Adaptive One-Class Ensemble-based Anomaly Detection: An Application to Insider Threats

Diana Haidar and Mohamed Medhat Gaber
School of Computing and Digital Technology Birmingham City University Birmingham, United Kingdom
Email: {diana.haidar,mohamed.gaber}@bcu.ac.uk

***Abstract*—The malicious insider threat is getting increased concern by organisations, due to the continuously growing number of insider incidents. The absence of previously logged insider threats shapes the insider threat detection mechanism into a one-class anomaly detection approach. A common shortcoming in the existing data mining approaches to detect insider threats is the high number of False Positives (FP) (i.e. normal behaviour predicted as anomalous). To address this shortcoming, in this paper, we propose an anomaly detection framework with two components: one-class modelling component, and progressive update component. To allow the detection of anomalous instances that have a high resemblance with normal instances, the one-class modelling component applies class decomposition on normal class data to create k clusters, then trains an ensemble of k base anomaly detection algorithms (One-class Support Vector Machine or Isolation Forest), having the data in each cluster used to construct one of the k base models. The progressive update component updates each of the k models with sequentially acquired FP chunks; segments of a predetermined capacity of FPs. It includes an oversampling method to generate artificial samples for FPs per chunk, then retrains each model and adapts the decision boundary, with the aim to reduce the number of future FPs. A variety of experiments is carried out, on synthetic data sets generated at Carnegie Mellon University, to test the effectiveness of the proposed framework and its components. The results show that the proposed framework reports the highest F1 measure and less number of FPs compared to the base algorithms, as well as it attains to detect all the insider threats in the data sets.**

I. Introduction

Anomaly detection tackles the rare-class problem by building a model based only on the normal class label, then predicting whether acquired new data is normal or anomalous. This class of anomaly detection algorithms is referred to as semi-supervised anomaly detection in [1]. However, semi-supervised classification refers to those techniques that operate having only a small set of labelled instances along with a typically larger set of unlabelled ones, regardless of the assigned class [2]. Thus, in this paper, we use the term anomaly detection to refer to the one-class machine learning problem.

The ongoing monitoring and logging of the insiders' activities establishes huge useful data sets of information to learn the normal baseline of behaviour. However, the absence of previously logged malicious insider threats, among the logs of normal users' behaviour in an organisation, shapes the insider threat detection mechanism into a one-class data mining approach, namely anomaly detection. The topic of insider threat detection is getting increased concern by organisations, as a result of the significant number of malicious insider threats reported in recent years [3]. These threats are attributed to insiders; current or former employees, contractors, or business partners in an organisation, who have privileged access to the network, system, and data. The risk of the malicious activities carried by insiders is worth a considerable attention more than that of outsiders, due to the horrendous costly corruption that it causes to an organisation. Most of the security mechanisms implemented in an organisation usually tackle the outsider attacks (e.g. anti-viruses, firewalls, intrusion prevention and detection systems), which fortunately reduces the risk of outsiders. However, the privileges given to insiders make the detection of the malicious insider threats more challenging. The users within an organisation are aware of the system and have authorised access to sensitive information, which, if disclosed, will result in costly consequences.

The machine learning approaches proposed for detecting insider threats still have a common shortcoming, which is the great number of false alarms flagged [4], [5], deceiving the administrator(s) about suspicious behaviour of many users. This consumes a valuable time from the administrator's schedule, while investigating the suspected users. On the other hand, it impacts the level of trust between the suspected users and their senior executives in an organisation. A recent Real-time Anomaly Detection In Streaming Heterogenity (RADISH) system, based on k Nearest Neighbours was proposed in [4]. The experimental results showed that 92% of the alarms flagged for malicious behaviour are actually benign. We attribute such high number of False Positives (FPs) to the fact that previously proposed machine learning methods attempt to find all suspicious behaviours (instances) associated with any possible insider threat. However, the focus should be on detecting one or more instance(s) of malicious behaviour(s) per threat. Designing machine learning methods with such a relaxing condition, we argue, has the potential to reduce the frequent false alarming problem.

Taking into consideration this relaxing condition, and to tackle this shortcoming of the high number of FPs, in this paper, we propose an anomaly detection framework that consists of two components: one-class modelling component, and progressive update component. First, the role of the one-class modelling component ramifies to decompose the normal class data into k clusters, and to train an ensemble of k

base anomaly detection algorithms (One-class Support Vector Machine –ocsvm– or Isolation Forest –iForest–). Each base model of the ensemble is trained over one cluster, resulting in an ensemble of k base models. This allows the detection of anomalous *trapper* instances that have a high resemblance with normal instances, which would not be detected by the algorithm if trained over the whole normal class data. Second, the role of the progressive update component is to adapt the decision boundary of each base model with sequentially acquired FP chunks. It includes a selective oversampling method to generate artificial samples for FPs per chunk, with the aim to reduce the number of FPs.

The proposed anomaly detection framework provides the following major contributions:

- a class decomposition method on the normal class data to detect the anomalous *trapper* instances;
- a progressive update method with sequentially acquired FP chunks to address the shortcoming of high number of FPs;
- an outlier-aware artificial oversampling method for FPs to avoid model overfitting by intelligently adapting decision boundaries of base models of the ensemble fed with the synthetically oversampled data; and
- a thorough performance evaluation and a statistical significance test of a variety of experiments utilising ocsvm and iForest, validating the effectiveness of class decomposition, progressive update, and oversampling, compared to that of base ocsvm and base iForest.

The rest of the paper is organised as follows. In Section II, we review the related work on the anomaly detection approaches, including ocsvm and iForest approaches, for insider threat detection. In Section III, we propose an anomaly detection framework, with a detailed description of its components. In Section IV, we describe the utilised data sets [6], and we present the experimental setup and the refined versions of evaluation measures. In Section V, we evaluate a variety of experiments, to assess the effectiveness of the proposed framework and its components. Finally, we conclude our paper with a summary in Section VI.

