established continuous distance-based outlier detection techniques (MCOD and AnyOut). It presents the proposed *E-RAIDS* for insider threat detection with a detailed description of the feature subspace anomaly detection and the aggregate component (ensemble voting).

6.3.1 Insider Threat Feature Space

In this book chapter, we utilise the synthetic data sets, including a variety of malicious insider threat scenarios, generated by Carnegie Mellon University (CMU-CERT) [16]. The CMU-CERT data sets comprise system and network logs for the activities carried out by users in an organisation over 18 months (e.g. logons,

connecting removable devices, copying files, browsing websites, sending emails, etc.). We extract a feature set from these logs in the CMU-CERT data sets according to the literature [5, 30, 31]. This feature set allows us to assess the behaviour of users, and compare it to the previous behaviour of the users or their community of users. We extract each feature considering the evidence it would reveal about an undergoing anomalous behaviour. For example, consider the feature *logon after hours*; this feature if its values is greater than 1, it reveals an evidence of an unusual activity undergoing after the official working hours. Thus, it contributes in the overall decision of the system whether a malicious alarm should be flagged or not.

In the following, we give a more detailed description of the feature set used in this work. We categorise the features into five groups with some examples of each.

- Frequency-based features: assess the frequency of an activity carried out by a community of users over each session slot ($integer$, e.g. *frequency of logon, frequency of connecting devices*);
- Time-based features: assess an activity carried out within the non-working hours period of time ($integer$, e.g. *logon after work hours, device usage after work hours*);
- Boolean features: assess the presence/absence of an activity-related information ($flag = \{0, 1\}$, e.g. *non-empty email-bcc, a non-employee email recipient, sensitive file extension*);
- Attribute-based features: are more specialised features, which assess an activity with respect to a particular attribute (feature) value ($integer$, e.g. *browsing a particular URL (job websites, WikiLeaks)*); and
- Other features: assess the count of other activity-related information. ($integer$, e.g. *number of email recipients, number of attachments to emails*).

This feature set defines the feature space of the insider threat problem, and is used to construct community behaviour profiles for users having the same role (e.g. Salesman, IT admin). A community behaviour profile represents instances (i.e. vectors of feature values) evaluated over session slots, where a session slot represents a period of time from *start time* to *end time*. Each vector of feature values is extracted from the behaviour logs of the community users during a session slot.

In this work, we define the session slot per 4 h to find local anomalous behaviour within a day which would not be detected per day. The rationale behind choosing the session slot *per 4 h* is that this period of time is long enough to extract an instance (i.e. vector of feature values) which provides an adequate evidence of anomalous behaviour. Thus, it supports the system to capture the anomalous behaviours in feature space. If the session slot is chosen per minutes, for example, the extracted instances would lack the adequate evidence of the occurrence of anomalous behaviour. On the other hand, if the session slot is chosen per days/weeks, for example, the period of time will be too long to capture the anomalous behaviour blurred among the normal behaviour in the extracted vector of feature values.

After constructing the community behaviour profiles, we normalise each vector of feature values (over a session slot) to the range [0, 1], and associate it with a class label $\{Normal, Anomalous\}$.

6.3.2 Background on Distance-Based Outlier Detection Techniques

The state of the art presents one of the most widely employed techniques for anomaly detection, which are *distance-based* outlier detection techniques. According to the definition [27], an instance X^t is an outlier, if there exists less than k number of neighbours located at a distance at most r from X^t.

In this work, we are interested in *continuous outlier detection* over a data stream, where recent instances arrive and previous instances expire. The demo paper [15] gives a comparison of four continuous distance-based outlier detection techniques: STream OutlieRMiner (STORM) [2]; Abstract-C [44]; Continuous Outlier Detection (COD) [28]; and Micro-cluster-based Continuous Outlier Detection (MCOD) [28]. COD and MCOD have $O(n)$ space requirements, and they have a faster running time than the exact algorithms of both [2] and [44]. Note that since all algorithms are exact, they output the same outliers [15]. According to [15], COD and MCOD demonstrate a more efficient performance compared to STORM and Abstract-C in terms of lower space and time requirements. Hence, the latter are excluded, but not COD and MCOD. Furthermore, based on the experimental evaluation in [28], MCOD outperforms COD over benchmark tested data sets. Hence, MCOD is selected to be utilised as a base learner in the proposed E-RAIDS approach.

The state of the art presents a further continuous distance-based outlier detection technique, called Anytime Outlier Detection (AnyOut) [3]. However, AnyOut has not been compared to the four techniques in the demo paper. We select MCOD and AnyOut as base learners in E-RAIDS to compare their performance. It is worth to note that the aforementioned distance-based outlier detection techniques are implemented by the authors of [15] in the open-source tool for Massive Online Analysis (MOA) [33].

In the following, a brief description of MCOD and AnyOut techniques is provided.

6.3.2.1 Micro-Cluster-Based Continuous Outlier Detection

Micro-cluster-based Continuous Outlier Detection (MCOD), an extension to Continuous Outlier Detection (COD), stems from the adoption of an event-based technique. The distinctive characteristic of MCOD is that it introduces the concept of evolving micro-clusters to mitigate the need to evaluate the range query for each acquired instance X^t with respect to all the preceding active instances. Instead, it evaluates the range queries with respect to the (fewer) centres of the micro-clusters. The micro-clusters are defined as the regions that contain inliers exclusively (with no overlapping). The micro-clustering is fully performed online [28].

Given that, the centre mc_i of a micro-cluster MC_i may or may not be an actual instance X^t; the radius of MC_i is set to $\frac{r}{2}$, such that r is the distance parameter for outlier detection; and the minimum capacity (size) of MC_i is $k+1$. Below we give a brief formalisation of MCOD.

Let k represent the number of neighbours parameter. Let PD represent the set of instances that doesn't belong to any micro-cluster.

The micro-clusters (i.e. centres of micro-clusters) in MCOD are determined as described in [1]. In the initialisation step, seeds (with a random initial value) are sampled with a probability proportional to the number of instances in a given micro-cluster. The corresponding seed represents the centroid of that micro-cluster. In later iterations, the centre mc_i is the weighted centroid of the micro-cluster MC_i.

- Step 1: For each acquired instance X^t, MCOD finds (1) the nearest micro-cluster MC_i, whose centre mc_i is the nearest to it, and (2) the set of micro-cluster(s) R, whose centres are within a distance $\frac{3\times r}{2}$ from their centres.
- Step 2: If $dist(X^t, mc_i) \leq \frac{r}{2}$; such that mc_i is the centre of the nearest cluster MC_i, X^t is assigned to the corresponding micro-cluster.
- Step 3: Otherwise, a range query q for X^t is evaluated with respect to the instances in (1) the set PD and (2) the micro-clusters of the set R.
- Step 4: Consider n: the number of neighbours $N^{t'} \in P$ to X^t, such that $dist(X^t, N^{t'}) \geq \frac{r}{2}$. If $n > \theta k; \theta \geq 1$, then a new micro-cluster with centre X^t is created and the n neighbours are assigned to this micro-cluster.
- Step 5: A micro-cluster whose size decreases below $k+1$ is deleted and a range query similar to that described for X^t is performed for each of its former instances.
- Step 6: An instance X^t is flagged as an outlier, if there exists less than k instances in either PD or R.

6.3.2.2 Anytime Outlier Detection

Anytime Outlier Detection (AnyOut) is a cluster-based technique that utilises the structure of ClusTree [29], an extension to R-tree [20, 39], to compute an outlier score. The tree structure of AnyOut suggests a hierarchy of clusters, such that the clusters at the upper level subsume the fine grained information at the lower level. ClusTree is traversed in top-down manner to compute the outlier score of an acquired instance X^t until it is interrupted by the subsequent (next) instance X^t. Thus, the descent down the tree improves the certainty of the outlier score, nevertheless, the later the arrival of the subsequent instance the higher the certainty [3].

Given that, a cluster is represented by a Cluster Feature tuple $CF = (n, LS, SS)$, such that n is the number of instances in the cluster, LS and SS are respectively the linear sum and the squared sum of these instances. The compact structure of the tree using CF tuples reduces space requirements. From BIRCH [46] to CluStream [1], cluster features and variations have been successfully used for online summarisation of data, with a possible subsequent offline process for global clustering.

Let e_s represent a cluster node entry in ClusTree. Given defined two scores to compute the degree of outlierness of an instance X^t: (1) a mean outlier score is $s_m(X^t) = dist(X^t, \mu(e_s))$, such that $\mu(e_s)$ is the mean of the cluster node entry $e_s \in$ ClusTree where X^t is interrupted; and (2) a density outlier score is $s_d(X^t) = 1 - g(X^t, e_s)$, such that $g(x_i, e_s)$ is the Gaussian probability density of X^t for $\mu(e_s)$ as defined in [3]. Below we give a brief formalisation of AnyOut.

