Anomaly Detection for Insider Threats Using Unsupervised Ensembles

Duc C. Le, *Student Member, IEEE,* and Nur Zincir-Heywood, *Member, IEEE*

***Abstract*—Insider threat represents a major cybersecurity challenge to companies, organizations, and government agencies. Insider threat detection involves many challenges, including unbalanced data, limited ground truth, and possible user behaviour changes. This research presents an unsupervised learning based anomaly detection approach for insider threat detection. We employ four unsupervised learning methods with different working principles, and explore various representations of data with temporal information. Furthermore, different computational intelligence schemes are explored to combine these models to create anomaly detection ensembles for improving the detection performance. Evaluation results show that the approach allows learning from unlabelled data under challenging conditions for insider threat detection. Insider threats are detected with high detection and low false positive rates. For example, 60% of malicious insiders are detected under 0.1% investigation budget, and all malicious insiders are detected at at less than 5% investigation budget. Furthermore, we explore the ability of the proposed approach to generalize for detecting new anomalous behaviours in different datasets, i.e. robustness. Finally, results demonstrate that a voting-based ensemble of anomaly detection can be used to improve detection performance as well as the robustness. Comparisons with the state-of-the-art confirm the effectiveness of the proposed approach.**

***Index Terms*—insider threat detection, anomaly detection, ensemble learning, unsupervised learning, temporal data, dependable and robust learning.**

I. Introduction

One of the most prevalent and destructing security threats to computer networks, data, and intellectual property of companies and organizations is *insider threat*. In insider threat, malicious actions are carried out by authorized personnel/employees of organizations, which may be familiar with its structure, valued properties, and security layers. Therefore, detecting and mitigating insider threats represent a major challenge [1]. According to recent reports, insider threats account for a fourth of all cyberattacks experienced by U.S. organizations [2]. Up to 53% of organizations and 42% of U.S. federal agencies faced insider threat attacks every year [3]. Furthermore, insider threat attacks have become more frequent recently [4].

According to the CERT Insider Threat Center, insider threat is defined as threats that originated from malicious or unintentional insiders, whose authorized access to networks, systems, and data of an organization is exploited to negatively affect the confidentiality, integrity, availability, or physical state of the organization's information, information systems, or workforce [5]. Typical threats caused by malicious insiders are trade secrets / intellectual property theft, disclosure of classified information, theft of personal information, and IT system sabotage [5]. A major challenge in detecting insider threats come from the fact that malicious insiders are authorized to access the organization's systems and networks. In addition, data describing insider threat activities is typically rare and poorly documented [6]. Furthermore, storing, processing, and analyzing multiple sources of organizational data – from network traffic, authentication logs, to web and email history – for identifying malicious insiders and other potential threats present another challenge in implementing detection solutions.

The authors are with Faculty of Computer Science, Dalhousie University, Canada (e-mail: lcd@dal.ca, zincir@cs.dal.ca).

This work presents our proposed *anomaly detection* approach, where the focus is on employing unsupervised machine learning (ML) methods and different representations of data with temporal information for identifying signs of anomalous behaviours that may indicate insider threats. This is the initial and important detection step in cybersecurity workflow, where early signs of user behaviour changes (anomaly) is flagged for further investigation, potentially to detect both known and unknown (zero day) attacks/vulnerabilities [7]. In doing so, the contributions of this paper are summarized as follows: (i) Different representations of data, namely concatenation, percentile, mean and median difference, are introduced for ML-based anomaly detection algorithms, where temporal information is encoded to highlight user behaviour changes; (ii) Capabilities and characteristics of popular unsupervised ML methods for anomaly detection – Autoencoder, Isolation Forest, Lightweight On-line detection of anomalies, and Local Outlier Factor – are examined under different working conditions, namely training data poisoning, number of users in training data and the duration of training data; (iii) Anomaly detection ensembles are created to explore performance enhancing capabilities and characteristics of different combining schemes; (iv) Comprehensive anomaly detection results are presented, per instance and per user; and (v) Evaluating on publicly available datasets, the proposed approach demonstrates the ability to generalize and detect malicious insiders under very low investigation budgets.

The rest of the paper is organized as follows. Section II summarizes the related literature. Section III introduces the proposed anomaly detection approach. Section IV details the experiments and presents the evaluation results, while Section V further discusses the results and makes comparisons. Finally, conclusion and future research directions are presented in Section VI.