II. Related Work

The approach to address the insider threat problem depends on whether the organisation historically collected system and network logs of the users' activities. If the data is available, it would either consist of normal instances, or normal instances with insider threat instances based on whether insider attacks previously occurred in the organisation. In this paper, we focus on the cases when data logs for only normal behaviour are available. Based on this, the insider threat problem may be addressed from the perspective of anomaly detection. In the following, we give the related work on the anomaly detection approaches for insider threat detection, including the approaches that utilised ocsvm and iForest.

Zargar et al. [7] introduced a Zero-Knowledge Anomaly-based Behavioural Analysis (XABA) method that learns each user's behaviour from raw logs and network traffic in real-time. Gates et al. [8] used the structure of the file system hierarchy and access similarity measure techniques to build user behaviour profiles and detect anomalous behaviour.

To our knowledge, the ensemble-ocsvm proposed by Parveen et al. [5] is the only approach that utilised ocsvm to classify data into normal versus anomalous for detecting malicious insider threats. It acquires data chunks (e.g. daily logs) of a continuous data stream, where it learns a new model for each chunk, and continuously updates the ensemble with the k models having the minimum prediction error. The results showed the superiority of ocsvm over two-class Support Vector Machine (two-class SVM) in terms of detecting threats. Furthermore, the ensemble-based ocsvm with the updating stream concept achieves better performance than ocsvm with no updating. The authors extended their work in a follow-up paper [9], where the ensemble approach was applied on unsupervised Graph-Based Anomaly Detection (GBAD). The ensemble-ocsvm outperformed ensemble-GBAD in terms of FPs.

A recent framework based on a graph approach and iForest was presented by Gamachchi et al. [10] to isolate suspicious malicious insiders from the workforce. The graph approach extracts graph and subgraph properties based on user activities as well as time dependent features to generate input for iForest. iForest then calculates anomaly scores to separate anomalous behaviour of a user from normal behaviour, without profiling normal behaviour.

A further recent unsupervised ensemble-based anomaly detection system named PRODIGAL was presented in [11]; a result of five years work on the insider threat detection problem [12], [13], [14]. iForest is configured as one of the user-day detectors in PRODIGAL to detect complex insider threat scenarios in real user activities.

The above methods have shown merit in addressing the insider threat detection problem, however, as aforementioned, they do suffer from high false alarms. In this paper, we design an approach aiming at false alarm reduction, adopting a number of proposed methods. Details of the proposed approach are given in the following section.

III. Anomaly Detection Approach 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 presents the proposed anomaly detection framework for insider threat detection, and provides a detailed description of its components.

A. Insider Threat Feature Space

The first step to tackle the insider threat problem is to identify the feature space. In this paper, we utilised the synthetic data sets generated by Carnegie Mellon University - Community Emergency Response Team (CMU-CERT) [6], where different malicious insider threat scenarios are simulated. The data sets log the behaviour of users as system and

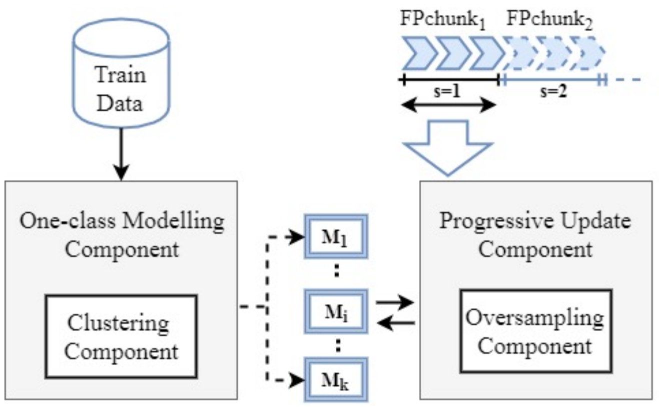


Fig. 1: Conceptual Framework.

network logs (e.g. logons/logoffs, connecting removable media devices, copying files, browsing websites, sending emails, etc.). Based on the literature [4], [15], we extract a feature set from these logs to represent the baseline of users' behaviour including malicious insider threat records. The features used in this work are categorised in four groups: frequency-based (frequency of logon, frequency of connecting device); time-based (logon after work hours, device usage after work hours); Boolean $flag$=$\{0, 1\}$ (non-empty email-bcc, email to a non-employee); attribute-based (browsing a particular URL *job websites, WikiLeaks*); and others (number of email recipients, number of attachments to emails, access to sensitive files based on *file extension*).

Based on the identified feature set, we create community behaviour profiles for users, such that each profile represents the behaviour of users having the same role (e.g. Salesman, IT admin) over session slots. A session slot defines the period of time from *start time* to *end time*, such that the behaviour logs of all users in the community during this period of time are used to extract the identified features. In this work, the session slot is defined per 4 hours, so that the feature set associated to each session slot maps to the community users' behaviour logged during the 4 hours from *start time* to *end time*.

B. Anomaly Detection Framework

In the following, we introduce the proposed anomaly detection framework with its components. Fig. 1 illustrates two phases: one-class modelling component, and progressive update component.

1) One-class Modelling Component: The one-class modelling component acquires training data in the initial modelling phase. It consists of a clustering component that applies k-means clustering algorithm on the normal class data in order to create k clusters. It then trains each cluster on a base algorithm. The result is an ensemble of a base algorithm over k clusters, resulting in k models. In this work, we utilised two highly performing anomaly detection algorithms: ocsvm and iForest, as base algorithms in the proposed framework to detect malicious insider threats. We adopted ocsvm and iForest, because each method has been utilised for insider threat detection, either as a base algorithm for the proposed approaches [5], [9], [10], [11], or as a benchmark against which the performance of a deep learning approach was compared to its performance [16].

a) Clustering Component: Malicious insiders have authorised access to the network, system, and data, and are aware of the system management and security policies. These aspects aid the malicious insider to deceive the detection system, where some anomalous behaviour may have a high resemblance with the normal user's behaviour. This manifests as local anomalous instances located among normal instances. To address this issue, we apply class decomposition [17] on the normal class data. The idea is to decompose the normal class data into clusters and train a detector per cluster, giving more opportunity for the detector to identify local anomalous instances with respect to a cluster which might not be detected over the whole data. We utilise k-means clustering algorithm to identify patterns in the normal class data, given its efficiency.

