Received 14 June 2023, accepted 3 July 2023, date of publication 10 July 2023, date of current version 17 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3293825

RESEARCH ARTICLE

Applying One-Class Algorithms for Data Stream-Based Insider Threat Detection

RAFAEL BRUNO PECCATIELLO, JOÃO JOSÉ COSTA GONDIM, AND LUÍS PAULO FAINA GARCIA
Department of Computer Science, University of Brasília, Brasília 70910-900, Brazil

Corresponding author: Rafael Bruno Peccatiello (rafael.peccatiello@aluno.unb.br)

This work was supported in part by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brazil (CAPES)—Finance Code 001.

ABSTRACT An insider threat is anyone who has legitimate access to a particular organization's network and uses that access to harm that organization. Insider threats may act with or without intent, but when they have an intention, they usually also have some specific motivation. This motivation can vary, including but not limited to personal discontent, financial issues, and coercion. It is hard to face insider threats with traditional security solutions because those solutions are limited to the signature detection paradigm. To overcome this restriction, researchers have proposed using Machine Learning which can address Insider Threat issues more comprehensively. Some of them have used batch learning, and others have used stream learning. Batch approaches are simpler to implement, but the problem is how to apply them in the real world. That is because real insider threat scenarios have complex characteristics to address by batch learning. Although more complex, stream approaches are more comprehensive and feasible to implement. Some studies have also used unsupervised and supervised Machine Learning techniques, but obtaining labeled samples makes it hard to implement fully supervised solutions. This study proposes a framework that combines different data science techniques to address insider threat detection. Among them are using semi-supervised and supervised machine learning, data stream analysis, and periodic retraining procedures. The algorithms used in the implementation were Isolation Forest, Elliptic Envelop, and Local Outlier Factor. This study evaluated the results according to the values obtained by the precision, recall, and F1-Score metrics. The best results were obtained by the ISOF algorithm, with 0.78 for the positive class (malign) recall and 0.80 for the negative class (benign) recall.

INDEX TERMS Insider threat detection, data stream, machine learning, one-class classification.

I. INTRODUCTION

Insiders refer to anyone who has owned or provided access to information from an organization (including personnel, facilities, equipment, networks, and systems). Moreover, insider threat is the potential that an insider has to use their authorized access in a harmful way [1]. This harm can include malicious, complacent, or unintentional acts that could weaken the organization's integrity, confidentiality, and availability.

Insider threats raise the detection complexity because they are individuals who have activated hostile activities, such as information theft and sabotage, without the need to bypass the protections of the environments and systems [2]. These agents are equally harmful compared to external malicious agents, with the advantage that they do not need the same technical capacity to perform their activities, making their attacks seem like normal work activities [3].

According to the Cost of Insider Threat, Global Report [4], the average time to contain an incident caused by insider threats is 85 days, and 12% of the reported incidents were contained in less than 30 days. In recent years, insider threat incident frequency has increased. In 2018, 53% of the companies surveyed had this type of incident. In 2020, that number rose to 60%, and in the current year, 67% of the companies surveyed reported suffering damage related to international threats [4]. Even with the increase in occurrences, insider

The associate editor coordinating the review of this manuscript and approving it for publication was R. K. Tripathy.

threats remain less common than external ones, but the damage caused by an internal agent can have just as serious effects. This is due to the high level of access the malicious insider has to information and systems. As a result, insiders often require less effort and technical knowledge to perform malicious actions [5].

Insider threats differ from external threats in terms of access to information terms, as internal users, who perform malicious activities, usually have legitimate access to systems, network infrastructure, and information. They also know the organization's critical assets and security policy [6]. As the insider's malicious activities will cause little or no interference detectable by conventional network infrastructure security systems [7], the detection methods must be better designed. Furthermore, the malicious activities performed by an internal user are proportionally much smaller than the sum of the benign activities performed by the same user and the others. The perception of what would be malicious or normal behavior within the network ends up overshadowing due to the imbalance between the types of behaviors [7].

According to Saxena et al. [8], internal malicious agents can be classified into three types:

- **Malicious:** A person who intentionally abuses his access privileges to perform harmful actions to the organization for personal gain. An example would be a disgruntled employee who deliberately sells privileged information.
- **Compromised:** Someone who has been the target of some malicious attack, the result of which is the compromise of personal information that provides access to internal systems to third parties. For example, there are cases of employees being a victim of social engineering, bribery, or spear phishing.[1]
- **Careless:** They are people who are not concerned about or are unaware of the security standards in force in their organization, so they make mistakes that lead to data exposure and, hence, damage to the organization. For example, there are situations where employees write down their credentials and do not save them properly, and at some point, they end up falling into the third party's hands.

Conventional security solutions such as firewalls, Intrusion Prevention Systems (IPS), Intrusion Detection Systems (IDS), and antivirus programs have restrictions when dealing with insider threats. This happens because these solutions aim to detect or prevent external malicious actors, implementing a detection paradigm centered on signatures and rules [9]. Because of that, researchers are proposing Artificial Intelligence (AI), and Machine Learning (ML) methods [10], [11], [12], [13], [14] to respond to insider threat events as those methods provide conditions for creating models capable of learning, updating and adapting to ever-changing attack scenarios [15]. Among the proposals, researchers mention using supervised [7], [11], [12], [14], [16] and unsupervised ML techniques [7], [14], [16], [17], varying between batch [9], [11] and stream approaches [10], [14], [17].

In batch learning, all data is available for training the algorithm, which generates a decision model after processing the data [18]. The traditional batch learning process occurs over a finite dataset with a stationary distribution [19]. This characteristic may not be suitable for developing insider threat detection methods that are scalable and effective [20]. That is because the behavior of the insider threat network traffic continuously changes as new insider threat cases emerge [4]. Data generated by insiders fits best into a data type named stream, which is characterized by being generated sequentially, indefinitely, presenting or not changes in its distribution over time (concept drifts), and which may or may not have a temporal relationship [21]. Due to these characteristics, approaches that handle insider threat detection as a data stream analysis [10], [13], [14] seem to be the most promising for real-world implementations [22].

To support this research, version 4.2 of the Insider Threat Dataset (ITD), published by the Insider Threat Center (ITC) of the Computer Emergency Response Team (CERT) Division of the Software Engineering Institute (SEI) at Carnegie Mellon University, was selected. This research chose ITD due to the lack of datasets with current data composed of real scenarios of incidents caused by insider threats [23]. The dataset was generated by ITC using information from its database about insider threats collected since it was created [24]. The positive points of the dataset are the variety of scenarios, the available attributes, the detailed documentation, the extensive use by the community of researchers of the subject [23], and the fact that it comes from one of the richer databases of insider attacks [6]. Thus, according to the bibliographical [6], [9], [10], [11], [23], [25] research carried out and after analyzing the dataset documentation, this work considered that the ITD would be the best choice for providing a richer range of information. The way that most of the cited authors manipulate the dataset is similar, as they usually extract new attributes from the original ones. The dataset shows a marked imbalance between activities considered benign (negative class) and malign (positive class), which is inherent to the very nature of the problem.

This work applied a semi-supervised learning approach through one-class classification algorithms to overcome the ITD imbalance issues. The semi-supervised learning assumes that for the classifier training, there are only samples of one class. In the present study, the benign class is the well-known one. While executing the classifier, a new class sample different from the trained one may appear [26]. The trained model must identify these samples by deriving a classification boundary that may separate the known objects from the unknown ones. Thus, semi-supervised learning can be applied, for the binary classification of data streams or innovative class identification, without knowing all other types of samples (classes) [26]. Moreover, we do not need an entire labeled dataset such as supervised learning, which is

[1]https://www.enisa.europa.eu/topics/csirts-in-europe/glossary/phishing-spear-phishing

very hard and costly to obtain. This characteristic limits the application of supervised-based solutions in real life [27].

This article proposes a framework for a real-world insider threat detection system. It implements a detection system with the framework characteristics and tests it with a synthetic dataset. The system uses one-class classification algorithms, stream data processing, and periodic retraining to deal with concept drift. This study treated insider threat detection as an anomaly detection problem because the dataset contains highly imbalanced classes, where the majoritarian one corresponds to benign samples. The features used in the research dataset derive from the original ITD, as observed in the researched references. The research dataset simulates a continuous data stream. This work uses supervised and semi-supervised learning because getting enough labeled anomaly samples is hard in the real world and deploys a multi-class classification algorithm for comparison purposes. The performance is analyzed through activity detection, scenario detection, and malicious user detection. The results show if the retraining procedure offers a real gain in the model performance.