II. Related Work

Research in insider threat detection and mitigation has attracted more and more attention from organizations and cybersecurity firms recently. Guides to detect and common practices to combat insider threats in organizations were released by the CERT Insider Threat Center and U.S. National Cybersecurity and Communications Integration Center [5], [8]. In [5], Collins *et al.* described case studies of misbehaviour and 20 practices for organizations to prevent insider threats. Recent surveys by Homoliak *et al.* [1] and Liu *et al.* [9] address the definition, taxonomy and categorization of insider threats, and provide an overview of the countermeasures.

Following the successes in ML applications for intrusion detection and anomaly detection tasks [7], [10], ML techniques have been applied in insider threat detection, based on their ability to learn from large amount of data to detect anomalous / malicious behaviours of insiders. Most of the proposed approaches are based on anomaly detection. Some notable directions in anomaly detection for insider threat are graph-based approach [11], [12], Hidden Markov Model [13], [14], oneclass-SVM [15], and deep learning-based autoencoders and recurrent networks [16]–[18]. In [19], an approach employing anomaly detectors on combined information from multiple domains of user activities is proposed by Eldardiry et al. The approach aims to detect blend-in anomalies and unusual change anomalies. In [11] and [12], different varieties of graph-based anomaly detection are employed for insider threat detection, where user and system data are represented in graph formats based on their interactions and connections. In several works, different anomaly detection models are built for each employee/user in an organization in order to detect changes in the user behaviours [13], [16], [20]. Rashid et al. used Hidden Markov Models to model users' weekly activity sequences and detect possible insider threats based on noticeable changes (low probabilities) in weekly user activity sequences [13]. In [21], a framework for modelling the insider threat problem based on behavioural and psychological observations is proposed that allows analyst to reason on user data and construct hypothesis trees to describe potential insider threats. Furthermore, to improve detection performance in intrusion detection applications, mixture of models and ensemble approaches have been used to improve detection results [22], [23].

Other ML concepts have been introduced to deal with different aspects of the insider threat detection problem, such as non-stationary environment and imbalanced data. Stream online learning are employed in [12] and [16] for learning under non-stationary conditions of user behaviours, and to provide continuous learning and predicting on data arrival. With the recent trend in deep learning applications, many deep learning based approaches to insider threat detection have been proposed as well [16]–[18], [24]. Evolutionary computation [25], [26] and supervised learning techniques [27]–[29] are some other examples.

Another closely related problem to insider threat detection is lateral movement detection, in which many malicious actions are also performed using insider accounts. Lateral movement is used as a technique in advanced persistent threat by threat actors to gain access to their intended targets, moving through compromised accounts and systems of the victim organization. Recently proposed machine learning approaches for lateral movement detection are graph analysis [11], [30], recurrent neural networks [24], and other supervised learning techniques [31].

Some works in the literature report the importance of temporal information in dealing with insider threat, which is highly related to the human factors [13], [16], [17], [20], [22]. Notable attempts to leverage temporal information include a moving average approach [16], [20], graph embedding [11], or employing ML models with temporal learning capabilities [13], [16]. This work instead explore the representation of temporal information in data for anomaly detection training. With dynamics and user behaviour changes in mind, we keep the focus on detecting the changes in each user's most recent activities instead of the whole / averaging over the time range of data. Furthermore, this work constructs a single anomaly detection model for a given training dataset under reasonable training time (III-C). This shows advantages over other approaches that build one / many models per user [13], [16], [20] and employ not as scalable learning algorithms [15]–[17]. In our previous work [32], a preliminary version of the proposed anomaly detection was presented. In this work, we aim to present a comprehensive anomaly detection system for insider threat with expanded scopes and directions: (i) different anomaly detection methods are employed to provide insights under other unsupervised learning paradigms, such as online learning; (ii) different ensembles for combining anomaly detection models are investigated; (iii) different temporal data representation methods and new datasets for system evaluations are explored; and (iv) extended learning conditions and further analysis / comparison of system performance are presented.