Let X^t=$\{x_1^t, x_2^t, ..., x_m^t\}$ represent the feature vector at session slot t, where $x_f^t; 1 \preceq f \preceq m$ represents the value of the f^{th} feature. Let y represent the normal class label. Each instance (i.e. feature vector) X^t either belongs or does not belong to normal class y. Let N=X^t $\forall t; X^t \in y$ represent the set of instances which belong to the normal class y. If we apply k-means clustering algorithm on the set N, then N decomposes into k clusters. Let C=$\{C_1, C_2, ..., C_k\}$ represent the set of k clusters.

Fig. 2 represents the normal instances with blue circles, and the anomalous instances with squares. Let the solid-line outer circle represent the decision boundary generated by the base algorithm over the whole normal data to separate the normal instances from the anomalous instances. Consider k=2, so that the normal data instances are grouped into 2 clusters. We construct an ensemble of k base algorithms and train a base algorithm on each cluster of the k clusters. Let the dash-line inner circles represent the decision boundaries generated by each cluster's base algorithm. Fig. 2 reveals *two types of anomalous instances* defined below:

- *Borderline and outlier instances*: represented by red filled squares. Those instances are located at the borderline with respect to the solid-line outer decision boundary, or are far outliers; and
- *Trapper instances*: represented by red empty squares. Those instances are located in a sparse or dense area of normal instances.

The borderline and outlier anomalous instances can be easily detected by one of the aforementioned base algorithms trained over the whole normal data. However, as shown in Fig. 2, the solid-line outer decision boundary would not be able to detect trapper instances. As aforementioned, the trapper instances are located among normal instances, and therefore the base algorithm would declare normal instances (e.g. iForest would not assign a high anomaly score).

To address this issue, we propose to decompose the normal class data into clusters and to train an ensemble of a base

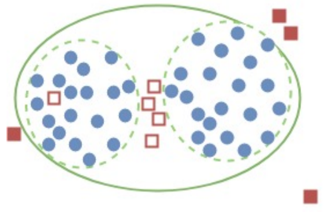


Fig. 2: Clustering normal instances.

algorithm per cluster. So that the trapper instances can be identified by the base algorithm(s) as anomalous with respect to cluster(s). Fig. 2 shows that the dash-line inner decision boundaries, generated by clusters, can detect the trapper instances located in the disjoint area (i.e. sparse area of normal instances).

Nevertheless, there would still exist some trapper instances inside the clusters which may not be detected as positives, due to their existence in an inner dense area of normal instances.

Upon decomposing the normal training data set into k clusters, the role of one-class modelling component ramifies into training an ensemble of a base algorithm on the k clusters to generate k initial models. Let M=$\{M_1, M_2, ..., M_k\}$ represent the set of models generated by the ensemble. Each initial model M_i for $C_i; 1 \leq i \leq k$ is then used to detect the malicious insider threats in the testing data set. The decision d_i^t for a testing instance X^t with respect to M_i comes in Boolean form of {True, False}. In case of ocsvm, an instance X^t either belongs or does not belong to the normal class y. In case of iForest, X^t is identified as an anomalous instance if its anomaly score is greater than a defined anomaly score threshold τ. The parameter τ requires to be tuned as examined later in Section IV.

After that, the ensemble acquires a set of decisions D^t=$\{d_1^t, d_2^t, ..., d_k^t\}$ for a testing instance X^t to vote whether X^t is normal or anomalous. Each decision $d_i^t \in D^t; 1 \leq i \leq k$ is taken by a model M_i for a cluster C_i in the ensemble. The voting mechanism is executed as follows: (1) If $d_i^t \in D^t$ votes for anomalous $\forall i$ (i.e. by all models), then the overall decision D^t declares X^t as anomalous behaviour, and consequently flags an alarm warning of a malicious insider threat; (2) If $\exists d_i^t \in D^t$ votes for normal, then the overall decision D^t declares X^t as normal behaviour.

2) Progressive Update Component: The importance of the insider threat problem requires a continuous monitoring of the implemented detection system by the system's administrator(s). A false alarm is a result of a normal behaviour detected as anomalous by the system. This maps to normal instances that have a high similarity with anomalous instances, thus appear as suspicious events. To address this issue and to minimise the number of false alarms raised, we introduce the progressive update component. The role of this component ramifies to progressively update the detection system with acquired FP chunks. We define an FP chunk as follows:

Definition III.1. FP Chunk An FP chunk is a segment of capacity c that accumulates test instances declared as FPs. Let FPchunk$_s$=$\{$FP^1, FP^2, ..., FP$^c\}$ represent an FP chunk acquired at sequence s, such that D^t for each FP$^t \in$ FPchunk$_s$; $1 \leq t \leq c$ is predicted as a positive (i.e. anomalous instance) while the actual class label for FP^t is normal. Thus, each FP^t is declared as FP.

Consider, in Fig. 1, the set of blue arrows represent the testing instances declared as FPs, and each segment of blue arrows represents an FP chunk acquired sequentially. For example, the segment of blue arrows at sequence s=1 represents FPchunk$_1$. We define c as the capacity (size) of FP chunks; each FP chunk FPchunk$_s$ can accumulate c number of FPs. Fig. 1 illustrates FP chunks of capacity c=3; each FP chunk accumulates 3 FPs.

The progressive update method relies on the continuous monitoring by the administrator to investigate whether the flagged alarms are true or false. Along the run of the detection system, when an alarm is flagged, the administrator is required to investigate the suspected user and decide whether it is a malicious insider threat or a false alarm. In the latter case, the FP (false alarm) is accumulated into the FP chunk. This procedure continues until the current FP chunk is full (i.e. capacity c of FP chunk is reached). The FP chunk is then fed to the progressive update component to oversample the FP instances in order to generate artificial samples. The oversampling component is later described in this section. The set of FP instances together with the set of artificial instances are then utilised to update the pre-generated models. Let N represent the pre-generated set of genuine normal instances, and let A_s represent the set of artificial instances generated for FPchunk$_s$. Hence, the pre-generated models are retrained on R=$N\cup$ FPchunk$_s \cup A_s$. Similarly, this process is repeated progressively for each accumulated FP chunk.