This work intends to deepen the study of internal threat detection because this problem does not have a definitive solution in the academic community. The main contributions of this work are: (*i*) to gather techniques available in the bibliographic review to propose a new framework to detect insider threats in the scenarios of the analyzed dataset (ITD); (*ii*) to test the proposed framework implementation in a dataset designed for insider threat detection (ITD); and (*iii*) to conclude on the viability of implanting a system based on the proposed framework.

This document comprises five more sections. Section II presents the related work with a more detailed view of the nuances involving insider threat detection and the state of art used in detecting these threats. Section III-A addresses the proposed insider threat detection framework. Section IV presents the carried-out experiment details. Section V shows the results and their analysis. Finally, Section VI presents the research conclusions and future works.

II. BACKGROUND

This section presents a review of some works related to this study. AI and ML papers commonly address Insider Threat Detection in two ways: batch approach (Section II-A) and stream approach (Section II-B). Finally, Section II-C discusses the related works.

A. BATCH APPROACH

Some studies [5], [7], [9], [11], [12], [16], [28] used batch learning to address insider threat detection. The problem with batch implementations is that they do not correspond to real scenarios. Although some of these studies have good results, their implementations move away from the application in real-world scenarios.

Gavai et al. [16] applied supervised and unsupervised learning techniques in their study. The unsupervised one used the Isolation Forest (IF) algorithm applied to a private real-world dataset with artificially introduced insider threat events. The real-world part of the dataset comes from an endpoint audit and investigation software. The study obtained an AUC metric value of 0.77. For supervised learning, they applied the Random Forest (RF) algorithm, and the value obtained for accuracy was 73.4%. The authors did not address the treatment of imbalances in the dataset.

Kim et al. [28] used the properties of Markov Chains to list the user behaviors over time and to classify to which state they belong, malign or benign. The study used the original ITD to generate sequential datasets corresponding to each user's activities. These datasets were generated according to the influence of n occurrences of Markov attributes (states created according to the user's activities to compose the Markov chain) and classified by ML algorithms. The study makes a random selection of users present in the ITD to carry out its experiments, using it partially. The algorithms used were Naive Bayes (NB), Support Vector Machine (SVM), Multilayer Perceptron (MLP), and RF. All the algorithms obtained close results for the metrics Accuracy, False Positive Rate (FPR), recall, and F1-Measure, with values ranging from 0.96 to 0.97. The only exception was the NB algorithm which obtained inferior results.

Le and Zincir-Heywoo [7] used supervised and unsupervised approaches. The unsupervised approach applied Self Organizing Maps (SOM) and Hidden Markov Models algorithms. The supervised one used the C4.5 Decision Tree (DT). This study used ITD version 4.2 and summarized the user's activities by day and week. The metrics used were Detection Rate (DR), False Positive Rate (FPR), and Accuracy, which obtained the following best results respectively 82.51, 0.01, and 99.93. These results were produced through supervised learning using per-day summarized data and only an extract from the dataset. The study disregarded all data from the first twenty weeks as they contained only benign activities.

Another study that used ITD version 4.2 was Chattopadhyay et al. [11] which performed an analysis based on the scenarios provided by the ITD version 4.2. In this study, the authors used an approach based on sliding windows applied to a new dataset derived from the original, composed of statistical attributes set. They established specific features for each of the three insider threat scenarios implemented in the dataset. Thus, they perform a scenario-based detection using a specific set of features for each scenario detection. In this study, the authors balanced training data by reducing the majority class samples and used a deep autoencoder neural network as a classification algorithm. The detection of the scenario that concerns the user who steals organizational information obtained the best results. For this scenario, the experiment reached 92.88 for precision, 99.48 for recall, and 96.06 for the F1-score. They applied a 40-day sliding training window and balanced it using the Synthetic Minority Oversampling Technique (SMOTE) to achieve these values. In this specific detection, the minority class had more samples in the training set than the majority class in the proportion of 2:1.

Johannessen [9] used unsupervised learning in his study by implementing an anomaly detection mechanism based on the IF, Elliptic Envelope (EV), and Local Outlier Factor (LOF) algorithms. He used the batch approach and treated the imbalance through the configuration of the hyper-parameters related to the imbalance rate of the algorithms used, except for the LOF. The dataset was organized so that there were at least five malign samples in the training data. He used version ITD version 4.2. This study suggests using information from the most diverse sources of organizational data, such as physical access to locations and analysis of psychosocial components described by the personality trait analysis method called BIG FIVE. This method makes it possible to describe a person's personality by examining the following personality traits: openness, conscientiousness, extraversion, agreeableness, and neuroticism. Because this study explores different nuances of insider threat detection, it did not examine the ML results in depth. Nevertheless, this is a very informative and comprehensive study that offers multiple perspectives on the problem and focuses on developing a solution.

Kim et al. [5] proposed an insider threat detection framework based on user behavior modeling. The study created three sets of distinct data, based respectively on activities carried out by users at their workstations, the email exchange networks between users, and the content of emails exchanged between users. This study used ITD version 6.2 and evaluated the Gauss, Parzen, K-Means, and PCA algorithms as one-class classification algorithms. Thus, the study attempts to mitigate the effects of class imbalance as such algorithms could train with a substantial benign sample amount and then classify data with malign ones. The authors used 90% of the randomly selected benign data for the training set. The test set consisted of 10% of the remaining benign data plus all malign ones. Lastly, the authors created an anomaly ranking for each dataset to classify each activity sample. Using the proposed framework was possible to detect 53.67% of the malign activities in the top 1% of the ranking of malign activities and 90% in the top 30% of the malign activities.

A methodology for processing organizational log data was presented by Hall et al. [12] in their study using ITD version 6.2. The study addressed the problem with a supervised approach and summarized the dataset according to criteria related to time and users. They first aggregated user activity data by day and after by users. After the dataset was ready, they built a model to identify each malign scenario separately. For compiling the training set, the study only used data from one month and data from users with the potential to run some of the attack scenarios available. In addition, they undersampled the majority class until the imbalance reached a ratio of 15:1. With these measures, the authors aimed to reduce the disproportionality presented by the ITD between malign and benign activities and between malign and benign users. The study used the Neural Network(NN), Naive Bayesian Network (NBN), SVM, RF, DT, and Logistic Regression (LR) classifiers to compose single models and compared the obtained results with the best Boosting version of such classifiers (the study used the NBN Boosting version in the comparison). The study concluded that the model that used the composition of classifiers was superior, obtaining an AUC of 0.988 against an AUC of 0.980 obtained by the Boosting version of the NBN.

B. STREAM APPROACH

Insider Threat Detection has been treated as a stream problem by some authors [10], [14], [17], [29]. These authors have used stream ML techniques to deal with possible deviations from concepts and the scarcity of samples from the malign class. Despite being more complex, this approach is more realistic and enables the creation of models capable of being applied in real scenarios once insider data is stream generated.

Senator et al. [13] used private data from real enterprise networks to research insider threat detection. They installed specialized programs on the workstations of the participating companies to collect the study data. The malicious scenario data was inserted synthetically by a team of specialists in offensive cybernetic actions. The study presents an insider threat detection system using anomaly detection algorithms based on: suspected scenarios of insider threat behavior; unusual activity indicators; high-dimensional statistical patterns; temporal sequences; and evolution graphs. As their best result, they mention an AUC of 0.979 and cite studying the feasibility of applying their method in a real scenario as future work. Unlike the present study, the work by Senator et al. [13] focuses on specific scenarios discovery and does not cite any presence or need for the treatment of imbalanced datasets. Moreover, as their dataset is not public, it is impossible to reproduce their study.

Parveen et al. [14] compared supervised and unsupervised approaches for insider threat detection from a data stream mining point of view, using the Lincoln Laboratory Intrusion Detection dataset [30]. Their work demonstrated the superiority of the supervised one. The study also points out that ensemble algorithm implementations present better results than non-ensemble ones. Although the supervised approach performs best, Parveen et al. [14] report that it was hard to implement real-world detection with supervised-based mechanisms. Because insider threat data is a stream of unbounded length and constantly evolving, supervised deployments are often inadequate. As an alternative, the study proposes using unsupervised learning. To this, the authors implemented an ensemble of graphic-based anomaly detection models. This ensemble is kept up-to-date by an ensemble update process which also aims to allow the model to be resilient to concept drifts. This kind of implementation is more suitable in real-world scenarios.