In the case of a constant data stream:

- Step 1: Initialisation: Each X^t in the data stream is assigned with an actual confidence value $conf(X^t) = e^{s(X^t)}$.
- Step 2: Distribute the computation time for each X^t based on the scattered confidences.
- Step 3: For each acquired instance X^t, AnyOut traverses the tree in a top-down manner until the arrival of the instance X^{t+1} in the data stream.
- Step 4: At the moment of interruption, X^t is inserted to the cluster node entry $e \in$ ClusTree, where X^t arrives (pauses).
- Step 5: The outlier score of X^t, according to the specified parameter $s_m(X^t)$ or $s_d(X^t)$, is computed with respect cluster node entry e_s.
- Step 6: The expected confidence value for the outlier score of X^t is updated based on the computation time. *The confidence value (certainty) increases as the computation time increases*.

6.3.3 E-RAIDS Approach

The established continuous distance-based outlier detection techniques (MCOD and AnyOut), described and formalised in Sect. 6.3.2, are utilised as building block data stream clustering techniques for the proposed E-RAIDS approach.

In this work, we propose a streaming anomaly detection approach, namely Ensemble of Random subspace Anomaly detectors In Data Streams (E-RAIDS), for insider threat detection. In other words, E-RAIDS is an ensemble of an established distance-based outlier detection technique (MCOD or AnyOut), such that each model of the ensemble learns on a random feature subspace. The idea of E-RAIDS is to employ an outlier detection technique on a feature subspace, to detect *local outliers* which might not be detected over the whole feature space. These local outliers may refer to anomalous behaviours (instances) attributed to a malicious insider threat. Hence, the ultimate aim of the E-RAIDS approach is to detect any-behaviour-all-threat (*threat hunting* as defined in Sect. 6.1); a process that leads to a reduction of the number of false alarms.

Figure 6.2 presents the E-RAIDS framework. The set of blue arrows represent a data stream, where each arrow represents an instance (feature vector) X^t acquired at a session slot t. Consider the formalisations below:

- window: a segment of a fixed size w that slides along the instances in a data stream with respect to time (i.e. session slots);

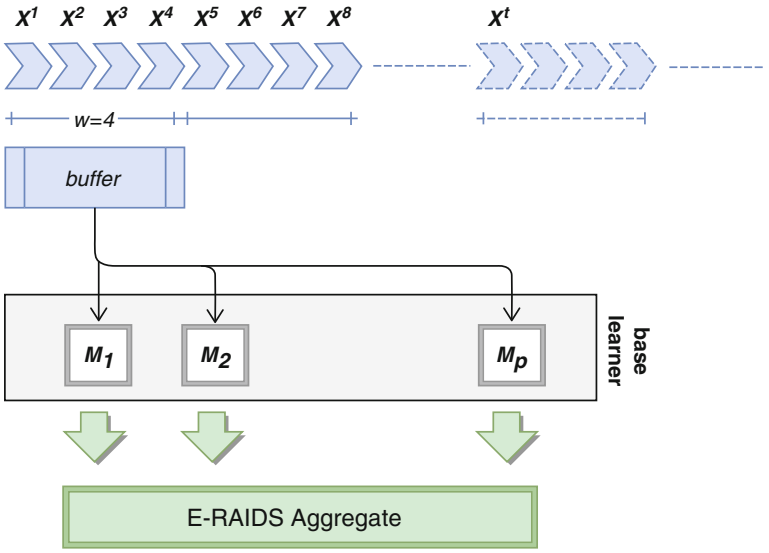

Fig. 6.2 E-RAIDS framework

- *buffer*: a temporary short memory of allocated capacity (equals to w). It temporarily accumulates the instances in a data stream. The *buffer* starts to accumulate the instances in a data stream at the start of the window and stops once the *buffer* is full after w number of instances. The full *buffer* is then input to the base learner component to be processed; and
- window iteration $wIter$: an iteration of the window slide. It starts at the already processed instances (in the previous *buffer*) $procInst$ and ends at $procInst + w$. For instance, $wIter = 0$ starts at $procInst = 0$ and ends at w. $wIter = 1$ starts at $procInst = w$ and ends at $2 \times w$. The window iteration $wIter$ allows the synchronisation between the window slide and the *buffer* accumulation.

Based on the aforementioned formalisations, at each window iteration $wIter$, the *buffer* accumulates w number of instances. Once the *buffer* is full, the instances in the *buffer* are input to the base learner component. A base learner component refers to the distance-based outlier detection technique to be utilised (MCOD or AnyOut). It employs a p number of base models to learn on randomly selected p feature subspaces. A feature subspace $FS_i \subseteq F$ is defined as a subset of features selected from the whole feature space f, where $F = \{f_1, f_2, \ldots, f_n\}$. The rationale behind the idea of random feature subspaces is to detect local outlier(s) (anomalous behaviour(s)) which might not be detected over the whole feature space.

Let $p = n + 1$ represent the number of feature subspaces selected randomly, such that n is set to the number of features (dimension) of the feature space F in the community data set. n represents the number of feature pairs (i.e. 2 features per subspace), and 1 represents the whole feature space (i.e. all features). The p feature

subspaces are utilised to build the ensemble of p models $\{M_1, M_2, \ldots, M_n, M_p\}$, such that $\{M_1, M_2, \ldots, M_n\}$ learn on feature pairs, and M_p learns on the whole feature space. In this way, E-RAIDS is capable to detect *local outliers* in the n feature pairs, as well as *global outliers* in the whole feature space.

6.3.3.1 Feature Subspace Anomaly Detection

For each model M_i on a feature subspace FS_i, we define the following data repositories and a survival factor:

- $outSet$: a temporary set of the outliers detected by a base learner (MCOD or AnyOut) at $wIter$;
- $outTempList$: a list that stores the triples (1) an outlier $out \in outSet$, (2) the $wIter$ where it was first detected, and (3) a temporal count $tempC$ which counts the number of subsequent windows out was detected at. It has the form $\langle out, wIter, tempC \rangle$; and
- $subVoteList$: a list that stores the triples (1) a $wIter$, (2) a $subVote$ parameter set to 1 if the feature FS_i votes for an alarm to be generated at $wIter$ and 0 otherwise, and (3) an outlier set $subOutSet$. It has the form $\langle wIter, subVote, subOutSet \rangle$.
- vf_s: a survival factor that confirms whether a feature subspace FS_i votes for an alarm to be generated at $wIter$. In other words, if an outlier out survived (has been detected) for a vf_s number of subsequent windows, then out is defined as a *persistent* outlier. A persistent outlier boosts the chance of an alarm to be generated at $wIter$.

Over each feature subspace FS_i, the base learner (MCOD or AnyOut) processes the *buffer* at the current $wIter$ to update the model M_i. M_i identifies the outlier set $outSet$, which includes (1) the outliers from the *buffer* at the current $wIter$; and (2) the outliers from the previously learned instances before the model being updated. The type (2) outlier refers to an instance that was identified as an inlier, however, turned into an outlier in the current $wIter$.

For each outlier $out \in outSet$, if out does not exist in $outTempList$, a new triple $\langle out, wIter, 1 \rangle$ is appended to $outTempList$. In this case, $tempC$ is assigned to 1 to declare that it is the first time an outlier out detected. If out exists in the $outTempList$, $tempC$ is incremented by 1 in the triple for out.

For each outlier $out \in outTempList$, E-RAIDS checks (1) if $tempC = vf_s$, then out survived for a vf_s number of subsequent windows. We call it *persistent outlier*. Thus, a persistent outlier confirms that the FS_i votes for an alarm at $wIter$. Thus, $subVote$ is set to 1 in the triple for $wIter$ in $subVoteList$. (2) If $wIter - tempC = vf_s$, then out has turned into an inlier. We call it *expired outlier*. Thus, the triple for the expired outlier out is removed from the $outTempList$. (3) If $wIter - tempC < vf_s$, then out is neither a persistent outlier nor an expired outlier. We call it *potential outlier*.