The rationale behind the progressive update method is not simply to retrain the models with new instances, however, to enrich the models with recently detected FPs and synthetically generated artificial samples *close (not replicates)* to FPs. So that the decision boundary in each model adapts to the falsely detected behaviour and minimises the chances of FPs in the upcoming testing instances.

The idea of continuous monitoring to investigate flagged alarms to identify FPs has been applied in [16]. The authors defined cumulative recall measure based on a *daily budget*. The term *daily budget* refers to maximum number of alarms an analyst can investigate per day to judge whether it is TP or FP.

a) Oversampling Component: The method of updating pre-generated models with FP instances in the FP chunks may not have a significant influence on the decision boundary. However, the oversampling of FP instances generates artificial instances, and enriches the updated model with recently acquired FP instances as well as normal artificial samples. In this way, the recently acquired normal behaviour of a user or a community will be well represented, and in turn will trigger the base algorithm to adapt the model's decision boundary.

The oversampling component is in charge of generating artificial samples from the FP instances upon the acquirement of each FPchunk$_s$. The task of allocating the number of samples to be generated for each FP instance in the FPchunk$_s$ depends on the degree of outlierness of each FP instance. Thus, the number of samples to be generated varies among FP instances. We utilise a density-based method, namely, Local Outlier Factor (LOF) [18], to calculate the local outlier factor (score) lof_N^t for each FP instance FP^t in acquired FPchunk$_s$ with respect to the k nearest neighbours from only the set of genuine normal class instances N. lof_N^t is tuned for k=$\sqrt{1+card(N)}$ (thumb-rule), where the radicand $1+card(N)$ represents the number of instances utilised to calculate lof_N^t. The motivation to integrate LOF to calculate the anomaly score for FP instances, though iForest can provide it, is that LOF can be used independently of the base algorithm (ocsvm or iForest).

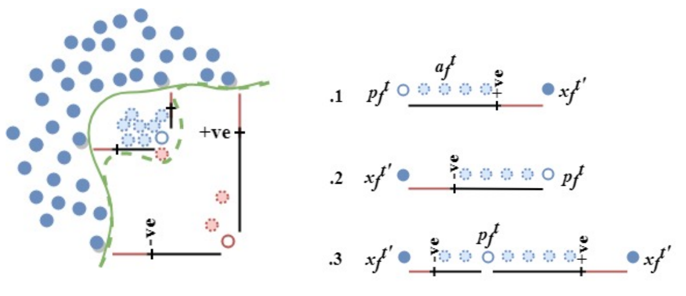
Let $perclof_N^t$ represent the percentile rank for each FP^t compared to the set of normal instances N. In Fig. 3a, we represent the genuine normal instances N by blue filled circles. We define two *types of FP instances*, where each type is oversampled based on its degree of outlierness lof_N^t:

- *Outlier instance*: represented by a solid-line red empty circle. Each FP instance FP^t is considered an outlier instance, if it has a **high** lof_N^t value; located far away from the genuine normal instances N. For example, if $perclof_N^t$=90, this means that the lof_N^t for FP^t is greater than 90% of the normal instances N.
- *Safe instance*: represented by a solid-line blue empty circle. Each FP instance FP^t is considered a safe instance, if it has a **low** lof_N^t value; located at or near the borderline of genuine normal instances N.

In Fig. 3a, the dash-line red circles represent the artificial samples generated for FP outlier instances, while the dash-line blue circles represent the artificial samples for FP safe instances. The idea is that safe instances are given more chance to generate artificial samples around them, while the outlier instances are given less chance. This gives the update component more conservative control on the adaptation of the decision boundary. Oversampling more safe instances than outlier instances safeguards the system from fast movement of the decision boundary due to outliers. Otherwise, more False Negatives (FN) (i.e. anomalous instances predicted as normal) in the upcoming FP chunks.

Let $perc.over$ represent the percentage of artificial samples to be generated, and let $numS$=$(perc.over/100)\times c$ represent the number of artificial samples to be generated. The process of generating artificial samples associated to each FP$^t \in FPchunk_s$ instance is executed feature wise over a number of iterations.

Recall that $x_f^{t'}$ represents the value of the f^{th} feature of X$^{t'}$ at session slot t'. Likewise, let p_f^t represent the value of the f^{th} feature of FP^t at session slot t, given that FP^t=$\{p_1^t, p_2^t, ..., p_m^t\}$. For each feature $f; 1 \preceq f \preceq m$, we find the nearest neighbour



(a) Over two features. (b) Over one feature.

Fig. 3: Artificial oversampling of FP instances.

$x_f^{t'}$ of $X^{t'} \in N$ for p_f^t of FP^t. In other words, we search the set of normal instances N at the level of feature f only, and we find the closest feature $x_f^{t'}$ for p_f^t. At the level of feature f, there exists two directions: positive ($+ve$), and negative ($-ve$). Thus, p_f^t may have (1) only $+ve$ neighbours from the set N, (2) only $-ve$ neighbours, or (3) both $+ve$ neighbours and $-ve$ neighbours. We define the positive ($+ve$) nearest neighbour as follows:

Definition III.2. Positive nearest neighbour A $+ve$ nearest neighbour is the closest $x_f^{t'}$ of $X^{t'} \in N$ for p_f^t of FP^t, such that $x_f^{t'}$ is located in the $+ve$ direction to p_f^t.