Krawczyk and Wozniak [26] proposed using one-class classifiers with incremental learning and forgetting for data streams with concept drift. Their study mentions three approaches to deal with concept drift: the first is to retrain the classifier every time new data is available; the second is trying to detect concept drift occurrence and then retrain the classifier; and the last is adopting incremental learning techniques to, smoothly, adapt the model to the data stream concept drifts. The third approach is noteworthy because it can combine different forgetting mechanisms and retraining procedures aiming to adjust the model to the data stream's natural changes. They presented a system that uses a weighted sample mechanism to build the retraining dataset based on an initial one and a forgetting sample mechanism to discard old samples. The model analyses the data stream in chunks of data. The study uses five datasets, two synthetic and three real-life ones. Although not directly related to insider threat detection, this work has contributed to data stream analysis.

A real-time unsupervised system was created from a Deep Learning-based model by Tuor et al. [17] to filter the system log data from ITD version 6.2. As insider threat behavior varies widely, they were concerned with modeling the recognition of the characteristics of each user on the network using Deep NN and Recurrent NN. Thus, they tried to determine whether the behavior was habitual or anomalous. The results obtained by the NN algorithms were compared with those obtained by IF, SVM, and PCA algorithms and showed the superiority of the NN ones. The study focused on creating and analyzing profiles for each user based on their role within the company.

The work presented by Bose et al. [10] proposes a system called Real-Time Anomaly Detection in Heterogeneous Data Streams (RADISH), which has two main components. The first is the RADISH-L which is responsible for learning behaviors by defining new models and thresholds for normal behaviors based on the data stream analysis. The second is the RADISH-A component which is responsible for comparing the new streams with the defined threshold. To do this, models created by the RADISH-L are applied to the new streams, generating alarms for those that exceed the previously defined normality threshold. The recall achieves around 0.5 and a precision of 0.08. The study made a self-justifying calculation between the losses experienced by companies that are victims of insider threats and the number of false positives. The study suggests that it could be financially advantageous to verify a substantial number of false alarms than to spend resources to respond to an incident not previously alarmed.

Finally, Zhang et al. [29] proposed a framework for resampling unbalanced datasets. Resampling techniques were used in sets from data streams, aiming to reduce the impact of class imbalance and concept changes. Among the various used techniques, they mention the creation of a minority class buffer of samples, intending to insert them in the training dataset due to their scarcity and periodic retraining procedure to keep the model up to date. This technique will be reproduced in the present study but applied in the insider threat detection context.

C. RELATED WORK

This study gathered some techniques from the works mentioned in Section II to develop a framework applicable to insider threat detection in real-network environments. This study used the batch approach in the first steps of its insider threat detection proposal. The batch implementation helped to define the first parameters of our models. To that end, this study implemented a Grid Search algorithm over a training set. The data regarding insider activities must be analyzed at its generation time, as in the case of the data stream, aiming to produce timely alarms. The data streams have some peculiarities such as the concept drifts, which the solutions must handle. Analyzing data generated by insiders as a batch or chunk could lead to delayed notifications.

Starting from batch works, Johannessen [9] sought to make maximum use of the various sources of information provided by the dataset for feature extraction. The study of Chattopadhyay et al. [11] was the reference for some of the features used in this work, mainly those considered relevant to detect all the scenarios because in the mentioned study the detections were individualized by scenarios. Kim et al. [5] influenced the use of one-class algorithms. This type of algorithm is able to perform its training with only one class and detect the presence of samples that are different from those trained. Thus, the high availability of samples from the benign class in the ITD dataset will favor the one-class algorithm's training, which may increase malign class detection.

From stream references, as applied in the study carried out by Krawczyk & Wozniak [26], this work proposed some techniques to adapt the model incrementally to the data stream changes. Among them, we can mention procedures to retrain the algorithm, build a retraining dataset, and discard samples. Differently from Tuor et al. [17], the present study focuses on the abnormal-behavior discovery, evaluating the information contained in user sessions without distinguishing them. As in the present study, Bose et al. [10] and Senator et al. [13] address the insider threats problem through a data stream analysis. Bose et al. [10] also used the ITD but in version 2, Senator et al. [13] used a private one. However, they did not address the dataset imbalance issue in their studies. Bose et al. [10] updated their models training them with a training set composed of recent benign samples and a buffer that stored old and new malign samples. The present study also does that, but instead of maligns saves the benign ones. Parveen et al. [14] concluded that their supervised experiment would not occur in the real world. Their experiments demonstrate that applications based on data streams are more suitable for real-world applications. Finally, Zhang et al. [29] used new arriving instances to update their models dynamically as proposed by this study. The following Section will present how this study used cited techniques to build a framework for insider threat detection.

III. METHOD

This section will present the proposed framework, with details about its offline and online phases. The main objective is to show the framework and its implementation details to clarify how it works.

A. INSIDER THREAT DETECTION FRAMEWORK

The framework proposed in this study comes from the combination of approaches used in previous studies and aims to allow the detection system deployment in a real-world environment [5], [10], [11], [13], [26], [29]. This environment requires the handling of the premises imposed by the problem. The most significant are the following [31]: (*i*) process one example at a time and inspect only once; (*ii*) use a limited amount of memory; (*iii*) analyze a limited period; and (*iv*) be ready to predict at any time. As well as these premises, there are some obstacles also imposed by the insider threat detection problem such as the extreme imbalance of class distribution [32] having the malign class as the minor one, the need to accurately identify this class, the possibility of concept drift occurrence, and the similarity between the data generated by malign and benign insiders [14].

The present study set up a hybrid analysis approach to satisfy these premises, with an offline batch phase for algorithm first training and an online stream phase for retraining and sample classification. It also implements some incremental learning aspects. To perform its retraining tasks, the model must be able to select new samples, implement a forgetting mechanism to discard the old ones, and adapt algorithms parameters. Figure 1 illustrates the framework overview.

The pre-processing module represents the first stage of the system that deals with feature extraction from the data sources. In real-life implementations, each data source must have its parser. In this work, handling the dataset log files abstracted the feature extraction step.

The model evaluation module sequentially classifies the samples as they arrive and perform the periodic retraining of the classifier. During this step, the model chooses which samples will be part of the training set. At each new training sample insertion, the model removes the oldest one. This work performs periodic retraining tasks to keep the model resilient to concept changes that may occur during data generation. The samples included in the retraining use the labels predicted by the model after its classification.

The first batch training of the model evaluation module takes place offline with known benign samples and with a deliberately inserted contamination. This work inserted the contamination because it is unlikely that samples originating from real-network communications do not contain any malign activity. It is a consequence of the similarity of actions performed by malign and benign insiders [14]. The objective of the offline phase is to establish the initial hyperparameters of the algorithms that the model will use.

In the online phase, the model predicts each stream sample as soon as received. To conduct the evaluation, the model uses one-class classification algorithms. The model saves the

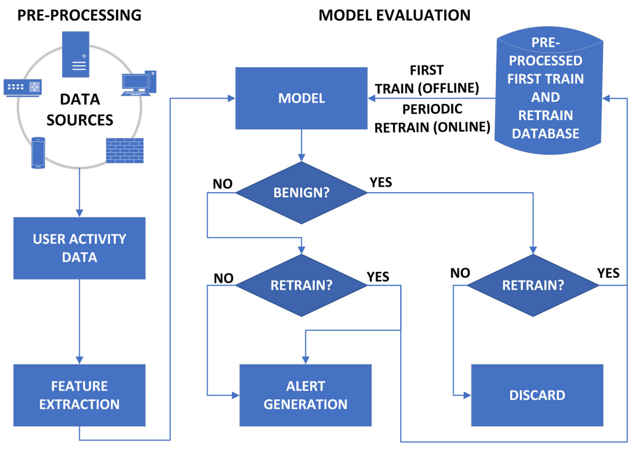

FIGURE 1. Framework overview.

predictions and their scores to support the retraining process. Still, in the online phase, the retraining takes place. The model implements a retraining mechanism for constantly adapting the model to possible concept drift. The model compiles the retraining sets using the first training set as a starting point. The retraining set is formed by replacing old samples from the first training set with new ones from the data stream analyzed by the model. During its execution, the model evaluates sample scores and labels and chooses those that will be part of the retraining set. The sample scores and labels are those provided by the algorithms during the stream analysis.

After the algorithms classify a sample and the model evaluates its pertinence to compose the training set, it is discarded or reused. The model will discard samples in two situations: (*i*) every time the algorithms classify it as benign, and the model does not select it to compose the retraining set, or (*ii*) every time it is replaced by a new sample in the retraining set. The model will reuse samples whenever they are considered benign or malign, and the model selects them to compose the retraining set. Every time the algorithm classifies a sample as malign, the model will use it to generate an alert.