III. Anomaly detection system for insider threat

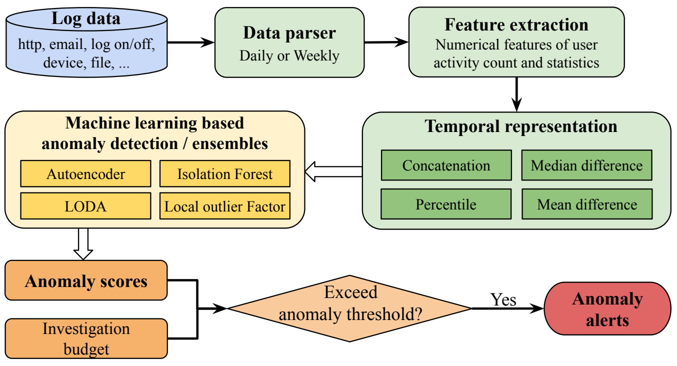


Fig. 1. Components of the proposed anomaly detection system

Fig. 1 shows an overview of the proposed insider threat detection system. From raw collected log data of user activities, the data is pre-processed to extract numerical features by day or week, with different temporal representations. The pre-processing and feature extraction processes are detailed in III-A. The extracted data are then used to train anomaly detection models using unsupervised learning. The employed ML methods are described in III-C. Post training, anomaly scores are assigned by the detection model. Based on a selected investigation budget, a decision threshold can be calculated so that data samples with high anomaly scores (i.e. exceeding a threshold) are flagged for further investigation of possible malicious actions.

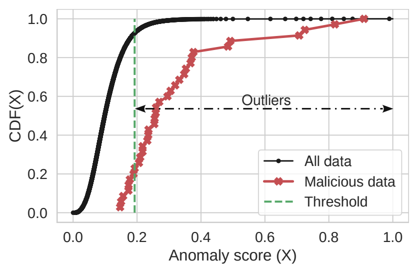


Fig. 2. Demonstration of anomaly detection and threshold.

Using anomaly detection based on unsupervised learning, the assumption is that malicious behaviours are often rare and deviated from normal user behaviours, which constitute the vast majority of the collected data [7], [33]. Thus, although no label information is used, a trained anomaly detection model may capture the normal data and reveal anomalous behaviours as outliers. Outliers identified by the anomaly detection model are defined by a threshold of anomaly scores, as demonstrated in Fig. 2. In this work, different thresholds are examined through changing the *investigation budget* (IB), which is the amount (%) of data that the security analyst can examine for confirmation of malicious behaviours [7], [34]. This represents the available human resources for analyzing the highest ranked data instances, post-training of the detection system, and performing the necessary actions in response.

A. Data Pre-processing with Temporal Information

Assuming common monitoring data in organizations, such as web access, email and file access logs, the data pre-processing step is performed based on aggregated user activity data, e.g. *daily* or *weekly*. These time periods for data aggregation are selected to summarize a complete view of users' activities over a day or a week into a data instance [35]. Daily or weekly data may be selected in deployment depending on each organization's human resources for inspecting anomaly alerts and requirements for timely detection. More fine-grained data, e.g. session of user activities, could be extracted as well [35]. However, that may not be beneficial in terms of utilizing human resources, as extracting fine-grained data increases the data count, and thus raises the workload to inspect anomaly alerts in the unsupervised anomaly detection setting, where false alerts are unavoidable [7].

Numerical features are then extracted for each day or week of a user's activities. Two main types of features from the data are extracted: (i) Frequency features, i.e. numbers of different user actions over a day / week, e.g. *number of external emails received, number of file access after work hours*, and (ii) Statistical features, i.e. the mean and the standard deviation of changing statistics, e.g. *email size, file size*. When possible, user information, such as user's role and user-user relationship, is employed in the extraction process to provide context for ML algorithms. Further details of the process to extract numerical features with different information depicting PC, action's time, or email/HTTP categories from log files can be found in [35].

B. Temporal information in data representation

Exploiting the fact that insiders are essentially regular employees before they start performing malicious actions [5], we propose data representation approaches using temporal information. The goal is to highlight the trends / changes in the user behaviour over time. This may potentially reveal the transitions in behaviours of malicious insiders. The approach performs concatenating or comparing a data point to a time window of the most recent data of the same user.

Using a window of time, the approach compares a user's activities to only his/her most recent and relevant behaviours. As concept shift and drift are likely in user behaviours [36], this may be more effective than normalizing all data instances of each user from the beginning. Furthermore, by processing each data instance via time window, our approach is ready to apply anytime a new data instance appears, which is critical in online stream learning. Additionally, we note that in contrast to extracting time series data from a time window, where all data points in the window contributes similarly to the output, in our work, the focus is in using the time window to define a baseline comparison for each new data instance (see V-D).