Algorithm 1 Feature subspace

```
wIter ← 0
foreach X^t ∈ stream do
    accumulate X^t to buffer
    if buffer is full then
        outSet ← get outliers detected by base
        foreach out ∈ outSet do
            if outinoutTempList then
                a new triple ⟨out, wIter, 1⟩ is appended to outTempList
            else
                increment tempC by 1 for out in outTempList
            end
        end
        foreach out ∈ outTempList do
            if tempC = vf_s then
                set subVote to 1 in subVoteList for the current wIter
                remove out from outTempList
                subOutSet ← persistent out∪ potential outliers
                append subOutSet to subVoteList at wIter
            end
            if wIter − tempC = vf_s then
                remove out from outTempList
            end
        end
        increment wIter
    end
    empty buffer
end
return subVoteList
```

Eventually, $outTempList$ consists of persistent outliers and potential outliers (expired outliers have been removed). E-RAIDS appends persistent outliers and potential outliers in $outTempList$ to the $subOutSet$ in the triple for $wIter$ in $subVoteList$.

In the preliminary version RandSubOut [22], only the persistent outliers in $outTempList$ are appended to the $subOutSet$ in the triple for $wIter$ in $subVoteList$. The $subOutSet$ at $wIter$ represents the set of anomalous behaviours detected over a feature subspace FS_i. As later described in Sect. 6.3.3.2, if the ensemble votes to generate an alarm, then the $subOutSet$ for each feature subspace is utilised to evaluate whether all malicious insider threats are detected (i.e. the aim of any-behaviour-all-threat). The experiments carried out on both versions (E-RAIDS and RandSubOut) showed that E-RAIDS outperforms the latter in terms of detecting more malicious insider threats. Analytically, unlike RandSubOut, E-RAIDS considers further the potential outliers in the $subOutSet$ to check for detected malicious insider threats. Thus, the use of potential outliers significantly boosts the detection performance of the proposed approach.

Finally, the *buffer* is emptied to be prepared for the subsequent (next) window of upcoming instances ($wIter + 1$). A step-by-step pseudocode for the E-RAIDS base learner procedure is provided in Algorithm 1.

Algorithm 2 Ensemble of random feature subspaces

foreach $wIter$ **do**
  foreach $FS_i \in FS$ **do**
    $subVote \leftarrow$ get $subVote$ for $wIter$ from $subVoteList$ for FS_i
    $eVote \leftarrow eVote + subVote$ for $wIter$ in $eVoteList$
    $subOutSet \leftarrow$ get $subOutSet$ for $wIter$ from $subVoteList$ for FS_i
    append $subOutSet$ to $eOutSet$ for $wIter$ in $eVoteList$
  end
  if $eVote = vf_e$ **then**
    flag an alarm
  end
end

6.3.3.2 Ensemble of Random Feature Subspaces Voting

For the ensemble of p models $\{M_1, M_2, \ldots, M_n, M_p\}$ on feature subspaces $\{FS_1, FS_2, \ldots, FS_n, FS_p\}$ respectively, we define the following data repository and a vote factor:

- $eVoteList$: a list that stores the triples (1) a $wIter$, (2) an $eVote$ parameter that counts the number of feature subspaces that vote for an alarm, and (3) an outlier set $eOutSet$ that appends the $subOutSet$ from each FS_i votes for an alarm. It has the form $\langle wIter, subVote, subOutSet \rangle$.
- vf_e: a vote factor that confirms whether an alarm to be generated at $wIter$ by the whole **e**nsemble. In other words, if a vf_e number of feature subspaces vote for an alarm, then an alarm is generated at $wIter$.

As illustrated in Fig. 6.2, the E-RAIDS aggregate component aggregates the results from the p feature subspaces, in order to confirm whether to generate an alarm or not at each window iteration $wIter$. For each feature subspace FS_i, if $subVote = 1$ for $wIter$, E-RAIDS adds $subVote$ to $eVote$ for $wIter$ in $eVoteList$. Furthermore, E-RAIDS gets $subOutSet$ for $wIter$ from $subVoteList$, and appends to $eOutSet$ for $wIter$ in $eVoteList$.

After getting the votes from all the feature subspaces in the ensemble, E-RAIDS checks if $eVote = vf_e$. If the condition is satisfied, then an alarm of a malicious insider threat is generated at $wIter$. The voting mechanism is technically controlled by the vote factor vf_e, such that if a vf_e number of feature subspaces vote for anomalous behaviour(s) at a window iteration $wIter$, then an alarm is generated. This accordingly handles the case of a conflict, where $\frac{p}{2}$ (50%) of the feature subspaces in the ensemble vote for anomalous behaviour(s) and the other $\frac{p}{2}$ vote for normal behaviour(s). The ensemble technically checks if $\frac{p}{2} = vf_e$, then an alarm is generated.

A step-by-step pseudocode for the E-RAIDS aggregate procedure is provided in Algorithm 2.

6.4 Experiments

We evaluated the effectiveness of the proposed approach on the CMU-CERT data sets using Windows Server 2016 on Microsoft Azure (RAM $140GB$, OS 64-$bits$, CPU Intel Xeon $E5-2673v3$) for data pre-processing and Microsoft Windows 7 Enterprise (RAM $12GB$, OS 64-$bits$, CPU Intel Core $i5$-$4210U$) for experiments. First, MATLAB $R2016b$ was used to pre-process the data set and generate community behaviour profiles per session slots of 4 h. Second, we implemented E-RAIDS-MCOD and E-RAIDS-AnyOut and carried out the experiments in Java environment (Eclipse Java Mars) using the open-source **MOA** package [33].

6.4.1 Description of the Data set

A significant impediment to researchers who work on the insider threat problem is the lack of real world data. The real data logs the activities executed by the users in an organisation. These data log files contain: private user profile information (e.g. name, email address, mobile number, home address, etc.); intellectual property (e.g. strategic or business plans, engineering or scientific information, source code, etc.); and confidential content (e.g. email content, file content, etc.) [7]. Organisations commonly refuse to give researchers access to real data to protect its users and assets.

In the current decade, there exists a great recent trend towards the utilisation of the CMU-CERT data set(s) for the insider threat detection systems [5, 12, 13, 30, 41, 43]. The CMU-CERT data sets are synthetic insider threat data sets generated by the CERT Division at Carnegie Mellon University [9, 16]. CMU-CERT data repository is the only one available for insider threat scenarios (5 scenarios) and has recently become the evaluation data repository for researchers addressing the insider threat problem [5, 17, 41].

In the preliminary version [22], we used $r5.1$ CMU-CERT data set, where each community consists of one malicious insider threat which map to one scenario.

For this chapter, we used $r5.2$ CMU-CERT data set which logs the behaviour of 2000 employees over 18 months. Unlike the previously released data sets, the communities in the $r5.2$ data set consist of multiple malicious insider threats which map to different scenarios. Among those employees, we extracted the data logs for employees belonging to the following three communities: Production line worker com-P, Salesman com-S, and IT admin com-I. The community com-P consists of 300 employees, 17 malicious insider threats (associated with 366 anomalous behaviours), where each threat maps to one of the scenarios $\{s1, s2, s4\}$. The community com-S consists of 298 employees and 22 malicious insider threats (associated with 515 anomalous behaviours), where each threat maps to one of the scenarios $\{s1, s2, s4\}$. The community com-I consists of 80 employees and 12 malicious insider threats (associated with 132 anomalous behaviours), where each threat maps to one of the scenarios $\{s2, s3\}$.

6.4.2 Experimental Tuning

In this work, we define two experiments based on the proposed E-RAIDS approach. The E-RAIDS-MCOD learns an ensemble of p MCOD base learners, such that each MCOD base learner is trained on a feature subspace of the p feature subspaces. Likewise, the E-RAIDS-AnyOut learns an ensemble of p AnyOut base learners.

The experiments are tuned for different values of parameters. Table 6.1 presents the values for the parameters tuned in each of MCOD and AnyOut, with a short description. Note that an extensive number of experiments were done to select the presented tuning values for the parameters. The values were selected based on E-RAIDS achieving the best performance in terms of the evaluation measures described below.

For MCOD, the parameter k has a default value $k = 50$ in the MOA package. In this chapter, we present $k = \{50, 60, 70\}$ to evaluate the E-RAIDS-MCOD for different number of neighbours. The parameter r, with a default value $r = 0.1$, is presented for a range for $r = \{0.3, 0.4, 0.5, 0.6, 0.7\}$ to check the influence of r.

For AnyOut, the parameter $|D_{train}|$ is presented for 500 instances, knowing that no threats are present at the beginning of the stream in these 500 instances. The parameters *confAgr* and *conf* did not show a significant influence on the utilised data sets, so both are assigned to their default values, in the MOA package, 2 and 4, respectively. τ has a minimum value 0 and a maximum value 1, so it is presented for $\tau = \{0.1, 0.4, 0.7\}$ to evaluate the influence of varying the outlier score threshold on the outliers detected. $oscAgr$ has a minimum value 1 and a maximum value 10, so it is presented for the $oscAgr = \{2, 4, 6, 8\}$.