B. MODEL EVALUATION

The next Sections will present details about the offline and online phases of the framework represented in Figure 2. This figure shows a flowchart of the model evaluation where the offline phase provides the initial hyperparameters of the algorithms and the initial training dataset as inputs for the online one. The online phase depends on the results of the offline phase to start its execution. The offline phase executes once and the online one executes indefinitely.

1) OFFLINE PHASE

The offline phase's main objective is to provide the values of the initial hyperparameters to be used by the algorithms in the model. To this, the present study performs a Grid Search algorithm implementation over a labeled training set.

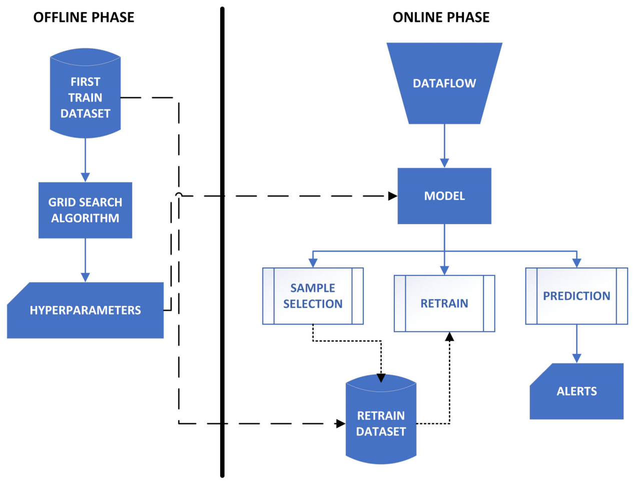

FIGURE 2. **Model evaluation overview.**

The Grid algorithm searches for the best combination of a hyperparameter subset [33]. Each algorithm has its own hyperparameters subset. The Grid Search relies on the score obtained by a metric to choose the best hyperparameter set. In this article, the reference metric was the average recall between malign and benign classes. The algorithm calculates the metric values for each label and the mean between them. The higher the average between the recall values of the malign and benign classes, the higher the individual value of each recall metric. We have used the contamination samples labels to perform the RF tests with Grid Search, as RF is a multi-class algorithm.

2) ONLINE PHASE

After defining the initial hyperparameters of the algorithms and having the algorithms trained, starts the online phase. This phase executes the selection of the samples for the build of the retraining set, the sample predictions, and the retraining procedures of the model.

The model uses the initial training set for building the retraining one. Before doing this, it separates the initial set into two sets according to the labels of your samples, benign and malign (contamination). The model constructs a sample score distribution of both sets to identify in what quartile the samples malign and benign likely are. After that, two thresholds are defined by the model, the higher and the lower one. When the algorithm starts to receive and classify the samples, the model compares each score obtained by each sample with the higher and lower thresholds in the following manner:

1) If the algorithm predicts a sample as benign and its score is higher than the higher threshold, it is inserted in the training set, and the model discards the oldest sample in the training set.
2) If the algorithm predicts a sample as malign and its score is higher than the lower threshold, it is inserted in the training set, and the model discards the oldest sample in the training set.
3) In any other case, the model does not reuse the classified sample.

The model will retrain the algorithm using the set resulting from the operation above. The idea is to take likely benign samples and the malign ones whose score is high, to compose the contamination in the retraining set. Both thresholds will always be positive due to the high imbalance of the dataset in favor of the benign class.

To retrain, the model checks whether the retraining interval has been reached. The retraining interval can be defined by time or the number of analyzed samples. This article calculates a time interval using the number of samples as a reference, 1,200 samples are equivalent to approximately 1 day in the ITD. Whenever the retrain interval checks return positively, the model redefines the higher and lower thresholds and retrains the algorithm using the new retraining set. The model redefines the thresholds using the same procedure explained in the second paragraph of this Subsection. The contamination is the only algorithm hyperparameter that may change in the retraining process. This happens according to the distribution of scores obtained by the training/retraining set samples. By using the training/retraining set score distribution of the malign (contamination) sample, the following comparison is made: If the 75 percentile value of the malign sample distribution of the training set is positive, the contamination hyperparameter value of the algorithm remains the same as predicted by grid search. Otherwise, the hyperparameter is set as automatic or with the default value, depending on the classifier.

It should be noted that although the model selects benign and malignant samples with their respective labels and scores to compose the retraining set, such a set is given to the algorithms to train without labels, as this work used one-class classifiers. Despite not using labels for training/retraining, the model saves them for building the distribution of scores for benign and malignant classes. Only in tests with the RF algorithm was it necessary to use labels.

IV. METHODOLOGY

Section IV will present the methodology followed to run the experiment. It will introduce the dataset, the way of feature extraction, and the execution of the experiments based on the framework presented in Section III-A.

A. THE DATASET

The dataset used in this study was the Insider Threat Dataset (ITD) [24], published by the Insider Threat Center (ITC) at Carnegie Mellon University.[2] This research selected the dataset by analyzing the researched bibliography on the subject. The selection criteria were the widespread presence of the dataset in research related to insider threat detection, the

[2]https://kilthub.cmu.edu/articles/dataset/Insider_Threat_Test_Dataset/12841247/1

diversity of informational domains present in the dataset, and the quality of the documentation provided with the dataset. The dataset consists of six files and one folder. All files have in common the fields *id* (event id), *date* (event date-time group), *user* (user id), and *pc* (personal computer id).

- ***LDAP***: Folder with Lightweight Directory Access Protocol log files (18 files).
- ***device.csv***: Removable device log file (405,380 lines). Particular field: activity.
- ***email.csv***: Email log file (2,629,979 lines). Particular fields: cc; bcc; from; size; attachment_count; content.
- ***http.csv***: Web browsing log file (28,434,423 lines). Particular fields: URL and content.
- ***logon.csv***: Users logon/logoff log file (854,859 lines). Particular field: activity.
- ***file.csv***: File handling log file (445,581 lines). Particular fields: filename and content.
- ***psychometric.csv***: User personality trait information file (1000 lines, not used).

The only folder provided by the dataset refers to the LDAP data. Inside this folder are files with monthly updated data about the employees. Each employee has the following attributes: *employee_name*, *user_id*, *email*, *role*, *business_unit*, *functional_unit*, *department*, *team*, and *supervisor*.

The dataset implements three scenarios of malign activities. The dataset documentation describes each scenario as shown below:

1) **Scenario 1:** User who did not previously use removable drives or work after hours begins logging in after hours, using a removable drive, and uploading data to wikileaks.org. Leaves the organization shortly thereafter.
2) **Scenario 2:** User begins surfing job websites and soliciting employment from a competitor. Before leaving the company, they use a thumb drive (at markedly higher rates than their previous activity) to steal data.
3) **Scenario 3:** The system administrator becomes disgruntled. Downloads a keylogger and uses a thumb drive to transfer it to his/her supervisor's machine. The next day, he/she uses the collected key logs to log in as his/her supervisor and sends out an alarming mass email, causing panic in the organization. He leaves the organization immediately.

The information in the files/directories simulates activities on a corporate network with 1,000 users over approximately 16 and a half months. Of the 1,000 users in the dataset, only 70 are responsible for some malicious activity. The malicious users are distributed across the scenarios as follows, 30 distinctive malicious users perform Scenario 1, 30 more perform Scenario 2, and 10 perform Scenario 3.

B. FEATURE EXTRACTION

The dataset has its information divided into days, but for the execution of the research, the features were extracted and organized by the user session. A session is considered the period between logon and logoff performed by a user. The ITD has some logon events without their respective logoffs. Therefore, it was necessary to insert implicit logoffs every time a user registered consecutive logon events on the same workstation without a logoff event interspersing these logon events. Another definition in the study, aiming for feature extraction, was the beginning and end time for work hours. This work defined these times by analyzing the behavior of logons and logoffs recorded in the dataset. Thus, it can be stated that, for the analysis carried out in the present study, the beginning work time is at 7:00 am, and its end time is at 5:00 pm. Below are the extracted features and their respective definitions.

- ***diff_begin_first***: Difference between beginning working time and first login.
- ***device_count_out***: The number of removable storage devices used out of work time.
- ***device_count***: The number of removable storage devices used.
- ***count_dwn_exe_file***: The number of .exe files downloaded.
- ***url_blocklist***: The number of blocklisted URLs accessed.
- ***unit_code***: User functional unit code.
- ***tfidf_jobsites***: TF-IDF comparison of job web pages corpus and web pages accessed by the users.

The dataset obtained after extracting the features has the following characteristics: 470,608 rows, 1,460 of which correspond to the malign class and 469,148 to the benign one. These values generate an imbalance ratio of around 1:321. This imbalance rate reflects how the problem presents itself in real life.