Similarly, the **negative** ($-ve$) **nearest neighbour** is defined. Fig. 3b illustrates generating artificial samples of FP instances over one feature (i.e. one dimension). Let the blue filled circle represent $x_f^{t'}$, the blue empty circle represent p_f^t, and the dash-line blue circle represent an artificial feature value a_f^t associated to p_f^t. The process of generating an artificial feature value a_f^t is executed as follows:

- If p_f^t has only a $+ve$ nearest neighbour at the level of feature f (Fig. 3b.1), then a_f^t is calculated in the $+ve$ direction along the segment joining p_f^t and $x_f^{t'}$ according to Eq. 1, such that dir=$+1$.
- If p_f^t has only a $-ve$ nearest neighbour at the level of feature f (Fig. 3b.2), then a_f^t is calculated in the $-ve$ direction along the segment joining p_f^t and $x_f^{t'}$ according to Eq. 1, such that dir=-1.
- If p_f^t has both a $+ve$ nearest neighbour and a $-ve$ nearest neighbour at the level of feature f (Fig. 3b.3), then a_f^t can be calculated in the $+ve$ or $-ve$ direction. A random direction dir is selected at each iteration, and a_f^t is calculated in the selected direction dir according to Eq. 1.

$$a_f^t = p_f^t + dir \times rand(0 : \lambda \times dist(p_f^t, x_f^{t'})) \qquad (1)$$

Note that λ, tuned for λ=.8, denotes a parameter that controls the distance permitted to generate artificial features along the segment joining p_f^t and $x_f^{t'}$. The rationale behind this is to generate the artificial samples a bit closer to the FP instances and not the normal instances, so that the adapted decision boundary is influenced by these samples.

Consequently, an artificial sample A^t=$\{a_1^t, a_2^t, ..., a_f^t\}$ associated to an FP^t instance is generated at each iteration. The

TABLE I: Definition of Experiments.

E-ocsvm	E-iForest	**E**nsemble
ocsvm-U	iForest-U	progressive **U**pdate
E-ocsvm-U	E-iForest-U	**E**nsemble+**U**pdate
ocsvm-OU	iForest-OU	**U**pdate+**O**versampling
E-ocsvm-OU	E-iForest-OU	**E**nsemble+**U**pdate+**O**versampling

steps described are repeated for a number of iterations until $numS$ of artificial samples are generated.

IV. Experiments

The experiments are conducted on the CMU-CERT data sets in Windows Server 2016 on Microsoft Azure (RAM $140GB$, OS $64-bits$, CPU Intel Xeon $E5-2673v3$). First, MATLAB $R2016b$ was used to preprocess the data sets and generate community behaviour profiles per session slots of 4 hours. Second, we implemented the experiments in R environment ($R-3.4.1$).

A. Description of the Dataset

For this paper, we used $r5.2$ data set from the insider threat data sets generated by CMU-CERT. This data set 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 4 different scenarios. Among these employees, we extracted the data logs for employees belonging to the following communities: Production line worker (com-P) with 17 malicious insider threats, Salesman (com-S) with 22, and ITAdmin (com-I) with 12. More information regarding CMU-CERT data sets [19] and simulated scenarios can be found in [6].

B. Experimental Setup

In Table I, we define a variety of experiments performed using the aforementioned anomaly detection algorithms. Each experiment is set up to ocsvm or iForest base algorithms with or without the following: ensemble method, progressive update method, and oversampling method. To evaluate the methods, we performed 10 fold cross-validation on each of the utilised communities.

The experiments are tuned for different values of parameters. ocsvm is evaluated for the $kernel$ values: Linear L, Polynomial P, and Radial R. On the other hand, iForest is evaluated for different values of anomaly score threshold τ=$\{.35, .4, .45\}$, given the number of iTrees nt=20 and the subsample size psi. Note that no anomalous instances were isolated for $\tau > .5$ over the utilised CMU-CERT data sets. This reflects the complex patterns of malicious insider threat behaviour in the data set, and the high resemblance of anomalous behaviour with normal behaviour. Although the authors in [20] pointed out the efficiency of subsampling, psi is tuned for the whole sample size without extracting a subsample of the data set. It is either set up to $card(N)$ over the whole normal training data set, or to $card(C_i)$ over each cluster in the ensemble case. The rationale of using the whole sample size in both cases is related to the progressive update method. It assures that all the FP instances in the progressively acquired FP chunks are used to update the iForest model.

Regarding the ensemble method, we tuned the number of clusters k over a set of arbitrarily chosen small values k=$\{2, 4\}$, as the goal is to detect the anomalous trapper instances. However, the results for only k=2 are reported, due to revealing better performance than k=4. The capacity of FP chunks is tuned over the values c=$\{20, 40, 60, 80, 100\}$ to evaluate the effectiveness of updating the model with *early-acquired-early-updated FPs*. Regarding the oversampling method, the oversampling percentage is tuned for only $perc.over$=200 to test the influence of applying oversampling to the FP chunks on progressive update method.

C. Evaluation Measures

The ultimate aim of the proposed framework is to detect all the malicious insider threats, while minimising the number of FPs. We utilised the following measures to evaluate the experiments. First, we define *default* versions of measures that are evaluated per instance (behaviour): **FP** designates the number of normal instances (behaviours) that are detected as anomalous instances. FP is evaluated with respect to the whole testing set (not per chunk); and TN designates the number of normal instances predicted as normal.

Second, we define *refined* versions of measures that are evaluated per threat: **TP**$_T$ is used instead of TP to evaluate the number of threats detected by the system among all the P$_T$ malicious insider threats. TP$_T$ is incremented if at least one anomalous instance (behaviour associated to the threat) is predicted as anomalous; and FN$_T$ is used instead of FN to evaluate the number of insider threats not detected. Note that the rationale behind introducing refined versions of some measures is related to the ultimate aim of the framework, which is to detect all threats (not necessarily all behaviours), but to minimise FPs (all false alarms).

As a result, the **F1** measure is defined based on the values of the above defined measures. Note F1 is not close to 1 due to the use of refined versions of some measures, but this does not reflect low performance. However, knowing that the maximum TP$_T$ (evaluated per threat) is much less than the minimum FP (evaluated per behaviour), the defined F1 measure closer to 0.5 reveals *significant* performance.

V. Results and Discussion

In this section, we present the results of ocsvm experiments and iForest experiments in terms of the pre-defined evaluation measures. We then show the merit of the proposed framework according to the following objectives:

- Testing the statistical significance of the results using Wilcoxon Signed-Rank Test;
- Comparing ocsvm experiments and iForest experiments;
- Assessing the influence of the proposed components on the framework; and
- Assessing the time efficiency of the progressive update component.