In general, for E-RAIDS approach with either MCOD or AnyOut base learner, we present the vouch factor $vf_s = 2$, so it confirms an outlier as positive (anomalous) after it survives for 2 subsequent windows. We present the vote factor $vf_e = 1$, so if at least 1 feature subspace in the ensemble flags an alert, an alarm of a malicious insider threat is confirmed to be generated.

Likewise, for both E-RAIDS-MCOD and E-RAIDS-AnyOut, the window size is presented for $w = \{50, 100, 150, 200\}$. As previously described, an instance

Table 6.1 Tuned parameters

	Description
MCOD parameter	
$k = \{50, 60, 70\}$	Number of neighbours parameter
$r = \{0.3, 0.4, 0.5, 0.6, 0.7\}$	Distance parameter for outlier detection
AnyOut parameter	
$\|D_{train}\| = 500$	Training set size
$confAgr = 2$	Size of confidence aggregate
$conf = 4$	Initial confidence value
$\tau = \{0.1, 0.4, 0.7\}$	Outlier score threshold
$oscAgr = \{2, 4, 6, 8\}$	Size of outlier score aggregate

(feature vector) is a set of events executed during a pre-defined session slot. In this work, we define a session slot per 4 h. Let w represent the window size, which accumulates the acquired instances in a data stream. If $w = 50$ and $vf_s = 2$, then the instances in each window are processed after $4 \times 50 = 200$ h $\simeq 8$ days. Based on the description of E-RAIDS, a threat (outlier) may be confirmed as an outlier at least over the window it belongs to (after 8 days) or over the next window (after $\simeq 8 \times 2 = 16$ days). If $w = 200$ and $vf_s = 2$, then the instances in each window are processed after $4 \times 200 = 800$ h $\simeq 33$ days. A threat (outlier) may be confirmed as an outlier at least (after 33 days) or over the next window (after $\simeq 33 \times 2 = 66$ days). Note that the malicious insider threats simulated in the CMU-CERT data sets span over at least one month and up to 4 months. Hence, we hypothesise the following in Hypothesis 1.

Hypothesis 1 *The E-RAIDS approach is capable to detect the malicious insider threats in real-time (during the time span of the undergoing threat).*

6.4.3 Evaluation Measures

Many research work has been done to detect or mitigate malicious insider threats, but nevertheless has established standard measures to evaluate the proposed models [18]. The research practices show that the insider threat problem demands to measure the effectiveness of the models before being deployed, preferably in terms of true positives (TP) and false positives (FP) [19].

In the state of the art, a remarkable number of approaches were validated in terms of FP measure [4, 5, 14, 32, 37]. This sheds light on the importance of the FP measure to address the shortcoming of the high number of false alarms (FPs). Furthermore, some approaches were validated in terms of: TP measure [32]; F1 measure [4, 34]; AUC measure [8, 12, 17]; precision and recall [25, 30]; accuracy [25, 37]; and others.

The variety of the utilised evaluation measures in the state of the art reveals the critical need to formulate the insider threat problem and to define the measures that would best validate the effectiveness of the propose E-RAIDS approach. In the following, we give a formulation for the E-RAIDS approach and we define the evaluation measures utilised in this work.

As aforementioned, the ultimate aim of the E-RAIDS approach is to detect all the malicious insider threats over a data stream in real-time, while minimising the number of false alarms.

The challenge of the insider threat problem lies in the variety and complexity of the malicious insider threats in the data sets. Each malicious insider threat is devised of a set of anomalous behaviours. An anomalous behaviour is represented

by an instance (feature vector) which describes a set of events (features) carried out by a malicious insider. Based on the challenge of the problem, we formulate the E-RAIDS approach with the aim to detect *any-behaviour-all-threat* (as defined in Sect. 6.1). This means that it is sufficient to detect any anomalous behaviour (instance) in all malicious insider threats. Hence, E-RAIDS approach is formulated as a *threat hunting* approach, where a threat is detected if (1) a vf_e number of feature subspaces confirm an undergoing threat (flag alarm) over a window and (2) the outliers, associated with the alarm flagged over the window, include at least a true positive (anomalous behaviour detected as outlier). Note that although the detection of one behaviour confirms the detection of the threat, we hypothesise the following in Hypothesis 2.

Hypothesis 2 *The E-RAIDS approach is capable of detecting more than one behaviour from the set of behaviours which belong to a malicious insider threat. We refer to as (more than one)-behaviour-all-threat detection in E-RAIDS.*

Furthermore, the property of the E-RAIDS approach of using windows over data streams introduces a refined version of the evaluation of false positives (FP). In this work, we use the FPAlarm to evaluate the false positives. So that if all the outliers associated with the alarm generated over a window are actually normal, then the alarm is considered a false alarm (i.e. FPAlarm is incremented 1). A formal definition for FPAlarm is given in the following.

We define the measures used to evaluate the performance of E-RAIDS, which include a refined version of the default (known) evaluation measures, taking our ultimate aim into account.

- $\mathbf{P}_T$: *Threats* number of malicious insider threats associated with anomalous instances. In other words, P_T is the number of malicious insiders attributed to the anomalous behaviours;
- $\mathbf{TP}_T$: *True Positives* a refined version of default TP to evaluate the number of threats detected by the framework among all the P_T malicious insider threats. TP_T is incremented if at least one anomalous instance (behaviour attributed to the threat) is associated as an outlier to a flagged alarm;
- **FPAlarm**: *False Positive Alarm* a refined version of default FP to evaluate the number of false alarms generated. An alarm is declared false if all the outliers associated with the alarm are actually normal instances. This means that none of the instances contributed to generating the alarm, therefore, false alarm;
- $\mathbf{FN}_T$: *False Negatives* a refined version of default FN to evaluate the number of insider threats not detected; and
- **F1** measure: defined based on the values of the above defined measures.

The evaluation for E-RAIDS does not employ only the defined evaluation measures, however, it is required to prove the previously stated hypothesises to be true.

6.4.4 Results and Discussion

In the preliminary version [22], RandSubOut was evaluated in terms of TP_T detected threats and FPAlarm.

In this work, the results are presented and discussed for E-RAIDS-MCOD and E-RAIDS-AnyOut as follows: in terms of (1) the pre-defined evaluation measures; (2) voting feature subspaces; (3) real-time anomaly detection; and (4) the detection of (more than One)-behaviour-all-threat.

6.4.4.1 MCOD vs AnyOut Base Learner for E-RAIDS in Terms of Evaluation Measures

In the following, we analyse the performance of E-RAIDS with MCOD base learner vs AnyOut base learner in terms of the pre-defined evaluation measures: TP_T out of P_T; FPAlarm; and F1 measure.

Tables 6.2 and 6.3 present the maximum TP_T and the minimum FPAlarm attained E-RAIDS-MCOD and E-RAIDS-AnyOut over the communities. The results are reported in terms of the parameter values in the given sequence k, r, w for E-RAIDS-MCOD and $\tau, oscAgr, w$ for E-RAIDS-AnyOut, respectively.

E-RAIDS-MCOD

Figure 6.3 presents the variation of $F1$ measure as a function of window size w for E-RAIDS with MCOD base learner over the communities. The results are reported with respect to k and r parameter values.

A preliminary analysis of the F1 measure shows no evident pattern in terms of any of the parameters k, r, or w. Over the community com-P, E-RAIDS-MCOD

Table 6.2 Maximum TP_T of detected insider threats over communities

Community	E-RAIDS-MCOD	Parameters k, r, w
com-P	**16**	50,0.3,100 60,0.4,50
		60,0.6,100 60,0.7,150
		70,0.4,50
com-S	**21**	70,0.4,150
com-I	10	50,0.4,50 70,0.3,100

	E-RAIDS-AnyOut	Parameters $\tau, oscAgr, w$
com-P	**16**	0.1,2,100–200 {0.3, 0.7},2,50–200
com-S	20	0.1,2,50–100 0.3,2,50–100
		0.7,2,150
com-I	**12**	0.1,2,50–200 0.3,2,50–100
		0.7,2,{50, 200}

The bold values represent the maximum TPT achieved by either E-RAIDS-MCOD or E-RAIDS-AnyOut over each community

Table 6.3 Minimum FPAlarm over communities

Community	E-RAIDS-MCOD	Parameters k, r, w
com-P	**0**	50,0.4,200 60,{0.3, 0.5, 0.6},200
com-S	**0**	50,0.4,200 50,0.7,100
		60,0.3,200 60,0.5–0.7,100
		70,0.6–0.7,150
com-I	**0**	50,0.3,200 60,0.5,200
		70,0.5–0.7,200

	E-RAIDS-AnyOut	Parameters τ, $oscAgr$, w
com-P	2	$\forall\tau$,$\forall oscAgr$,200
com-S	2	$\forall\tau$,$\forall oscAgr$,150–200
com-I	1	0.1,{2, 4},200 0.3,2–6,200
		0.7,2–4,200

The bold values represent the minimum FPAlarm by either E-RAIDS-MCOD or E-RAIDS-AnyOut over each community

achieves the maximum F1 = 0.9411; 60, 0.7, 150. It detects the maximum $TP_T = 16$ out of $P_T = 17$, thus missing one malicious insider threat. However, it flags only a false positive alarm (FPAlarm = 1). Furthermore, Table 6.3 shows that E-RAIDS-MCOD reports the minimum FPAlarm = 0 at $w = 200$ for different values of k and r.