1) Concatenation: Inspired by the use of shift register and taps for representing time in data for intrusion detection [37], we introduce data examples to anomaly detection algorithms as concatenation of γ consecutive data instances of the same user (abbreviated as C$_\gamma$). The idea is to encourage the learning algorithms to construct comparisons/arithmetic operations between each user data instance and its previous records. In this data representation form, a data instance x_t at time t is adjoined with $\gamma - 1$ most recent instances to form a data point for anomaly detection:

$$x_t^{\text{concatenation}} = \text{concat}(x_t, x_{t-1}, x_{t-2}, ..., x_{t-\gamma+1}) \qquad (1)$$

Essentially, this creates a data instance with c times the number of features originally extracted.

2) Comparing to a time window – Percentile and Mean/Median difference representations: In order to explicitly include temporal information and reflect changes in user activities, we propose to represent data for anomaly detection via a function comparing each data instance x_t with a time window w leading to t. The procedure is summarized in Algorithm 1.

Each arriving data instance is compared with previous data instances of the same user in the most recent time window w to create percentile or mean/median difference representation. In this work, we set the window size w to 7 days, 30 days, or 60 days (IV-A). This setting allows contrasting each day (week) of user's activity against the same user's activities in the full week (month) leading to it, where both weekdays and weekends are taken into account to provide sufficient information for comparison.

Algorithm 1: Calculating Percentile and Mean/Median difference representation of data

```
Input  : $x_t$ of user $u$, window size $w$
Output: $x_t^{\text{output}}$
construct a $n \times |F|$ matrix $X$ of $x_{t-1}, x_{t-2}, ..., x_{t-n}$ of
  the same user $u$, based on $w$ ;   // $F$: features
$x_t^{\text{output}} = [\ ]$;
for feature $f$ in $F$ do
    if Percentile then
        $f' = findPercentile(x_t[f], X[:, f])$;
    else if Mean difference then
        $f' = x_t[f] - \overline{E}(X[:, f])$;
    else if Median difference then
        $f' = x_t[f] - median(X[:, f])$;
    $x_t^{\text{output}}$.append($f'$);
```

C. Unsupervised Machine Learning for Anomaly Detection

This work employs four popular ML methods for anomaly detection with different underlying concepts: Autoencoder (AE), Isolation Forest (IF), Lightweight On-Line Anomaly Detection (LODA), and Local Outlier Factor (LOF).

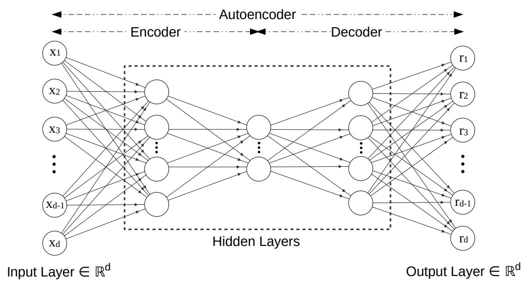


Fig. 3. An example of an autoencoder.

1) Autoencoder (AE): AE is a form of multi-layer neural network that compresses and reconstructs the data. Fig. 3 depicts an example of an AE with three hidden layers. The input and output layers both have d neurons (d: the number of dimensions). Each data dimension j in the input x is reconstructed into a corresponding dimension of r at the output layer by AE. By enforcing a "bottleneck" architecture through hidden layers (middle hidden layer size:h, $h \ll d$), AE compresses (encodes) the input data into h dimensions and reconstructs it at the output layer. AE is trained through minimizing the aggregated reconstruction error as the cost function:

$$E = \sum_{i}^{N} \sqrt{\sum_{j=1}^{d} (x_{ij} - r_{ij})^2}, \tag{2}$$

Post training, the lossy compression produced by AE essentially captures the lower-dimensionality representation of the majority of training data at the middle hidden layer. Assuming that normal user data constitute the majority of the training data, it is expected that AE shows a higher reconstruction error for anomalies [38], which may represent malicious insider behaviours. Thus, for each data instance x, the AE anomaly score is defined as the Euclidean distance between x and r: $e_i = \sqrt{\sum_{j=1}^{d} (x_{ij} - r_{ij})^2}$. To construct AE models in this work, the hidden layers and the output layer take the form of rectified linear [39] and sigmoid activation functions, respectively.