A. Results

Fig. 4 presents the variation of F1 measure as a function of FP chunk capacity c for ocsvm and iForest based experiments over the communities. The results are reported with respect to $kernel$ values and anomaly score threshold τ values for ocsvm based experiments and iForest based experiments respectively. Tables III and IV present the minimum FP and the maximum TP$_T$ attained respectively for ocsvm and iForest based experiments over communities. It reports $kernel$ and FP chunk capacity c associated to the minimum FP and maximum TP$_T$ attained for ocsvm based experiments. On the other hand, it reports anomaly score threshold τ and c associated for iForest based experiments. This allows to identify the superior $kernel$ or τ, and to analyse the influence of FP chunk capacity c on the progressive update method.

1) ocsvm Experiments: Over com-P, ocsvm-U outperforms ocsvm as well as all other ocsvm based experiments in terms of F1 measure, FP, and TP$_T$. ocsvm-U achieves the maximum F1=**0.261**;L while reducing the false positives to the minimum FP=**76**;L compared to FP=106 in ocsvm. Knowing that P$_T$=17, ocsvm-U detects all the malicious insider threats TP$_T$=17; P without missing any threat.

Over com-C, E-ocsvm-U and E-ocsvm-OU report better performance $\forall kernel$, where E-ocsvm-U achieves the maximum F1=**0.318**;P with the minimum FP=**89**;P compared to FP=120 in ocsvm. E-ocsvm-OU follows it with F1=0.300;L and FP=90;P. However, E-ocsvm-U attains the maximum TP$_T$=22;P out of 22, unlike the latter which missed one threat.

Over com-I, E-ocsvm-U reports the best performance in terms of all measures compared to other ocsvm based experiments. It achieves the maximum F1=**0.246**;P, while minimising the number of false positives to FP=**52**;P knowing that FP=149 in ocsvm. Furthermore, it attains TP$_T$=12 out of P$_T$=12 (same TP$_T$ for other experiments).

2) iForest Experiments: Over com-P, E-iForest-U and E-iForest-OU gave a significant boost to F1 measure for τ=.45, where it reaches F1=**0.365**,0.337 respectively. E-iForest-U reduces the number of false positives to the minimum FP=**50**; .45, however, it detects TP$_T$=16; .35, thus missing one threat. On the other hand, E-iForest-OU reaches a minimum FP=57; .45, while detecting all the malicious insider threats TP$_T$=17; .35.

Over com-C, E-iForest-U reports the best performance in terms of F1 and FP compared to other iForest based experiments. It achieves the highest F1=**0.422**; .45, and the minimum FP=**49**; .45. However, it reports TP$_T$=21; .35, thus missing the detection of one malicious insider threat. It is worth to note that the iForest based experiments with oversampling method (with or without the ensemble method) attain to detect all the threats for τ=.35

Over com-I, E-iForest-U shows s significant performance in terms of all measures. It achieves the maximum F1=**0.358**; .45, while minimising the number false positives to FP=**43**; .45. It also attains the maximum TP$_T$=12; $\forall\tau$ (same TP$_T$ for other experiments).

TABLE II: Wilcoxon Signed-Rank Test at .05 significance level.

E-ocsvm	ocsvm-U	E-ocsvm-U	ocsvm-OU	E-ocsvm-OU
0.1073	6.101e-09	8.243e-06	8.587e-07	1.687e-06
E-iForest	iForest-U	E-iForest-U	iForest-OU	E-iForest-OU
0.2719	1.945e-04	4.104e-08	**0.1165**	1.986e-05

B. Statistical Significance: Wilcoxon Signed-Rank Test

To test the significance of the results, we use Wilcoxon Signed-Rank Test which compares each pair of experiments. Each ocsvm based experiment and iForest based experiment is compared with base ocsvm and base iForest respectively. Table II tabulates the *p-value* calculated for each experiment at .05 significance level. With respect to ocsvm, all ocsvm based experiments, except E-ocsvm ($0.1073 > .05$), are significantly different from base ocsvm where *p-value* $< .05$. With respect to base iForest, all iForest based experiments are significantly different from base iForest, except for E-iForest and iForest-OU. The results were predictable, because both E-ocsvm and E-iForest show low performance in terms of the evaluated measures.

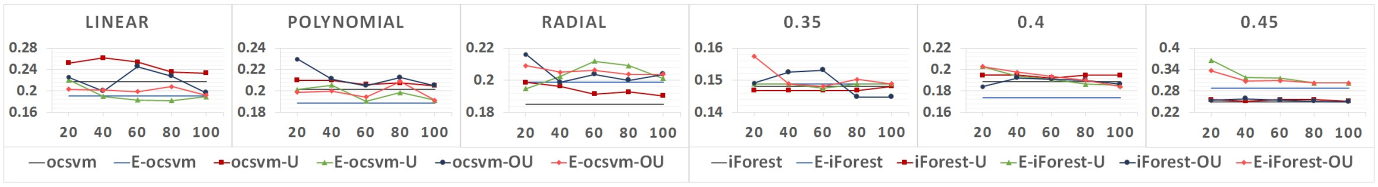
C. Comparing ocsvm Experiments and iForest Experiments

Finally, it is noteworthy that base iForest flagged a lower number of false alarms and reported a higher F1 measure compared to base ocsvm. Moreover, the best performing iForest based experiments over each community achieved the minimal FP compared to ocsvm based experiments. For instance, E-iForest attains the minimum FP=50; $.45, 20$, while ocsvm-U attains FP=76; $L, 40$, compared to FP=106 in base ocsvm over com-P. Over com-S, E-iForest-U reaches the minimum FP=49; $.45, 20$, while E-ocsvm-U and E-ocsvm-OU attain FP=89; $P, 60$ and FP=90 respectively. And lastly, E-iForest-U achieves the minimum FP=43; $.45, 20$, compared to FP=52; $P, 20$ in E-ocsvm-U over com-I. It is apparent that ocsvm works better for $kernel$=$\{L, P\}$, and iForest shows its best performance for τ=.45. Moreover, the less the capacity (size) c of FP chunk, the better the performance of the progressive update method. This emphasises the effectiveness of updating the model with *early-acquired-early-updated FPs* in minimising the number of FPs.