Over the community com-S, E-RAIDS-MCOD achieves the maximum F1= 0.9523; 70, {0.6, 0.7}, 150. It detects a $TP_T = 20$ out of $P_T = 22$, while flagging no false positive alarms (FPAlarm= 0). The maximum $TP_T = 21$ is attained at 70, 0.4, 150, however, flagging FPAlarm= 2.

Over the community com-I, E-RAIDS-MCOD achieves the maximum F1= 0.6451; 70, 0.3, 100. It detects a $TP_T = 10$ out of $P_T = 12$, thus missing two malicious insider threats, while flagging FPAlarm= 9. Nevertheless, Table 6.3 shows that E-RAIDS-MCOD reports the minimum FPAlarm= 0 at $w = 200$ for different values of k and r.

We can deduce that the window size $w = 150, 200$ gives the best performance for E-RAIDS-MCOD in terms of the evaluation measures.

E-RAIDS-AnyOut

Figure 6.4 presents the variation of F1 measure as a function of window size w for E-RAIDS with AnyOut base learner over the communities. The results are reported with respect to τ and $oscAgr$ parameter values.

It is evident that there exists a positive correlation between F1 measure and the parameter $oscAgr$ for E-RAIDS-AnyOut. The variation of F1 measure at $oscAgr = 2$ is the highest with respect to all the window sizes $w = 50$–200 over both communities. Moreover, Fig. 6.5a reveals a positive correlation between F1 measure and the parameter w. The value of F1 measure increases as the window size w increases.

Over the community com-P, E-RAIDS-AnyOut achieves the maximum F1= 0.9142; $\forall\tau$, 2, 200. It detects the maximum $TP_T = 16$ out of $P_T = 17$, while

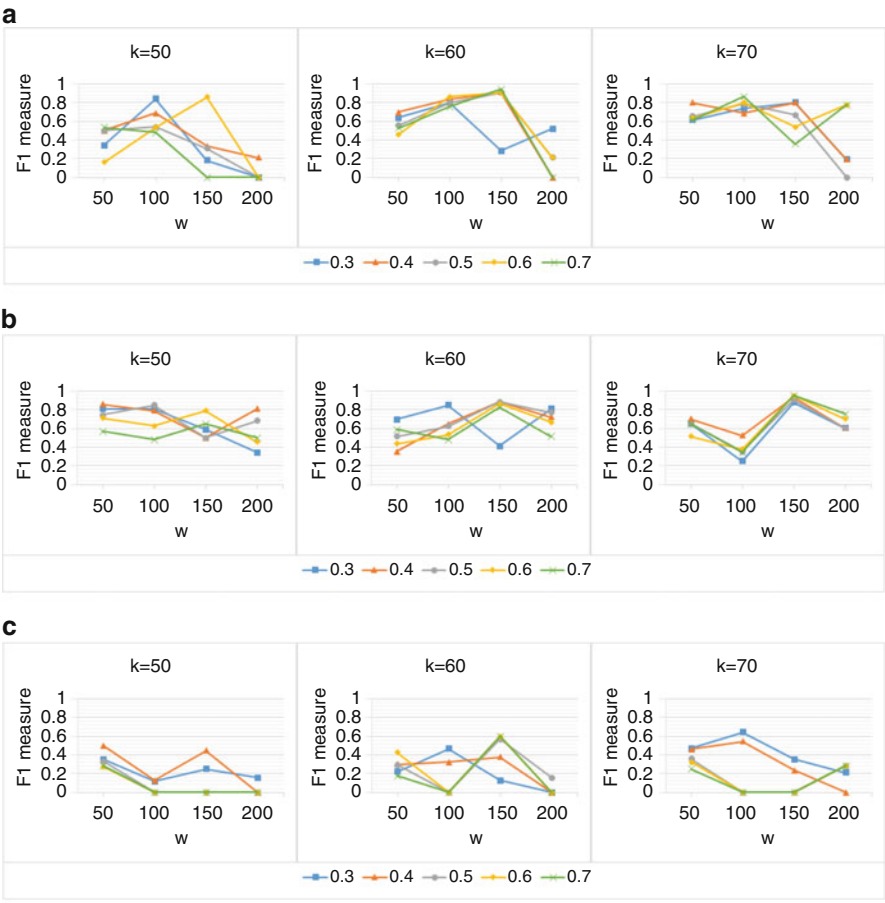

Fig. 6.3 The variation of F1 measure as a function of window size w for E-RAIDS with MCOD base learner over the communities. The legend represents the r parameter values. (**a**) com-P. (**b**) com-S. (**c**) com-I

reducing the false positive alarms to FPAlarm= 2. Table 6.3 shows that E-RAIDS-AnyOut reports the minimum FPAlarm= 2 $\forall \tau, \forall oscAgr$ at $w = 200$. Thus, the higher the window size, the lower the FPAlarm. Table 6.2 shows that E-RAIDS-AnyOut reports the maximum $\text{TP}_T = 16$ at $oscAgr = 2$ in general terms $\forall \tau, \forall w$. Thus, the lower the $oscAgr$, the higher the TP_T detected.

Over the community com-S, E-RAIDS-AnyOut achieves the maximum F1= 0.9090; $0.7, 2, 150$. It detects a $\text{TP}_T = 20$ out of $\text{P}_T = 22$, while flagging two false positive alarms (FPAlarm= 2).

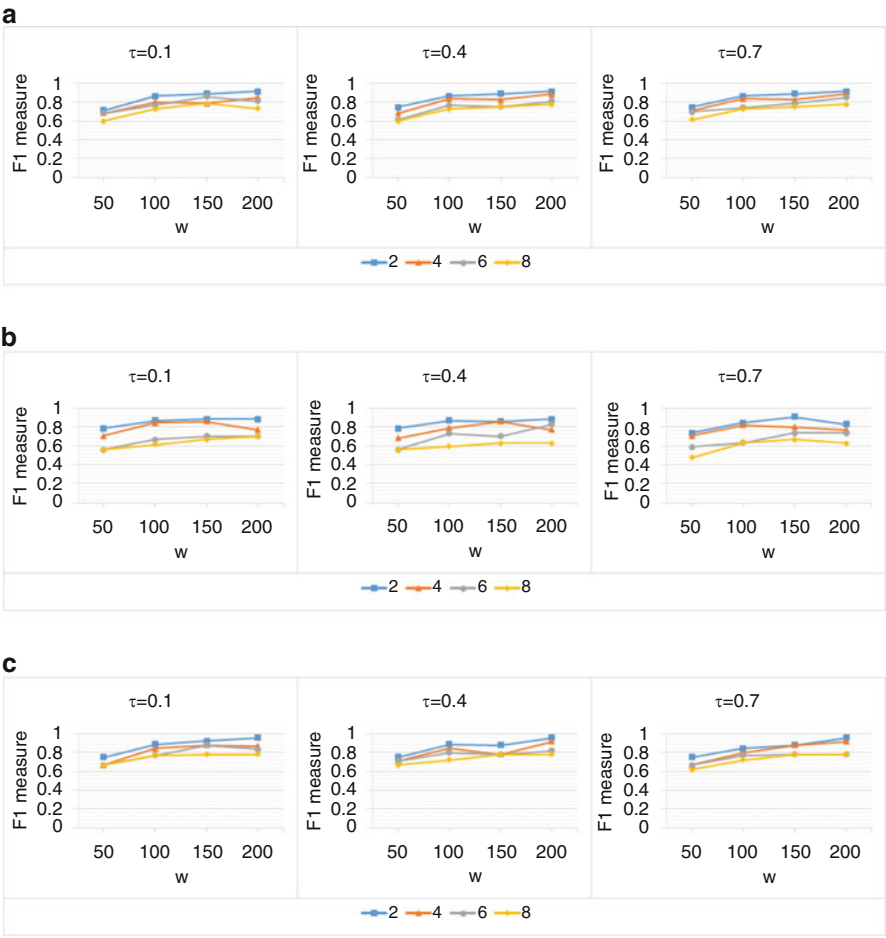

Fig. 6.4 The variation of F1 measure as a function of window size w for E-RAIDS with AnyOut base learner over the communities. The legend represents the $oscAgr$ parameter values. (**a**) com-P. (**b**) com-S. (**c**) com-I

Over the community com-I, E-RAIDS-AnyOut achieves the maximum F1= 0.9600; $\forall\tau, 2, 200$. It detects a $\text{TP}_T = 12$ out of $\text{P}_T = 12$, while flagging one false positive alarm (FPAlarm= 1).