2) Isolation Forest (IF): Based on the principle that anomaly examples are rare and significantly different in attribute-values from normal data points, IF [40] is designed as an ensemble of "isolation trees", whereas the anomalies – being easier to isolate – are assumed to be closer to the roots of the trees than normal instances. This is different from other anomaly detection methods, which build models of (mostly) normal data, and identify anomalies as any instances that do not conform to the model.

Each tree in IF works on a subset of training data and feature set. Binary splits are generated in each node of a tree by a randomly selected feature and split value. The process is recursively repeated until each instance is isolated in a leave. Having trained all isolation trees, the anomaly score of a data instance – $h(x) = \overline{E}(h_i(x))$ – is calculated as the average path length from root nodes to the corresponding leaves of the instance in the trees ($h_i(x)$).

Based on different principles from other outlier detection methods (such as AE), IF has been shown to possess some desired capabilities: To be able to deal with high dimensional data with irrelevant attributes, and to be trained with or without anomalies included the training set [40]. These characteristics are evaluated in IV-C.

3) LODA – Lightweight on-line detector of anomalies: LODA [41] is an ensemble method combining weak histogram based anomaly detectors into a strong detector. Similar to IF, each histogram anomaly detector in LODA works on a subset of input features in order to to promote diversity. This is achieved through the use of sparse random projections $\{w_i \in \mathbb{R}^d\}_{i=1}^k$, where k one-dimensional vectors, each has $\sqrt{d}$ non-zero components, are created to approximate the probability density of input data. Individual histograms are then calculated for each of k vectors. Each histogram shows an approximation of the original data distribution, which may reveal some aspects of outliers that come from different distribution than normal data. Furthermore, in online LODA training, each histogram is updated by a training sample by projecting the sample onto a vector and then the corresponding histogram bin is updated.

To produce anomaly score for a data sample, LODA uses the average of the logarithm of probabilities estimated on individual projection vectors:

$$f(x) = -\frac{1}{k} \sum_{i=1}^{k} \log \hat{p}_i(x^T w_i), \tag{3}$$

LODA was shown to achieve comparable detection performance to more complicated algorithms, while significantly reduce time and storage complexity [41]. Furthermore, it is also able to operate and update itself in real-time online environment and on data with missing variables. In cybersecurity, LODA's ability in identifying features of an outlier sample that deviates from the majority provides an useful tool to explain the causes of anomalous events detected.

4) Local Outlier Factor (LOF): LOF [42] is a popular anomaly detection algorithm, which proposes the concept of local density to identify anomalous data points. The local density measures how isolated a data sample is with respect to the surrounding neighbourhood. By comparing the local density of a data sample to that of its neighbours, LOF can identify data points that have a substantially lower density than their neighbours, which are considered to be outliers.

Considering k nearest neighbours to each data point, a k-distance(x) is defined as the distance of point x to its k^{th} neighbour, and $N_k(x)$ is the set of x's nearest neighbours. A reachability distance between two data points x and y is defined as reach-dist$_k(x, y) = \max\{k\text{-distance}(y), dist(x, y)\}$. The local reachability density of a data point x is then defined as the inverse of the average reachability distance based on $N_k(x)$ neighbours of x: $\text{lrd}_k(x) = 1 \bigg/ \left(\frac{\sum_{y \in N_k(x)} \text{reach-dist}_k(x,y)}{|N_k(x)|} \right)$. Finally, LOF assigns anomaly score (i.e. outlier factor) of a data point x as average local reachability density of x's neighbours divided by $\text{lrd}_k(x)$:

$$\text{LOF}_k(x) = \frac{\sum_{y \in N_k(x)} \text{lrd}_k(y)}{|N_k(x)| \cdot \text{lrd}_k(x)}, \tag{4}$$

A value significantly larger than 1 indicate outliers, where the considered point has much lower local reachability than its neighbours. Despite clear disadvantage in runtime, LOF has the capability to identify local outliers that could be skipped by other methods [42]. The algorithm also has been shown to perform well in cybersecurity domains [43].