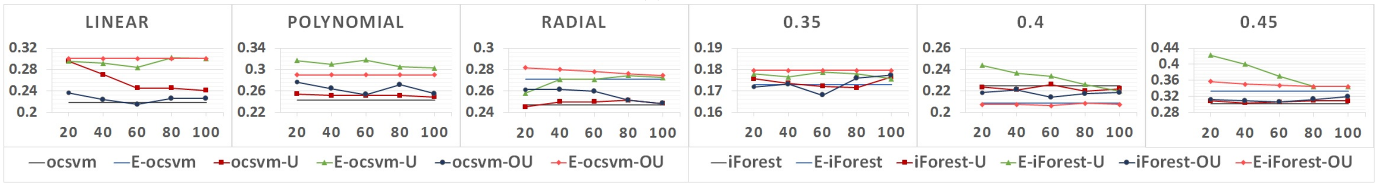
On a side note, the ensemble-ocsvm approach proposed by Parveen et al. [5] reports a percentage of FP rate (%FP=30%). However, our framework reports approximately the half, where %FP varies between (14%-18%) over all the utilised communities.

D. Influence of Proposed Components on the Framework

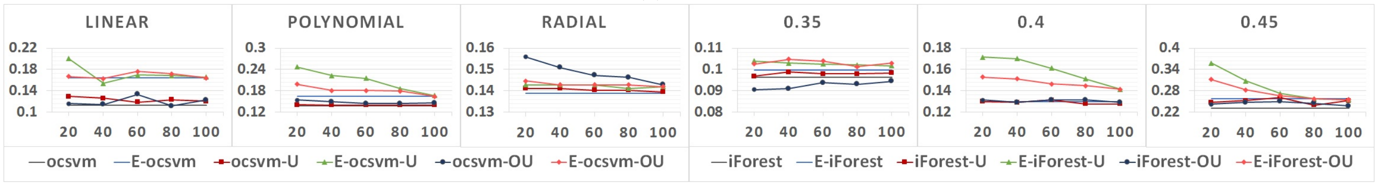
Based on the above extensive experiments carried out to evaluate the influence of the different components on the proposed framework, we sum up what follows. ocsvm-U, E-ocsvm-U, and E-ocsvm-OU outperform other ocsvm based experiments in terms of F1 measure, FP, and TP$_T$ $\forall$ communities. On the other hand, E-iForest-U and E-iForest-OU outperform iForest based experiments $\forall$ communities.



(a) com-P.



(b) com-S.



(c) com-I.

Fig. 4: The variation of F1 measure as a function of FP chunk capacity c for ocsvm and iForest based experiments over communities.

TABLE III: Minimum FP over Communities.

community	ocsvm	E-ocsvm	ocsvm-U	E-ocsvm-U	ocsvm-OU	E-ocsvm-OU
com-P	106 L	97 L	**76** L,40	88 L,20	83 L,60	95 L,80
com-S	120 R	90 P	99 L,20	**89** P,60	109 P,20	**90** P,20
com-I	149 R	100,$\{L, P\}$	146 R,$\{20, 40\}$	**52** P,20	132 P,20	97 P,20
community	iForest	E-iForest	iForest-U	E-iForest-U	iForest-OU	E-iForest-OU
com-P	88 .45	66 .45	85 .45,$\forall c \setminus 100$	**50** .45,20	84 .45,40	**57** .45,20
com-S	85 .45	73 .45	82 .45,$\{20, 100\}$	**49** .45,20	78 .45,100	70 .45,20
com-I	80 .45	69 .45	68 .45,60	**43** .45,20	72 .45,60	53 .45,20

TABLE IV: Maximum TP$_T$ of Detected Insider Threats over Communities.

community	ocsvm	E-ocsvm	ocsvm-U	E-ocsvm-U	ocsvm-OU	E-ocsvm-OU
com-P	**17** P	16 R	**17** P,$\forall c$	16 R,$\forall c \setminus 20$	**17** P,$\forall c$	16 R,$\forall c$
com-S	**22** P	21 R	**22** P,$\forall c$	**22** P,$\{20-40\}$	**22** L,$\{20, 60\}$ P, 80	21 R, $\forall$
com-I	**12** $\forall kernel$	**12** R	**12** $\forall kernel$,$\forall c$	**12** R,$\forall c$	**12** $\forall kernel$,$\forall c$	**12** P, 20 R,$\forall c$
community	iForest	E-iForest	iForest-U	E-iForest-U	iForest-OU	E-iForest-OU
com-P	16 .35	16 .35	16 .35,$\forall c$	16 .35,$\forall c$	**17** .35,$\{20-60\}$	**17** .35,20
com-S	21 .35	21 .35	21 .35,$\forall c$	21 .35,$\forall c$	**22** .35,$\forall c \setminus 60$	**22** .35,$\forall$
com-I	**12** $\forall \tau$	**12** $\forall \tau$	**12** $\forall \tau$,$\forall c$	**12** $\forall \tau$,$\forall c$	12 $\forall \tau$,$\forall c$	**12** $\forall \tau$,$\forall c$

The importance of the clustering component and the progressive update component as a whole is quite evident, where the number of FPs reached its minimal in experiments which utilised the ensemble method and the progressive update method with FP chunks (e.g., E-ocsvm-U, E-ocsvm-OU, E-iForest-U, E-iForest-OU). We deduce that the effectiveness of the proposed framework relies on the joint collaboration of both components. The use of the clustering component solely, in E-ocsvm and E-iForest, did not improve the performance significantly in terms of the evaluated measures. The Wilcoxon test supported this as revealed in Table II. However, the joint use of progressive update method with ensemble method showed outstanding performance in terms of minimising the number of FPs significantly and achieving higher F1 measure.

Furthermore, we can infer, from Table IV, the support of the oversampling method in detecting all the malicious insider threats over all communities without missing any threat. It is quite remarkable that E-ocsvm-OU and E-iForest-OU show competitive performance to E-ocsvm-U and E-iForest over com-S and com-P respectively. This reveals the potential of the oversampling component.

E. Time Efficiency of Progressive Update Component

Regarding the time complexity, the merit of the proposed progressive update method is its time efficiency. Let ta_s represent the time to accumulate an FPchunk$_s$. We hypothesise that the time to update the ensemble models with the current FPchunk$_s$ is much less than the time to accumulate FPchunk$_{s+1}$ (ta_{s+1}). This ensures that the progressive update of the models is synchronised with the accumulation of sequentially acquired FP chunks.