In terms of the evaluation measures, E-RAIDS-MCOD outperforms E-RAIDS-AnyOut over the communities, where E-RAIDS-MCOD achieves a higher F1 measure over com-P and com-S, a higher TP_T over com-S, and a lower FPAlarm= 0 over all communities.

6.4.4.2 MCOD vs AnyOut for E-RAIDS in Terms of Voting Feature Subspaces

In the following, we address the merit of using feature subspaces in the E-RAIDS approach. We compare the number of feature subspaces in the ensemble that voted for a malicious insider threat in each of E-RAIDS-MCOD and E-RAIDS-AnyOut. The rationale behind using a random feature subspace to train each model of the p models in the ensemble is to train the base learner (MCOD or AnyOut) on a subset of the features and to detect local outliers which might not be detected over the whole feature space.

Figure 6.5 illustrates the number of votes which contributed in flagging an alarm of a malicious threat with respect to the number of instances processed (given the window size w) over the communities. The number of votes actually corresponds to the number of feature subspaces in the ensemble which generated an alert. We selected E-RAIDS-MCOD and E-RAIDS-AnyOut with their parameters which showed the best performance in terms of the evaluation measures. For move

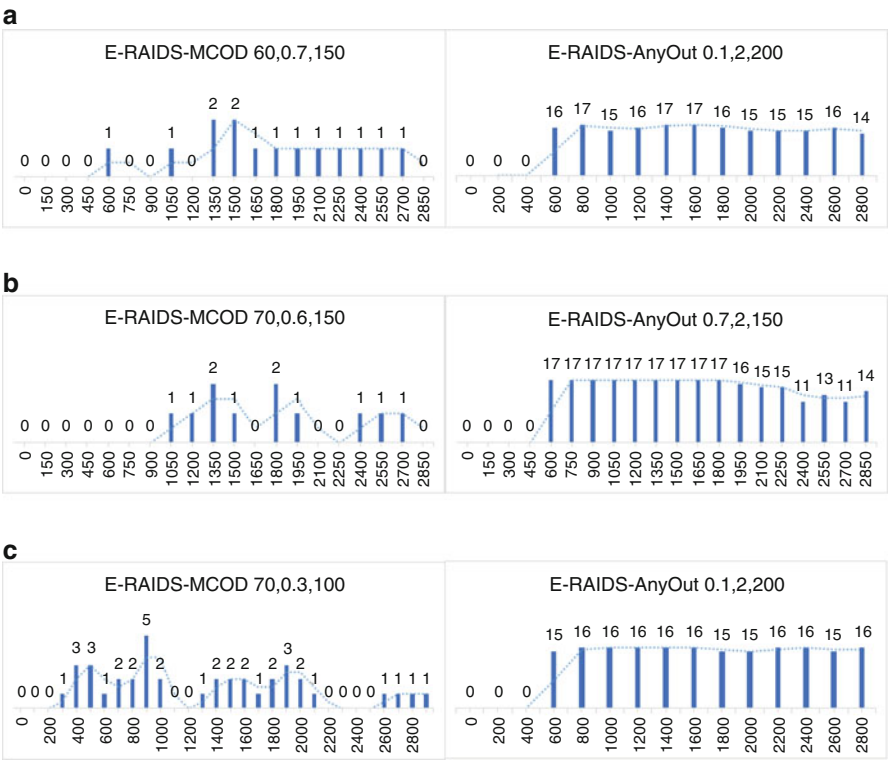

Fig. 6.5 The number of votes which contributed in flagging an alarm of a malicious threat with respect to the number of instances processed over the communities. (**a**) com-P. (**b**) com-S. (**c**) com-I

com-P down to be clear Fig. 6.5a shows the E-RAIDS-MCOD at 60, 0.7, 150 and E-RAIDS-AnyOut at 0.1, 2, 200. For com-S, Fig. 6.5b shows the E-RAIDS-MCOD at 70, 0.6, 150 and E-RAIDS-AnyOut at 0.7, 2, 150. For com-I, Fig. 6.5c shows the E-RAIDS-MCOD at 70, 0.3, 100 and E-RAIDS-AnyOut at 0.1, 2, 200. Given that the $|D_{train}| = 500$, this justifies why the votes for E-RAIDS-AnyOut start after 500 train instances has been processed over both communities.

Given the window size w, a window iteration $wIter$ starts at the number of already processed instances $procInst$ and ends at $procInst+w$. For example, given $w = 100$, the first window iteration $wIter = 0$ starts at $procInst = 0$ and ends at $procInst + w = 100$; $wIter = 1$ starts at 100 and ends at 200; etc.

A preliminary analysis of the results shows that E-RAIDS-AnyOut flags alarms continuously $\forall wIter$ after $procInst = 500$. However, E-RAIDS-MCOD shows distinct alarms flagged, with no alarms at certain windows iterations. We recall that E-RAIDS-MCOD outperforms E-RAIDS-AnyOut in terms of FPAlarm measure. The continuous alarms flagged $\forall wIter$ in E-RAIDS-AnyOut explains the higher FPAlarm, as well as it reveals the uncertainty of E-RAIDS-AnyOut compared to E-RAIDS-MCOD.

Knowing that the number of feature subspaces utilised in the ensemble is $p = 17$, the number of votes $\forall wIter$ in E-RAIDS-AnyOut has a minimum $votes = 11$ and a maximum $votes = 17$, which reveals a level of uncertainty. The case where 17 feature subspaces vote for an alarm indicates that all (17) models of the ensemble detect *at least one* outlier (positive) associated with a malicious insider threat. On the other hand, the number of votes in E-RAIDS-MCOD has a maximum $votes = 2$. This means that only 1 or 2 feature subspaces vote for an alarm. We recall the complexity of the malicious insider threat scenarios in the CMU-CERT data sets. The anomalous instances usually exist in (sparse or dense) regions of normal instances, and rarely as *global outliers* with respect to the whole feature space. To address this, the E-RAIDS approach aims to detect *local outliers* which may be found over ANY (not ALL) of feature subspaces. Having all the feature subspaces in E-RAIDS-AnyOut voting for a threat, compared to a couple (1 or 2) of feature subspaces in E-RAIDS-MCOD, reinforces the uncertainty of E-RAIDS-AnyOut. The reason behind the uncertain performance of AnyOut in the E-RAIDS approach may be due to that the outlier score of an instance X^t is computed upon the arrival of a new instance X^{t+1}. Thus, the processing of the instance X^t is interrupted at a certain level of the ClusTree, and the outlier score is computed with a lower level of confidence (i.e. uncertain).