C. EXPERIMENTS

This paper performed two types of experiments, with and without retraining. The experiments without retraining were performed to state if the retraining procedures could improve insider threat detection. The non-retraining experiments also used the Grid Search algorithm and other characteristics addressed in this study less the retraining.

Given the characteristics of the model, for the execution of the experiments, some methodological parameters need to be defined. As the training occurs offline and after the model evolves continuously, it was required to configure the training set length and the interval in which the retraining will occur. The training/retraining set size does not vary during model execution. In this work, handling the dataset log files abstracted the feature extraction step.

As stated in Section III-A, the first training set (offline phase), which will be the base of the retraining sets generation in the online phase, was predominantly composed of benign samples and a little contamination of malign ones. Table 1 shows the contamination ratio of each initial training set size used in the experiments.

As previously mentioned, contamination simulates a common problem in real-life environments. In these

TABLE 1. Contamination ratio of the initial training sets.

First Training Set Size	Contamination Ratio
15 days	1:382
30 days	1:552
60 days	1:1,106
150 days	1:2,788

environments, it can be hard to state that logs come from activities completely free of malign actions. Besides that, the contamination is pertinent for the supervised multi-class implementation of the RF classifier. Methodologically, it is advisable to have a training dataset with samples from all classes when multi-class supervised classifiers are employed. The simulated contamination is possible because the one-class classifiers do not need labeled training sets.

The algorithms used in the experiment were ISOF [34], ELV [35], LOF [36], and RF [37]. The first three are one-class algorithms, and their main feature is that they can train with samples of only one of the classes present in the problem domain. After their training, in any situation, these algorithms can identify, among the tested samples, those that differ from the trained ones [38]. This study also used the RF implementation to compare the results of a supervised multi-class algorithm with those obtained by the one-class algorithms.

Before performing the algorithm's first training (offline phase), this experiment used the Grid Search algorithm to establish the initial hyperparameters for the algorithms. To do this, it applied the Grid Search algorithm to the initial training set, which contains samples of both classes due to contamination. The metric used as a benchmark was the macro recall. Having chosen this, the model tried to obtain the highest possible average value between the recall of the malign and benign classes. The objective is to find parameters that reach the highest rate of true positives for each one of the classes. The first training of the algorithm is performed only after defining its initial hyperparameters.

The Grid Search algorithm indicated the values below as the best for the hyperparameters of the classifiers. The other hyperparameters not cited remained with the default value.

- **ISOF:** 0.3 for the contamination; 40 for the n_estimators; bootstrap activate; and 3 for the max_features.
- **EV:** 0.5 for the contamination; and 1 for the support_fraction.
- **LOF:** 0.3 for the contamination; 20 for the leaf_size; novelty activate; mahathan for the metric; and 18 for the n_neighbors.
- **RF:** 200 for the n_estimators; entropy for the criterion; and 6 for the min_samples_leaf.

After the initial training (offline phase), the model starts receiving the sample lines of the test dataset one by one (online phase). By doing this, in addition to simulating a stream of information, it was possible to evaluate the performance of each of the classifiers. The speed of data analysis is a crucial factor in an environment with data generated and analyzed continuously. This study varied the training dataset size and the retraining intervals as part of the experiments. All these changes aim to establish ideal values for these methodological parameters and to make conclusions about the system application in a real-world scenario. To do this, the importance of these parameters to the model performance should be known, and thus decide whether or not to update them during the execution of the model.

During its execution, the experiments varied the training/retraining window size values to 15, 30, 60, and 150 days and the retraining interval to 15, 60, and 120 days. The size of the training/retraining set is informative to know how many samples are needed for this set to become representative of the stream data. The length of the retraining interval is crucial to know if the model can adapt to concept changes that may be present in the data stream. The experiments that did not apply the retraining only had the first offline training, initializing sample predictions immediately afterward.

The metrics analyzed were precision, recall, and F1-score because they would be the best when dealing with highly unbalanced datasets whose classification is binary [11]. The precision metric seeks to identify among everything classified as positive by the model, which ones were classified correctly. The recall indicates, among the number of true positives, how many of them were predicted by the algorithm. The F1-score is the harmonic mean between precision and recall, this metric provides a unified representation of them [39]. Beyond those, this article analyzed the kappa and geometric mean (gmean) metrics. The former minimizes the class imbalance problem as it considers the class distribution in its calculation [40] and the latter is independent of class distribution and suitable for assessing classifier performance in imbalanced data sets [41].

This study also analyzed the true positive rate (TPR), intending to know the proportions of malicious class correct predictions. The TPRs were analyzed by scenario and in a general way.

V. RESULTS

To perform the experiments this study defined the size of the training/retraining set and the interval in which the retraining will occur. The experiments without retraining needed only the definition of a training set length. Training/retraining set sizes range from approximately 15, 30, 60, and 150 sample days. The retraining intervals vary between 15, 60, and 120 days. The best results are those in which the values between the recall of positive and negative classes are high and balanced with each other. That is because we want to avoid excessive wrong predictions for both classes. Given the imbalance of the data set, we found that in all situations where the recall of the positive class is high, the false negative results are also high. This fact in real life ends up camouflaging the positive results among a large amount of false positive results. The opposite can also be stated, because every time we have many correct predictions of the negative class, we drastically increase the false negative results, in this case, the model misses the prediction of the positive class. The gmean metric is also important to be checked because it gives a unified vision about the recall and the true negative rate (TNR), the higher the value, the better performance of the classifier.

Tables 2, 3, 4 and 5 show the results obtained by the algorithms used in the experiments. Table 2 shows the results obtained without performing the online retraining procedure. In this specific table, the columns labeled 15, 30, 60, and 150 represent only the length of the dataset used in the application of the Grid Search and the first training of the algorithm. Table 3 shows the results with the methodological metric, retraining period, configured to occur every 15 days. Next, Table 4 shows the results with the retraining period set to every 60 days. Finally, Table 5 shows the values with the retraining period set to every 120 days. The table's first column represents the algorithm used in the experiments, the second shows the metric names, and the other columns show the values. The columns labeled 15, 30, 60, and 150 represent the length of the training/retraining dataset. The retraining intervals and training/retraining set length are expressed in days and are approximations based on one day number of samples. The results in bold represent the best recall results obtained by each algorithm. In some cases, the algorithms can have two of their results marked in bold. This is due to the proximity between two results obtained by an algorithm in different situations.

Table 2 shows the results using a similar approach mentioned in Section IV, but without applying the retraining procedure. The main difference between this and the results using retraining is the drastic reduction of false positives, even with an increase in malign class detection. This reduction is relevant in the context of security and is related to working with a noisy solution. Excessive false alarms may delay the identification of insider threats or, in the worst case, result in their non-identification. True alarms can disappear among the false ones, making it even harder to identify insider threats. This problem is recurrent in the domains of information security and has already been described by Gheyas et al. [3] in the context of insider threat detection. As the benign class represents 287,860 of 289,230 samples, its proportionality reduction is much more significant than the malign class. For example, in absolute numbers, 10% of benign samples represent 28,923 samples, and the same percentage of the malign class represents 142 samples. For the best results of all scenarios, the retraining procedure reduced the difference between the recalls of malign and benign classes. As an example, taking the number of false positive alarms of the best results, with and without retraining, produced by the three algorithms, it can be observed that ISOF generates 57,770 of this kind of alarm executing the retraining against 157,352 without it. The difference is near one hundred thousand samples and reaches more than double those presented in the best case with retraining.

The tables show that the ISOF algorithm obtained the best recall result among the algorithms used, reaching **0.80** for the benign class and **0.78** for the malign one. This study considered that ISOF was the best because it obtained the best balance between benign and malign class recall metrics, reaching the highest value for gmean metric **0.79**. IOSF also was the only one that identified all malicious users with at least one on alert for each one. Moreover, ISOF was the less noisy algorithm.

The best result obtained by the ISOF used the retraining interval corresponding to approximately two months of activities (72,000 samples). In this case, the training/retraining set length was approximately two months of activities (72,000 samples). These are the best configuration of methodological parameters for this algorithm.

For the EV algorithm, the best methodological values were five months for the training/retraining set length and two months for the retraining interval. The values for class 0 and class 1 recall were 0.76 and 0.71, respectively. The EV algorithm was the fastest. It was, on average, nine times faster than ISOF and slightly faster than LOF.

The LOF algorithm had its best value using five months for training/retraining set length and five months for retraining interval. The recall values with this configuration were 0.76 for the benign class and 0.58 for the malign class.

This study used the RF algorithm to find out how a multi-class algorithm would behave when subjected to the difficulties that involve the problem of detecting insider threats, mainly concerning the imbalance of the dataset.