D. Combination of Anomaly Detection Scores

Ensemble methods have been shown to reduce variance and bias of anomaly detection models in several applications [38]. In this section, we present methods to combine results from four aforementioned algorithms to create anomaly detection ensembles, in order to test their ability in insider threat detection. Employing four anomaly detection methods with different working principles, we expect to see differences in their detection results, especially under different conditions or scenarios. This creates potentials for improvements by combining the individual models. Specifically, we investigate combining schemes to create aggregated anomaly scores based on the average / maximum of individual anomaly scores, or based on majority votes of individual algorithms:

- *Averaging (AVG)*: Anomaly scores of individual models are normalized by rank, i.e. percentile transformation. The combined score of a single data point is then computed as the average (mean) over the different scores of the point.
- *Maximum (MAX)*: This approach assigns the combined anomaly score to a data point as the maximum of normalized (by rank) scores reported by individual models on the point. In essence, this combining scheme reports the highest anomaly signal (alarm) generated for a data sample by any of the participating models.
- *Voting (VOTE^v)*: In this scheme, majority vote is used to select outlier data points at each investigation budget. The parameter $v \in \{2, 3, 4\}$ dictates how many votes required in order to flag a data sample as anomalous.

IV. Experiments and Results

In this section, we present the datasets and experiments using the proposed approach for insider threat detection. Specifically, Section IV-A summarizes the datasets and data pre-processing techniques with temporal representations. Experiment settings and results are presented in Sections IV-B and IV-C – IV-E, respectively.

A. Datasets

The CERT insider threat datasets are publicly available for development and testing of insider threat mitigation approaches [44], [45]. In this paper, we employ releases 4.2 (CERT R4.2) and 6.2 (CERT R6.2). CERT R4.2 simulates a company with 1000 employees, where 70 are malicious insiders under three threat scenarios. This enables us to perform more flexible experiments in anomaly detection and provide better understanding of the models' behaviours. On the other hand, R6.2 is the newest CERT dataset. It depicts a much larger company with 4000 employees, containing only five malicious insiders (five threat scenarios, with only a single malicious user per scenario). This makes the detection task in CERT R6.2 much more challenging and realistic.

The CERT datasets consist of user activity logs (log on/off, email, web, file and thumb drive connect), company structure and user information. Following the process detailed in III-A, daily and weekly numerical features (i.e. original data representations) are extracted from the log files. Then, different temporal representations of the data are created as presented in III-B (Table II). Specifically, for concatenation representation, $\gamma = 2$ or $\gamma = 3$ most recent data instances of each user are joined. In percentile and mean/median difference representations, the window size value w is set to 7 days and 30 days for day data, and 30 days and 60 days for week data (4 weeks and 8 weeks). In the experiments, original data representation (Org) is also included as a baseline. Finally, we note that each malicious insider in the CERT data belongs to one of five popular insider threat scenarios: Data exfiltration (scenario 1), intellectual property theft (scenarios 2, 4, 5) and IT sabotage (scenario 3). Details of the threat scenarios can be found in [44].

Additionally, LANL dataset [46] and TWOS dataset [47] are also used for evaluation in this work. LANL consists of log files collected over 58 consecutive days. The logs contain anonymized real users' process, network flow, DNS, and authentication information. Furthermore, redteam (attacking) authentications is provided for the dataset, but without any additional information. In this paper, we employ only authentication and process logs of the LANL dataset. These events are collected from Windows-based desktops, servers, and active directory servers. Due to limitations of the dataset, 30 days of the logs are extracted to user-day numerical data. Temporal representations with window size of 7 days are then applied to LANL data. Similarly, TWOS dataset provides anonymized authentication, mouse, keystroke, email, and network captures from a student competition with the aim of emulating insider threats, by both masqueraders and traitors. The competition comprises of 24 participants in six teams over five days, which includes 12 instances of masquerading and one instance of traitor. With provided timestamps, we can only extract features from authentication, mouse, and keystroke activities. Due to the dataset's limited duration and size, data instances of 30 minutes of activities are extracted with temporal representations based on time window of one day.

Table I shows the statistics of the employed data in each type and the number of normal and malicious users. Table II describes the abbreviations used for temporal representations for each dataset in this paper.

TABLE I
SUMMARY OF THE DATASETS. SC: INSIDER THREAT SCENARIO. MALICIOUS USER COUNTS ARE IN PARENTHESES.