6.4.4.3 Real-Time Anomaly Detection in E-RAIDS

To prove the aforementioned Hypothesis 1 to be true, it is required to check if the E-RAIDS detects the malicious insider threats in real-time (where real-time means that the alarm is flagged during the time span of the undergoing threat). Based on the previous conclusion regarding the uncertainty of E-RAIDS-AnyOut, and the

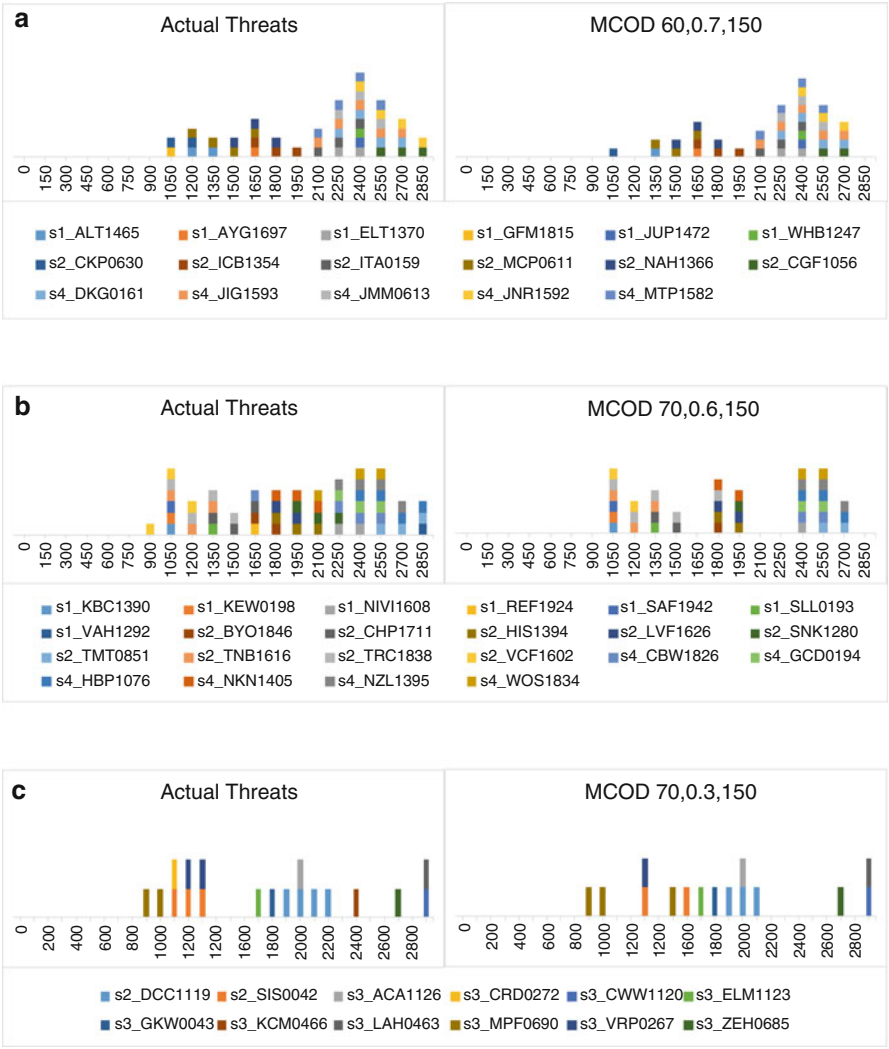

Fig. 6.6 The actual malicious insider threats vs the threats detected in E-RAIDS-MCOD with respect to the number of instances processed over the communities. (**a**) com-P. (**b**) com-S. (**c**) com-I

superiority of E-RAIDS-MCOD in terms of (1) the evaluation measures and (2) the voting feature subspaces, we select E-RAIDS-MCOD to verify Hypothesis 1.

Figure 6.6 illustrates the actual malicious insider threats vs the threats detected in E-RAIDS-MCOD with respect to the number of instances processed over the communities. The malicious insider threats are displayed in the legend over each community using the following label scenRef_insiderID (e.g. s1_ALT1465) such that scenRef (e.g. s1, s2, or s4) represents the reference number for the scenario

followed in the malicious insider threat; and insiderID (e.g. ALT1465, AYG1697) represents the user ID of the insider attributed to the threat. Hence, each colour in the legend refers to a malicious insider threat.

We observe in Fig. 6.6 that a malicious insider threat is detected either (obs1) at the current window iteration $wIter$ where it is actually simulated, or (obs2) at the subsequent (next) $wIter$. Hence, Hypothesis 1 is verified.

Based on the description of the E-RAIDS approach, a feature subspace votes for a threat at $wIter$ if at least an outlier survived for a vf_s number of subsequent windows, and consequently a specific threat is detected at $wIter$ if (cond1) a vf_e number of subspaces vote (an alarm is flagged), and (cond2) at least outlier (positive) associated with the alarm belongs to the threat. However, the outliers associated with the alarm (as mentioned in (cond2)) consist of *persistent* outliers (which survived from $wIter-1$) and *potential* outliers (at the current $wIter$). A potential outlier, if satisfies (cond2), allows real-time detection at the current $wIter$ (obs1). A persistent outlier, if satisfies condition (cond2), allows real-time detection as observed at the subsequent $wIter$ (obs2).

6.4.4.4 (More Than One)-Behaviour-All-Threat Detection in E-RAIDS

The final analysis addresses Hypothesis 2. As defined in Sect. 6.1, the idea of *threat hunting* aims to detect any-behaviour-all-threat, however, Fig. 6.6 shows the capability of E-RAIDS-MCOD to detect more than one behaviour (not only one) from the set of behaviours which belong to a malicious insider threat. It manifests as multiple alarms (colour spikes) generated for a specific threat over a number of windows. Hence, Hypothesis 2 is verified.

Analytically, this underlies in having multiple outliers, associated with the alarm(s) flagged over window(s), which are actually true positives belonging to a specific malicious insider threat.

6.5 Conclusion and Future Work

This chapter addresses the shortcoming of high number of false alarms in the existing insider threat detection mechanisms. The continuous flagging of false alarms deceives the administrator(s) about suspicious behaviour of many users. This consumes a valuable time from their schedule, while investigating the suspected users.

We present a streaming anomaly detection approach, namely Ensemble of Random subspace Anomaly detectors In Data Streams (E-RAIDS), for insider threat detection. The ultimate aim of E-RAIDS is to detect any-behaviour-all-threat (threat hunting as defined in Sect. 6.1), while reducing the number of false alarms.

E-RAIDS is built on top of established continuous outlier detection techniques (MCOD or AnyOut). These techniques use *data stream clustering* to optimise the

detection of outliers (which may refer to malicious insider threats). E-RAIDS is an ensemble of p outlier detectors (p MCOD base learners or p AnyOut base learners), where each model of the p models learns on a random feature subspace. The merit of using feature subspaces is to detect local outliers which might not be detected over the whole feature space. These outliers may refer to anomalous behaviour(s) which belong to a malicious insider threat. E-RAIDS presents also an aggregate component to combine the votes from the feature subspaces, and take a decision whether to flag an alarm or not.

We define two experiments: E-RAIDS-MCOD with MCOD base learner, and E-RAIDS-AnyOut with AnyOut base learner. The experiments are carried out on CMU-CERT data sets which include simulated malicious insider threat scenarios. We compare the performance of E-RAIDS using each of MCOD and AnyOut in terms of: (1) the evaluation measures: F1 measure, TP_t of threats detected, and FPAlarm flagged; (2) the effectiveness of the concept of voting feature subspaces; (3) the capability of E-RAIDS to detect insider threats in real-time (Hypothesis 1); and (4) the capability of E-RAIDS to detect more than one behaviour belonging to an insider threat (Hypothesis 2) despite our formulation to the insider threat approach (threat hunting).

The results show that E-RAIDS-MCOD outperforms E-RAIDS-AnyOut, where the latter shows a low level of certainty in the detection of outliers. E-RAIDS-MCOD reports a higher F1 measure $=$ 0.9411 and 0.9523 over com-P and com-S, a lower FPAlarm $=$ 0 over all communities, and misses only one threat $\text{TP}_T = 16$ and 21 over com-P and com-S. It is worth to also mention that the window size $w = 150, 200$ gives the best performance for E-RAIDS-MCOD compared to the tuned values. E-RAIDS verifies the hypothesised capabilities in terms of the detection of more than one behaviour per threat in real-time.

References

1. C.C. Aggarwal, S.Y. Philip, J. Han, J. Wang, A framework for clustering evolving data streams, in *Proceedings 2003 VLDB Conference* (Elsevier, Burlington, 2003), pp. 81–92
2. F. Angiulli, F. Fassetti, Distance-based outlier queries in data streams: the novel task and algorithms. Data Min. Knowl. Disc. **20**(2), 290–324 (2010)
3. I. Assent, P. Kranen, C. Baldauf, T. Seidl, Anyout: anytime outlier detection on streaming data, in *International Conference on Database Systems for Advanced Applications* (Springer, Berlin, 2012), pp. 228–242
4. A. Azaria, A. Richardson, S. Kraus, V. Subrahmanian, Behavioral analysis of insider threat: a survey and bootstrapped prediction in imbalanced data. IEEE Trans. Comput. Soc. Syst. **1**(2), 135–155 (2014)
5. B. Böse, B. Avasarala, S. Tirthapura, Y.Y. Chung, D. Steiner, Detecting insider threats using radish: a system for real-time anomaly detection in heterogeneous data streams. IEEE Syst. J. **11**(2), 471–482 (2017)
6. M.M. Breunig, H.P. Kriegel, R.T. Ng, J. Sander, LOF: identifying density-based local outliers, in *ACM Sigmod Record*, vol. 29 (ACM, New York, 2000), pp. 93–104