TABLE 2. Metrics obtained without retraining.

			Training Set Length (days)			
Algorithm	**Class**	**Metric**	**15**	**30**	**60**	**150**
ISOF	Benign	precision	1.00	1.00	1.00	1.00
		recall	0.68	**0.67**	0.66	0.63
		F1-score	0.81	0.80	0.80	0.77
	Malign	precision	0.01	0.01	0.01	0.01
		recall	0.87	**0.88**	0.88	0.89
		F1-score	0.03	0.03	0.03	0.02
	Benign/Malign	kappa	0.01	0.01	0.01	0.01
		gmean	0.76	**0.76**	0.76	0.74
EV	Benign	precision	1.00	1.00	1.00	1.00
		recall	0.48	**0.47**	**0.47**	0.45
		F1-score	0.65	0.64	0.64	0.62
	Malign	precision	0.02	0.02	0.02	0.01
		recall	0.93	**0.93**	**0.93**	0.94
		F1-score	0.02	0.02	0.02	0.02
	Benign/Malign	kappa	0.01	0.01	0.01	0.01
		gmean	0.66	**0.66**	**0.66**	0.64
LOF	Benign	precision	1.00	1.00	1.00	1.00
		recall	0.77	0.77	0.76	**0.73**
		F1-score	0.87	0.87	0.64	0.84
	Malign	precision	0.01	0.01	0.01	0.01
		recall	0.54	0.50	0.51	**0.59**
		F1-score	0.02	0.02	0.02	0.02
	Benign/Malign	kappa	0.01	0.01	0.01	0.01
		gmean	0.64	0.62	0.62	**0.65**
RF	Benign	precision	1.00	1.00	1.00	1.00
		recall	1.00	1.00	1.00	1.00
		F1-score	1.00	1.00	1.00	1.00
	Malign	precision	0.00	0.00	0.00	0.00
		recall	0.00	0.00	0.00	0.00
		F1-score	0.00	0.00	0.00	0.00
	Benign/Malign	kappa	0.00	0.00	0.00	0.00
		gmean	0.00	0.00	0.00	0.00

TABLE 3. Metrics obtained by running the retraining every 15 days.

		Training/Retraining Set Length (days)				
Algorithm	**Class**	**Metric**	**15**	**30**	**60**	**150**
ISOF	Benign	precision	1.00	1.00	1.00	1.00
		recall	**0.81**	0.80	0.80	0.79
		F1-score	0.89	0.89	0.89	0.88
	Malign	precision	0.02	0.02	0.02	0.01
		recall	**0.76**	0.75	0.76	0.77
		F1-score	0.04	0.04	0.04	0.02
	Benign/Malign	kappa	0.03	0.03	0.03	0.02
		gmean	**0.78**	0.77	0.78	0.78
EV	Benign	precision	1.00	1.00	1.00	1.00
		recall	0.87	0.87	**0.86**	0.88
		F1-score	0.93	0.93	0.92	0.94
	Malign	precision	0.02	0.02	0.02	0.01
		recall	0.50	0.52	**0.53**	0.44
		F1-score	0.04	0.04	0.04	0.02
	Benign/Malign	kappa	0.03	0.02	0.03	0.02
		gmean	0.65	0.67	**0.68**	0.70
LOF	Benign	precision	1.00	1.00	1.00	1.00
		recall	0.80	0.79	0.79	**0.78**
		F1-score	0.89	0.88	0.88	0.88
	Malign	precision	0.01	0.01	0.01	0.01
		recall	0.50	0.45	0.45	**0.55**
		F1-score	0.02	0.02	0.02	0.02
	Benign/Malign	kappa	0.03	0.02	0.03	0.02
		gmean	0.63	0.60	0.60	**0.66**
RF	Benign	precision	1.00	1.00	1.00	1.00
		recall	**0.98**	0.97	0.98	1.00
		F1-score	0.99	0.98	0.99	1.00
	Malign	precision	0.06	0.03	0.04	0.11
		recall	**0.23**	0.19	0.16	0.12
		F1-score	0.09	0.05	0.07	0.12
	Benign/Malign	kappa	0.08	0.04	0.06	0.11
		gmean	**0.47**	0.42	0.39	0.35

TABLE 4. Metrics obtained by running the retraining every 60 days.

		Training/Retraining Set Length (days)				
Algorithm	**Class**	**Metric**	**15**	**30**	**60**	**150**
ISOF	Benign	precision	1.00	1.00	1.00	1.00
		recall	0.80	0.80	**0.80**	0.79
		F1-score	0.88	0.89	0.89	0.89
	Malign	precision	0.02	0.02	0.02	0.02
		recall	0.76	0.76	**0.78**	0.77
		F1-score	0.04	0.04	0.04	0.04
	Benign/Malign	kappa	0.03	0.03	0.03	0.02
		gmean	0.78	0.78	**0.79**	0.78
EV	Benign	precision	1.00	1.00	1.00	1.00
		recall	0.80	0.80	0.80	**0.76**
		F1-score	0.89	0.89	0.89	0.86
	Malign	precision	0.02	0.02	0.02	0.01
		recall	0.65	0.64	0.65	**0.71**
		F1-score	0.03	0.03	0.03	0.03
	Benign/Malign	kappa	0.02	0.02	0.02	0.02
		gmean	0.72	0.72	0.72	**0.73**
LOF	Benign	precision	1.00	1.00	1.00	1.00
		recall	0.79	0.79	0.79	**0.77**
		F1-score	0.88	0.88	0.88	0.87
	Malign	precision	0.01	0.01	0.01	0.01
		recall	0.50	0.46	0.47	**0.57**
		F1-score	0.02	0.02	0.02	0.02
	Benign/Malign	kappa	0.01	0.01	0.01	0.01
		gmean	0.63	0.60	0.61	**0.66**
RF	Benign	precision	1.00	1.00	1.00	1.00
		recall	0.99	**0.99**	0.99	1.00
		F1-score	0.99	0.99	0.99	0.99
	Malign	precision	0.05	0.07	0.06	0.06
		recall	0.14	**0.18**	0.14	0.12
		F1-score	0.08	0.11	0.09	0.08
	Benign/Malign	kappa	0.07	0.05	0.09	0.10
		gmean	0.36	**0.42**	0.37	0.36

The best values obtained by RF for the recall metric were 0.98 for class 0 and 0.23 for class 1. To achieve these results, RF has used fifteen days for the training/retraining set length and fifteen days for the retraining interval. After verifying the RF results, this work concluded that the RF algorithm does not fit well when applied to the insider threat detection environment. The highly unbalanced dataset may be the crucial cause of the poor RF performance. Due to the shortage of positive class samples, RF may not have been able to identify this class.

Another significant analysis is to understand how the performance of algorithms varies over time. Figures 3, 4, and 5 show the precision, recall, and F1-score metrics behavior during the model execution. The y-axis represents the metrics values, and the x-axis represents the number of analyzed samples (sessions). The yellow line corresponds to ISOF results, the blue line corresponds to EV results, and the red line corresponds to the LOF results.

The figures show that whenever the recall curve stabilizes the precision curve tends to decline. In the same situation, the F1-score curve also decreases slightly, influenced by precision results. The likely cause for this is an increase in false negatives. The probable reason for this is that true and false results grow at proportionally different rates. This fact stems from two main factors, the high imbalance rate imposed by the problem and the fact that malign and benign activities are similar in the case of insider threats. After analyzing about 50000 samples, the lines of the graphics stop drastically oscillating and become smoother.

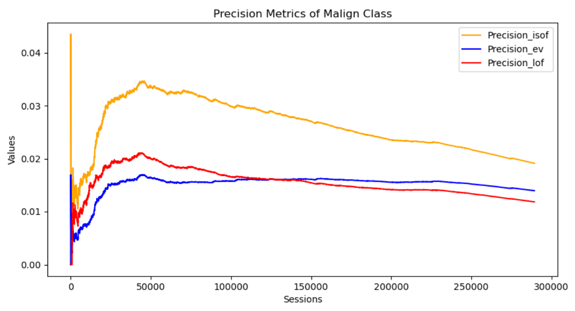

FIGURE 3. Malign class precision evolution.

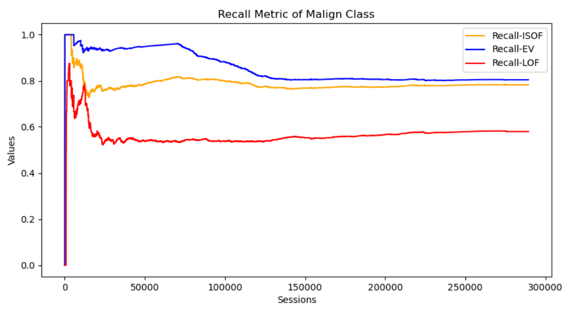

FIGURE 4. Malign class recall evolution.