Data	Dur-ation	Feature count	User count	Class distribution					
				Normal	Sc 1	Sc 2	Sc 3	Sc 4	Sc 5
CERT R4.2	day	500	1000	329466	85 (30)	861 (30)	20 (10)		
	week	661	1000	66840	52 (30)	254 (30)	10 (10)		
CERT R6.2	day	888	4000	1393941	3 (1)	20 (1)	2 (1)	9 (1)	1 (1)
	week	1176	4000	283205	2 (1)	4 (1)	1 (1)	7 (1)	1 (1)
LANL	day	1215	11814	229691	Attack: 176 (98)				
TWOS	*30mins*	278	24	1458	Attack: 38 (13)				

TABLE II
TEMPORAL REPRESENTATION ABBREVIATIONS FOR EACH DATASET

Temp. rep.	CERT week	CERT day	LANL	TWOS	Description
Concat. (C_γ)	C_2, C_3	C_2, C_3	C_2, C_3	C_2, C_3	γ is number of instances concatenated
Percentile (P_w)	P_{30}, P_{60}	P_7, P_{30}	P_7	P_1	w denotes the size of time window in days
Mean Diff. (E_w)	E_{30}, E_{60}	E_7, E_{30}	E_7	E_1	
Med. Diff. (M_w)	M_{30}, M_{60}	M_7, M_{30}	M_7	M_1	

B. Experiment Settings

In training the anomaly detection algorithms, we randomly select a number of users n_u whose data in the first n_w weeks is included in the training process. Essentially n_u and n_w control the amount of data for training the models to represent computation and real-world limitations: Only a limited amount of data collected before the time of training can be used. In the following experiments, unless specified otherwise, we use training data of *randomly* selected $n_u = 200$ users (2000 for LANL data) in the first 50% of dataset duration ($n_w = 37$ for CERT and 2 for LANL). In the case of TWOS dataset, $n_u = 24$ and $n_w = 1$, due to the dataset's limitations. Since the training process is label free (unsupervised), test results are reported on the entire dataset. The experiments are repeated 10 times in each setting, and the averaged results are reported.

The experiments are performed on compute nodes with Intel Xeon E5-2683v4 CPU and 125GB of RAM. We implemented the data pre-processing and analysis steps using Python 3. AEs are implemented using Tensorflow [48]. In this paper, each AE has three hidden layers, where the size of the first and the third hidden layers are set to $input\_dimension/4$, and the middle hidden layer's size is set to $input\_dimension/8$. AEs are trained using Adam optimization [49] for 100 epoch each. Implementations from Scikit-learn [50] and PyOD [51] is used for IF, LODA, and LOF. For IF, the number of trees is set to 200, and 256 is used for max sample size. LODA is built with 400 histograms and $1/\sqrt{d}$ sparsity, while LOF's number of neighbours is set to 20. These parameter values are chosen empirically.

1) Performance metrics: In this work, the insider threat detection performance is measured using ROC and AUC metrics. ROC (Receiving Characteristic Curve) depicts the relationship between Detection rate (DR) and False Positive Rate (FPR) under different decision thresholds (i.e. different investigation budgets), and AUC (Area Under the Curve) summarizes ROC in a single numerical metric for comparison between models.

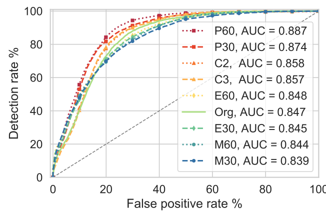
$$DR = \frac{TruePositive}{TruePositive + FalseNegative} \tag{5}$$

We also present DRs at critical IBs (see Section III) for better understanding of the performance at very low IBs.

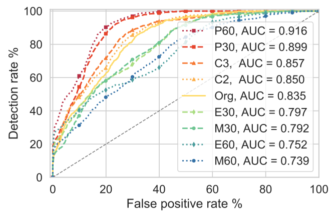
Furthermore, *user-based* results are presented in this section in terms of alarms that are raised per user through aggregation of raw (*instance-based*) anomalous alerts [35], [52]. Specifically, a normal insider (user) is misclassified if at least one of his/her data instances is classified as "malicious". On the other hand, a malicious insider is identified if at least one of his/her malicious data instances is labelled as "malicious" by the detection system. Consequently, we have two sets of performance metrics: Instance-based (DR, FPR, AUC) and User-based (UDR, UFPR, UAUC).

To compare between multiple algorithms or data representations on multiple datasets, we perform Friedman test, which is a non-parametric statistical test. The null hypothesis of Friedman test is that there is no significant differences between the variables. It is rejected when test statistic exceeds the critical value of the significance level ($p = 0.05$). Using average rank of a method on all datasets (R_j), the Friedman statistic is calculated as: $\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_1^k R_j^2 - \frac{k(k+1)^2}{4}\right]$, where N is the number of data points, and k is the number of methods. The statistic is distributed according to χ_F^2 with $k-1$ degrees of freedom [53]. If the null hypothesis is rejected, a post-hoc test (Bonferroni-Dunn) is carried out to compare the algorithms by pairs, i.e. the corresponding average ranks differ by at least the critical difference. The critical difference (CD) of the post-hoc test can be calculated based on k and N [53].