7. D.M. Cappelli, A.P. Moore, R.F. Trzeciak, *The CERT Guide to Insider Threats: How to Prevent, Detect, and Respond to Information Technology Crimes (Theft, Sabotage, Fraud)* (Addison-Wesley, Upper Saddle River, 2012)
8. Y. Chen, S. Nyemba, B. Malin, Detecting anomalous insiders in collaborative information systems. IEEE Trans. Dependable Secure Comput. **9**(3), 332–344 (2012)
9. CMU CERT Team, CMU cert synthetic insider threat data set. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099. Accessed 12 Apr 2018
10. CMU Software Engineering Institute, 2011 cybersecurity watch survey. https://www.sei.cmu.edu/news/article.cfm?assetid=52441. Accessed 14 Feb 2018
11. M. Ester, H.P. Kriegel, J. Sander, X. Xu et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in *Kdd*, vol. 96 (1996), pp. 226–231
12. A. Gamachchi, S. Boztas, Insider threat detection through attributed graph clustering, in *2017 IEEE Trustcom/BigDataSE/ICESS* (IEEE, Piscataway, 2017), pp. 112–119
13. A. Gamachchi, L. Sun, S. Boztas, Graph based framework for malicious insider threat detection, in *Proceedings of the 50th Hawaii International Conference on System Sciences*, Hawaii, 4–7 January 2017, pp. 2638–2647
14. C. Gates, N. Li, Z. Xu, S.N. Chari, I. Molloy, Y. Park, Detecting insider information theft using features from file access logs, in *European Symposium on Research in Computer Security* (Springer, Switzerland, 2014), pp. 383–400
15. D. Georgiadis, M. Kontaki, A. Gounaris, A.N. Papadopoulos, K. Tsichlas, Y. Manolopoulos, Continuous outlier detection in data streams: an extensible framework and state-of-the-art algorithms, in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data* (ACM, New York, 2013), pp. 1061–1064
16. J. Glasser, B. Lindauer, Bridging the gap: a pragmatic approach to generating insider threat data, in *2013 IEEE Security and Privacy Workshops (SPW)* (IEEE, Piscataway, 2013), pp. 98–104
17. H. Goldberg, W. Young, M. Reardon, B. Phillips et al., Insider threat detection in prodigal, in *Proceedings of the 50th Hawaii International Conference on System Sciences* (2017)
18. F.L. Greitzer, T.A. Ferryman, Methods and metrics for evaluating analytic insider threat tools, in *2013 IEEE Security and Privacy Workshops (SPW)* (IEEE, Piscataway, 2013), pp. 90–97
19. M.D. Guido, M.W. Brooks, Insider threat program best practices, in *2013 46th Hawaii International Conference on System Sciences (HICSS)* (IEEE, Piscataway, 2013), pp. 1831–1839
20. A. Guttman, *R-Trees: A Dynamic Index Structure for Spatial Searching*, vol. 14 (ACM, New York, 1984)
21. M. Hahsler, M. Bolanos, J. Forrest et al., Introduction to stream: an extensible framework for data stream clustering research with R. J. Stat. Softw. **76**(14), 1–50 (2017)
22. D. Haidar, M.M. Gaber, Outlier detection in random subspaces over data streams: an approach for insider threat detection. Expert Update **17**(1), 1–16 (2017)
23. D.M. Hawkins, *Identification of Outliers*, vol. 11 (Springer, Dordrecht, 1980)
24. Z. He, X. Xu, S. Deng, Discovering cluster-based local outliers. Pattern Recogn. Lett. **24**(9–10), 1641–1650 (2003)
25. M. Kandias, V. Stavrou, N. Bozovic, D. Gritzalis, Proactive insider threat detection through social media: the youtube case, in *Proceedings of the 12th ACM Workshop on Privacy in the Electronic Society* (ACM, New York, 2013), pp. 261–266
26. M. Khalilian, N. Mustapha, Data stream clustering: challenges and issues (2010, Preprint). arXiv: 1006.5261
27. E.M. Knox, R.T. Ng, Algorithms for mining distance based outliers in large datasets, in *Proceedings of the International Conference on Very Large Data Bases*. Citeseer (1998), pp. 392–403
28. M. Kontaki, A. Gounaris, A.N. Papadopoulos, K. Tsichlas, Y. Manolopoulos, Continuous monitoring of distance-based outliers over data streams, in *2011 IEEE 27th International Conference on Data Engineering* (IEEE, Piscataway, 2011), pp. 135–146

29. P. Kranen, I. Assent, C. Baldauf, T. Seidl, Self-adaptive anytime stream clustering, in *Ninth IEEE International Conference on Data Mining, ICDM'09* (IEEE, Piscataway, 2009), pp. 249–258
30. P.A. Legg, O. Buckley, M. Goldsmith, S. Creese, Automated insider threat detection system using user and role-based profile assessment. IEEE Syst. J. **11**, 503–512 (2015)
31. P.A. Legg, O. Buckley, M. Goldsmith, S. Creese, Caught in the act of an insider attack: detection and assessment of insider threat, in *2015 IEEE International Symposium on Technologies for Homeland Security (HST)* (IEEE, Piscataway, 2015), pp. 1–6
32. M. Mayhew, M. Atighetchi, A. Adler, R. Greenstadt, Use of machine learning in big data analytics for insider threat detection, in *MILCOM 2015–2015 IEEE Military Communications Conference* (IEEE, Piscataway, 2015), pp. 915–922
33. MOA Team, T.U.o.W., Massive online analysis open source framework. https://moa.cms.waikato.ac.nz/. Accessed 14 Feb 2018
34. P. Moriano, J. Pendleton, S. Rich, L.J. Camp, Insider threat event detection in user-system interactions, in *Proceedings of the 2017 International Workshop on Managing Insider Security Threats* (2017)
35. J.R. Nurse, P.A. Legg, O. Buckley, I. Agrafiotis, G. Wright, M. Whitty, D. Upton, M. Goldsmith, S. Creese, A critical reflection on the threat from human insiders—its nature, industry perceptions, and detection approaches, in *International Conference on Human Aspects of Information Security, Privacy, and Trust* (Springer, Cham, 2014), pp. 270–281
36. P. Parveen, Z.R. Weger, B. Thuraisingham, K. Hamlen, L. Khan, Supervised learning for insider threat detection using stream mining, in *2011 IEEE 23rd International Conference on Tools with Artificial Intelligence* (IEEE, Piscataway, 2011), pp. 1032–1039
37. P. Parveen, N. Mcdaniel, Z. Weger, J. Evans, B. Thuraisingham, K. Hamlen, L. Khan, Evolving insider threat detection stream mining perspective. Int. J. Artif. Intell. Tools **22**(05) (2013). https://doi.org/10.1142/S0218213013600130
38. Reuters, Harold marin theft of ip. http://www.reuters.com/article/us-usa-cybersecurity-nsa-contractor-idUSKBN15N2N4. Accessed 14 Feb 2018
39. T. Seidl, I. Assent, P. Kranen, R. Krieger, J. Herrmann, Indexing density models for incremental learning and anytime classification on data streams, in *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology* (ACM, New York, 2009), pp. 311–322
40. J.A. Silva, E.R. Faria, R.C. Barros, E.R. Hruschka, A.C. De Carvalho, J. Gama, Data stream clustering: a survey. ACM Comput. Surv. (CSUR) **46**(1), 13 (2013)
41. A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, S. Robinson, Deep learning for unsupervised insider threat detection in structured cybersecurity data streams (2017) arXiv preprint arXiv:1710.00811
42. J. Verble, The NSA and Edward Snowden: surveillance in the 21st century. ACM SIGCAS Comput. Soc. **44**(3), 14–20 (2014)
43. S. Walton, E. Maguire, M. Chen, Multiple queries with conditional attributes (qcats) for anomaly detection and visualization, in *Proceedings of the Eleventh Workshop on Visualization for Cyber Security* (ACM, New York, 2014), pp. 17–24
44. D. Yang, E.A. Rundensteiner, M.O. Ward, Neighbor-based pattern detection for windows over streaming data, in *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology* (ACM, New York, 2009), pp. 529–540
45. A. Zargar, A. Nowroozi, R. Jalili, Xaba: a zero-knowledge anomaly-based behavioral analysis method to detect insider threats, in *2016 13th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)* (IEEE, Piscataway, 2016), pp. 26–31
46. T. Zhang, R. Ramakrishnan, M. Livny, Birch: an efficient data clustering method for very large databases, in *ACM Sigmod Record*, vol. 25 (ACM, New York, 1996), pp. 103–114