TABLE 5. Metrics obtained by running the retraining every 120 days.

			Training/Retraining Set Length (days)			
Algorithm	**Class**	**Metric**	**15**	**30**	**60**	**150**
ISOF	Benign	precision	1.00	1.00	1.00	1.00
		recall	0.80	0.80	**0.80**	0.78
		F1-score	0.89	0.89	0.89	0.88
	Malign	precision	0.02	0.02	0.02	0.02
		recall	0.77	0.76	**0.78**	0.77
		F1-score	0.04	0.04	0.04	0.03
	Benign/Malign	kappa	0.03	0.03	0.03	0.02
		gmean	0.78	0.78	**0.79**	0.78
EV	Benign	precision	1.00	1.00	1.00	1.00
		recall	0.69	**0.69**	**0.69**	0.68
		F1-score	0.82	0.82	0.82	0.81
	Malign	precision	0.01	0.01	0.01	0.01
		recall	0.81	**0.82**	**0.82**	0.82
		F1-score	0.03	0.03	0.03	0.02
	Benign/Malign	kappa	0.02	0.01	0.01	0.01
		gmean	0.75	**0.75**	**0.75**	0.74
LOF	Benign	precision	1.00	1.00	1.00	1.00
		recall	0.79	0.79	0.79	**0.76**
		F1-score	0.88	0.88	0.88	0.86
	Malign	precision	0.01	0.01	0.01	0.01
		recall	0.51	0.47	0.49	**0.58**
		F1-score	0.02	0.02	0.02	0.02
	Benign/Malign	kappa	0.01	0.01	0.01	0.01
		gmean	0.63	0.61	0.62	**0.66**
RF	Benign	precision	1.00	1.00	1.00	1.00
		recall	0.99	**0.98**	0.99	1.00
		F1-score	0.99	0.99	0.99	1.00
	Malign	precision	0.08	0.04	0.09	0.15
		recall	0.14	**0.17**	0.14	0.11
		F1-score	0.11	0.07	0.12	0.15
	Benign/Malign	kappa	0.08	0.05	0.10	0.11
		gmean	0.37	**0.41**	0.37	0.32

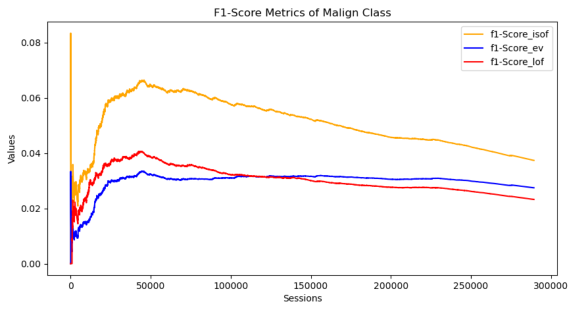

FIGURE 5. Malign class F1-score evolution.

In addition, this study carried out a malicious scenario analysis, as shown in Table 7. The first column represents the scenario analyzed, the second shows the metrics, and the other the algorithms, except the last which shows the total. The Total column represents the sum of all positive samples in each scenario, and the last line is the sum of negative ones. The last line is not divided by scenario because it represents only the negative class. In this table, the detection performance by scenario can be observed. The results come from the best model performance for each algorithm. The first scenario has 189 malicious activities executed by 30 users, the second one has 1110 malicious activities executed by 30 users, and the third one has 121 malicious activities executed by 10. The ISOF algorithm was the least noisy, having 57,770 false positive alarms.

TABLE 6. Confusion matrix by scenario.

Scenario	Metrics	ISOF	EV	LOF	Total
1	TP	89	99	46	
	FN	100	90	143	**189**
2	TP	927	939	737	
	FN	183	171	373	**1110**
3	TP	90	104	27	
	FN	31	17	94	**121**
-	TN	230,090	207,145	222,341	
	FP	**57,770**	**80,715**	**65,519**	**287,860**

TABLE 7. True positive rate by scenario.

Scenario	Metrics	ISOF	EV	LOF
1	TPR	0.47	0.52	0.24
2	TPR	0.83	0.84	0.66
3	TPR	0.74	0.85	0.22
ALL	TPR	0.78	0.80	0.57

The first and second scenarios have 30 malicious users, and the third has 10. Using the ISOF algorithm, the model detected all malicious users issuing at least one alert for each, and in most cases, an alarm was sounded for the first malicious activity. For the second and third scenarios, only three first malicious activities were not alerted, and for the first scenario, eight were also not. With the EV algorithm, the model did not detect only two malicious users from the first scenario. Concerning the first malicious activity, the EV algorithm did not identify seven from the first scenario, five from the second, and one from the third. Finally, when using the LOF algorithm, the model did not detect only seven malicious users from the first scenario and did not alert twenty first malicious activities from the first scenario, nine from the second, and seven from the third.

Observing the TPR, it can be stated that the first scenario was the most challenging to identify. The EV algorithm performed better in this scenario, but at the cost of a high generation of PF results, and even so, it failed to identify two of the thirty malicious users of the scenario. Despite a slightly worse performance for the first scenario, the ISOF algorithm identified all malicious users involved.

When analyzing the results, it can be noted that the one-class algorithms better adapt to the insider threat scenarios presented in the ITD. The insufficient labeled samples penalized the RF algorithm, which suffered from the highly unbalanced dataset.

VI. CONCLUSION

Insider Threat Detection Systems based on batch-supervised learning may not be suitable for real detect scenarios. It is because it is difficult to obtain real-time labeled datasets, and insider threat behaviors continually change over time [4]. There is no signature to identify insider threats and the more sophisticated the attack the more it looks like a benign activity. Considering the semi-supervised machine learning approach is focused on data flow, a model capable of dealing with the detection of insider threats can be built more practically. This approach eliminates labeled sample dependence which became a system based on this approach more feasible to implement in the real world.

The results demonstrated the effectiveness of the framework and its feasibility of being implemented in a real-world scenario. The retraining procedures helped to improve the model's adaptability to concept drifts. The retraining also made a relevant contribution to the feasibility of the proposed framework as it made the marked reduction of false positive results possible. The one-class algorithms seem to be a better option to compound a solution for insider threat detection. The possibility of training the model with only one of the classes in a semi-supervised way allowed us not to rely on using labeled samples. In the present case, ISOF was the algorithm that best adapted to the problem. This algorithm achieved a relative balance between the recall metrics of the benign (0.80) and malign (0.78) classes adapting better to the accentuated imbalance present in the dataset. Although the implemented model did not detect all malicious samples, it did detect malicious activities from all scenarios implemented in the dataset using the ISOF algorithm. The model also generated alarms for all malicious users in the data stream. In most cases, the model identified the malicious users in their first malicious activity.

In the case of insider threat detection, the problem is how to obey contamination-free samples. One option may be only using data considered benign by network security solutions. However, even in these cases, we would not be protected against exploiting zero-day vulnerabilities. This fact motivated the deliberate inclusion of contamination in the experiments carried out in this article. This study simulates a data stream with the ITD to supply a limitation of generating a data stream that makes sense for insider threat detection. In other domains may be easy to create aleatory test datasets, but this situation does not fit in this study context. Having said that, there is still room for improvement, such as using other types of algorithms and discovering new attributes that can increase the representativeness of the dataset, thereby reducing the occurrence of false positive results. Although, if we compare the estimated values for losses related to attacks performed by insider threats and the cost of checking for false results, we will see that checking for false results turns out to be worth it [11]. The next step is to test the model with real-life and actual data.

REFERENCES

[1] Cybersecurity e Infrastructure Security Agency. (Nov. 2020). *Insider Threat Mitigation*. [Online]. Available: https://www.cisa.gov/insider-threat-mitigation

[2] C. Soh, S. Yu, A. Narayanan, S. Duraisamy, and L. Chen, "Employee profiling via aspect-based sentiment and network for insider threats detection," *Exp. Syst. Appl.*, vol. 135, pp. 351–361, Nov. 2019, doi: 10.1016/j.eswa.2019.05.043.

[3] I. A. Gheyas and A. E. Abdallah, "Detection and prediction of insider threats to cyber security: A systematic literature review and meta-analysis," *Big Data Analytics*, vol. 1, no. 1, p. 6, 2016.