Recall that, an instance X^t from the acquired test instances is accumulated in the FP chunk, if X^t is declared FP. The FP chunk is progressively accumulated until the capacity c is reached. Hence, the time to accumulate an FP chunk FPchunk$_{s+1}$ actually depends on, (1) n: number of acquired test instances after FPchunk$_s$ until capacity c of FPchunk$_{s+1}$ is reached, and (2) $slot$: period of a session slot for a test instance. Analytically, ta_{s+1}=$n \times slot$.

Consider c=20 and $slot$=4 hours. If n=20, this means that c=20 FPs are accumulated in FPchunk$_{s+1}$ after 20 test instances are acquired. Thus, ALL the n acquired test instances are actually FPs. In this case, ta_{s+1}=20$\times$4=80 hours (approximately 3 days). This case represents the *worst case scenario*, where the next FPchunk$_{s+1}$ is accumulated while 80 hours (3 days) are available to update the models with the current FPchunk$_s$. 80 hours is more than enough to update the models using any state-of-the-art cloud facility. Nevertheless, it would require much less than 80 hours. Hence, the hypothesis is verified.

In the aforementioned worst case scenario, the progressive update is required to run after 80 hours. However, in a typical scenario where an FP chunk is accumulated after $n > c$ number of test instances is acquired, the progressive update will run much less frequently.

VI. Conclusion

The malicious insider threats are getting increased concern by organisations, due to the continuously growing number of insider incidents. A common shortcoming in the proposed detection approaches is the high number of FPs. In this paper, we address this shortcoming based on the availability of only data for normal behaviour, with no previously logged insider incidents.

We propose an adaptive one-class ensemble-based anomaly detection framework with a progressive artificial oversampling method of FPs in class decomposed data. First, a class decomposition method clusters the normal class data, so that an ensemble of k base algorithms (ocsvm or iForest) is trained over the clusters. This allows to detect anomalous *trapper* instances. Second, a progressive update method updates the pre-generated models with progressively acquired FP chunks. This allows the models to benefit from already declared FPs, in order to reduce FPs in upcoming data. Third, an oversampling method is integrated to progressively enrich the models with artificial samples of FPs.

We evaluate the proposed framework in a variety of experiments (with and without: class decomposition/progressive update/oversampling) using ocsvm and iForest. The results show the influence of each method on the performance in terms of higher F1 measure, lower FP, and detecting all malicious insider threats. Moreover, iForest based experiments report a better F1 and a lower FP compared to that of ocsvm.

References

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[2] X. Zhu, "Semi-supervised learning," in *Encyclopedia of machine learning*. Springer, 2011, pp. 892–897.

[3] J. R. Nurse, P. A. Legg, O. Buckley, I. Agrafiotis, G. Wright, M. Whitty, D. Upton, M. Goldsmith, and 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, 2014, pp. 270–281.

[4] B. Böse, B. Avasarala, S. Tirthapura, Y.-Y. Chung, and D. Steiner, "Detecting insider threats using radish: A system for real-time anomaly detection in heterogeneous data streams," *IEEE Systems Journal*, 2017.

[5] P. Parveen, Z. R. Weger, B. Thuraisingham, K. Hamlen, and L. Khan, "Supervised learning for insider threat detection using stream mining," in *2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*. IEEE, 2011, pp. 1032–1039.

[6] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *Security and Privacy Workshops (SPW), 2013 IEEE*. IEEE, 2013, pp. 98–104.

[7] A. Zargar, A. Nowroozi, and R. Jalili, "Xaba: A zero-knowledge anomaly-based behavioral analysis method to detect insider threats," in *Information Security and Cryptology (ISCISC), 2016 13th International Iranian Society of Cryptology Conference on*. IEEE, 2016, pp. 26–31.

[8] C. Gates, N. Li, Z. Xu, S. N. Chari, I. Molloy, and Y. Park, "Detecting insider information theft using features from file access logs," in *European Symposium on Research in Computer Security*. Springer, 2014, pp. 383–400.

[9] P. Parveen, N. Mcdaniel, Z. Weger, J. Evans, B. Thuraisingham, K. Hamlen, and L. Khan, "Evolving insider threat detection stream mining perspective," *International Journal on Artificial Intelligence Tools*, vol. 22, no. 05, p. 1360013, 2013.

[10] A. Gamachchi, L. Sun, and S. Boztas, "Graph based framework for malicious insider threat detection," pp. 2638,2647, 2017.

[11] 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.

[12] E. Ted, H. G. Goldberg, A. Memory, W. T. Young, B. Rees, R. Pierce, D. Huang, M. Reardon, D. A. Bader, E. Chow *et al.*, "Detecting insider threats in a real corporate database of computer usage activity," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 1393–1401.

[13] W. T. Young, H. G. Goldberg, A. Memory, J. F. Sartain, and T. E. Senator, "Use of domain knowledge to detect insider threats in computer activities," in *Security and Privacy Workshops (SPW), 2013 IEEE*. IEEE, 2013, pp. 60–67.

[14] W. T. Young, A. Memory, H. G. Goldberg, and T. E. Senator, "Detecting unknown insider threat scenarios," in *Security and Privacy Workshops (SPW), 2014 IEEE*. IEEE, 2014, pp. 277–288.

[15] P. A. Legg, O. Buckley, M. Goldsmith, and S. Creese, "Automated insider threat detection system using user and role-based profile assessment," *IEEE Systems Journal*, 2015.

[16] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," 2017.

[17] R. Vilalta, M.-K. Achari, and C. F. Eick, "Class decomposition via clustering: a new framework for low-variance classifiers," in *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE, 2003, pp. 673–676.

[18] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *ACM sigmod record*, vol. 29, no. 2. ACM, 2000, pp. 93–104.

[19] C. M. U. CERT Team, "Cmu cert synthetic insider threat data sets," https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099, [Online; accessed 04-April-2018].

[20] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 2008, pp. 413–422.