[4] Proofpoint Institute. (2022). *Cost of Insider Treat, Global Report*. [Online]. Available: https://www.proofpoint.com/us/resources/threat-reports/cost-of-insider-threats

[5] J. Kim, M. Park, H. Kim, S. Cho, and P. Kang, "Insider threat detection based on user behavior modeling and anomaly detection algorithms," Ph.D. dissertation, School Ind. Manag. Eng., Korea Univ., Seoul, South Korea, 2019.

[6] I. Homoliak, F. Toffalini, J. Guarnizo, and Y. Elovici, "Insight into insiders and IT: A survey of insider threat taxonomies, analysis, modeling, and countermeasures," *ACM Comput. Surveys*, vol. 99, no. 99, pp. 1–40, 2017.

[7] D. C. Le and A. N. Zincir-Heywood, "Machine learning based insider threat modelling and detection," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, Apr. 2019, pp. 1–6.

[8] N. Saxena, E. Hayes, E. Bertino, P. Ojo, K. K. R. Choo, and P. Burnap, "Impact and key challenges of insider threats on organizations and critical businesses," *Electronics*, vol. 9, no. 9, pp. 1–29, 2020.

[9] S. J. Berdal, "A holistic approach to insider threat detection," M.S. thesis, Dept. Inform., Fac. Math. Natural Sci., Univ. Oslo, Oslo, Norway, 2018.

[10] B. Böse, B. Avasarala, S. Tirthapura, Y. Chung, and D. Steiner, "Detecting insider threats using RADISH: A system for real-time anomaly detection in heterogeneous data streams," *IEEE Syst. J.*, vol. 11, no. 2, pp. 471–482, Jun. 2017.

[11] P. Chattopadhyay, L. Wang, and Y. Tan, "Scenario-based insider threat detection from cyber activities," *IEEE Trans. Computat. Social Syst.*, vol. 5, no. 3, pp. 660–675, Sep. 2018.

[12] A. J. Hall, N. Pitropakis, W. J. Buchanan, and N. Moradpoor, "Predicting malicious insider threat scenarios using organizational data and a heterogeneous stack-classifier," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 5034–5039.

[13] T. E. Senator, H. G. Goldberg, A. Memory, W. T. Young, B. Rees, R. Pierce, D. Huang, M. Reardon, D. A. Bader, E. Chow, I. Essa, and J. Jones, "Detecting insider threats in a real corporate database of computer usage activity," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2013, pp. 1393–1401.

[14] P. Parveen, N. McDaniel, Z. Weger, J. Evans, B. Thuraisingham, K. Hamlen, and L. Khan, "Evolving insider threat detection stream mining perspective," *Int. J. Artif. Intell. Tools*, vol. 22, no. 5, pp. 1–24, 2013.

[15] J. Noble and N. Adams, "Real-time dynamic network anomaly detection," *IEEE Intell. Syst.*, vol. 33, no. 2, pp. 5–18, Mar. 2018.

[16] G. Gavai, K. Sricharan, D. Gunning, J. Hanley, M. Singhal, and R. Rolleston, "Supervised and unsupervised methods to detect insider threat from enterprise social and online activity data," *J. Wireless Mobile Netw., Ubiquitous Comput., Dependable Appl.*, vol. 6, no. 4, pp. 47–63, 2015.

[17] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," in *Proc. AAAI Workshop Tech. Rep.*, 2017, pp. 224–234.

[18] J. Gama, *Knowledge Discovery from Data Streams*, 1st ed., V. Kumar, Ed. Boca Raton, FL, USA: Chapman & Hall/CRC, 2010.

[19] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, "A survey on ensemble learning for data stream classification," *ACM Comput. Surveys*, vol. 50, no. 2, pp. 1–36, Mar. 2018.

[20] G. H. Ribeiro, "Detecção de botnets utilizando classificação de fluxos contínuos de dados," M.S thesis, Faculdade Computação, Universidade Federal de Uberlândia, Brazil, 2020.

[21] J. M. C. de Sá, A. L. Rossi, G. E. Batista, and L. P. Garcia, "Algorithm recommendation for data streams," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 6073–6080.

[22] G. Silowash, D. Cappelli, A. Moore, R. Trzeciak, T. Shimeall, and L. Flynn, "Common sense guide to mitigating insider threats, 4th edition," Carnegie Mellon Univ., Softw. Eng. Inst., Pittsburgh, PA, USA, Tech. Rep. CMU/SEI-2018-TR-010, Dec. 2012.

[23] M. N. Al-Mhiqani, R. Ahmad, Z. Z. Abidin, and W. Yassin, "A review of insider threat detection: Classification, machine learning techniques, datasets, open challenges, and recommendations," *Appl. Sci.*, vol. 10, no. 15, p. 5208, 2020.

[24] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *Proc. IEEE Secur. Privacy Workshops*, May 2013, pp. 98–104.

[25] L. Liu, O. De Vel, Q. Han, J. Zhang, and Y. Xiang, "Detecting and preventing cyber insider threats: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1397–1417, 2nd Quart., 2018.

[26] B. Krawczyk and M. Wozniak, "One-class classifiers with incremental learning and forgetting for data streams with concept drift," *Soft Comput.*, vol. 19, no. 12, pp. 3387–3400, Dec. 2015.

[27] B. Kurlej and M. Wozniak, "Active learning approach to concept drift problem," *Log. J. IGPL*, vol. 20, no. 3, pp. 550–559, Jun. 2012, doi: 10.1093/jigpal/jzr011.

[28] D. W. Kim, S. S. Hong, and M. M. Han, "A study on classification of insider threat using Markov chain model," *KSII Trans. Internet Inf. Syst.*, vol. 12, no. 4, pp. 1887–1898, 2018.

[29] H. Zhang, W. Liu, S. Wang, J. Shan, and Q. Liu, "Resample-based ensemble framework for drifting imbalanced data streams," *IEEE Access*, vol. 7, pp. 65103–65115, 2019.

[30] K. Kendall, "A database of computer attacks for the evaluation of intrusion detection systems," M.S. thesis, Massachusetts Insitute Thechnology, Cambridge, MA, USA, Jun. 1999.

[31] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jun. 2009, pp. 139–147.

[32] S. Yuan and X. Wu, "Deep learning for insider threat detection: Review, challenges and opportunities," *Comput. Secur.*, vol. 104, May 2021, Art. no. 102221. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404821000456

[33] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: A big comparison for NAS," 2019, *arXiv:1912.06059*.

[34] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining*, Dec. 2008, pp. 413–422.

[35] S. Howard, "The elliptical envelope," 2007, *arXiv:math/0703048*. [Online]. Available: https://arxiv.org/abs/math/0703048

[36] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*. New York, NY, USA: Association for Computing Machinery, 2000, pp. 93–104, doi: 10.1145/342009.335388.

[37] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[38] S. S. Khan and M. G. Madden, "One-class classification: Taxonomy of study and review of techniques," *Knowl. Eng. Rev.*, vol. 29, no. 3, pp. 345–374, Jun. 2014.

[39] M. Vakili, M. Ghamsari, and M. Rezaei, "Performance analysis and comparison of machine and deep learning algorithms for IoT data classification," 2020, *arXiv:2001.09636*.

[40] A. Rossi, "Meta-aprendizado aplicado a fluxos contínuos de dados," Ph.D. dissertation, Instituto de Ciências Matemáticas e de Computação, Univ. de São Paulo, São Paulo, Brazil, 2014.

[41] R. H. Moulton, H. L. Viktor, N. Japkowicz, and J. Gama, "Contextual one-class classification in data streams," 2019, *arXiv:1907.04233*.

RAFAEL BRUNO PECCATIELLO received the Graduate degree in information systems, in 2010, and he is currently applying for a master's degree with the Department of Computer Science, University of Brasilia (UnB). He has experience in subjects related to cyber defensive and offensive operations. His research interests include security information, digital forensics, and cyber security.

JOÃO JOSÉ COSTA GONDIM received the M.Sc. degree in computing science from the Imperial College, University of London, in 1987, and the Ph.D. degree in electrical engineering from the University of Brasilia (UnB), in 2017. He is currently an Assistant Professor with the Department of Computer Science (CIC), UnB, where he is also a tenured member of the Faculty of Engineering. His research interests include networks, information, and cyber security.

LUÍS PAULO FAINA GARCIA received the Graduate degree in computer engineering, in 2010, and the Ph.D. degree in computer science from the University of São Paulo, in 2016. In 2017, his thesis was ranked among the best by the Brazilian Computer Society and received the CAPES Award for the best thesis in computer science in the country. He is currently an Assistant Professor with the Department of Computer Science, University of Brasília. He has experience in subjects related to noise detection, meta-learning, and data streams.