C. Detection Results



(a) Instance-based ROCs



(b) User-based ROCs

Fig. 4. ROCs of AE on R4.2 week data with different representations

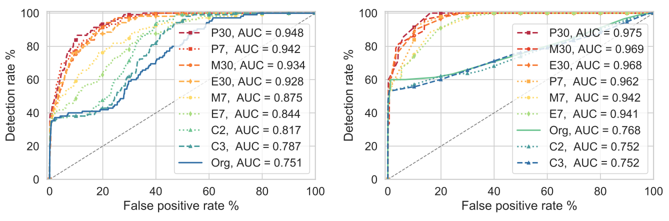
Instance-based anomaly detection results with different IBs are presented in Tables III and IV. Figures 4 and 5 show

TABLE III
INSTANCE-BASED ANOMALY DETECTION RESULTS WITH DIFFERENT INVESTIGATION BUDGETS ON CERT DATASETS. THE UNIT OF DR IS PERCENT (%)

Data	Data type	Temp. rep.	DR @ 0.1% IB				DR @ 1% IB				DR @ 5% IB				DR @ 10% IB				DR @ 20% IB				AUC			
			AE	IF	LODA	LOF	AE	IF	LODA	LOF	AE	IF	LODA	LOF	AE	IF	LODA	LOF	AE	IF	LODA	LOF	AE	IF	LODA	LOF
CERT R4.2	day	Org.	2.45	0.10	2.64	0.91	8.50	1.60	7.71	4.06	26.20	13.72	14.57	7.41	47.41	40.48	36.97	10.33	74.70	70.56	70.93	15.98	0.854	0.830	0.830	0.521
		C2	2.68	0.03	1.39	1.18	9.34	0.84	5.49	3.60	24.68	11.89	12.84	8.72	46.07	38.81	33.43	14.97	77.03	73.98	77.13	28.30	0.866	0.843	0.852	0.636
		C3	2.23	0.02	0.68	1.04	8.81	0.45	4.43	3.79	21.92	11.39	10.62	10.52	43.37	39.87	28.88	17.56	72.82	79.21	77.72	32.07	0.853	0.855	0.851	0.667
		P7	2.00	0.71	2.26	1.84	7.85	3.59	7.85	8.96	35.75	24.20	32.37	35.67	63.02	52.71	55.63	57.52	87.60	84.63	78.73	77.95	**0.903**	0.882	0.870	0.859
		P30	2.31	0.72	3.55	1.33	12.08	3.66	12.30	9.25	36.15	23.82	36.14	30.37	62.39	54.99	59.27	52.22	88.43	86.59	81.18	77.47	0.902	0.890	0.882	0.858
		E7	3.48	0.58	2.25	2.05	10.57	3.44	9.55	4.84	20.86	15.97	19.47	12.67	33.55	37.87	32.59	20.14	55.36	70.18	55.64	32.67	0.788	0.835	0.783	0.612
		E30	3.41	0.52	2.64	1.2	10.82	3.31	10.52	4.93	21.15	15.09	21.89	10.66	35.37	41.00	35.56	16.05	59.22	77.38	62.68	27.52	0.797	0.859	0.814	0.590
		M7	2.62	0.61	2.64	1.35	9.58	3.52	9.31	4.18	20.40	18.58	19.95	10.97	32.14	38.23	34.76	16.50	54.92	66.54	57.24	27.47	0.771	0.819	0.781	0.604
		M30	2.10	0.54	2.75	1.43	8.31	3.82	10.39	4.27	21.99	22.10	22.52	10.64	36.94	43.71	39.55	15.98	59.57	72.49	63.27	27.29	0.786	0.841	0.808	0.623
	week	Org.	2.53	0.03	0.25	2.25	9.97	0.85	6.14	7.18	22.41	6.71	20.98	17.47	40.13	23.89	33.70	22.82	72.53	70.47	67.18	32.06	0.847	0.825	0.844	0.611
		C2	2.88	0.03	0.16	2.75	9.21	0.66	2.85	6.23	22.15	4.78	16.20	18.13	41.65	19.53	30.98	24.62	77.03	72.75	64.59	34.49	0.858	0.838	0.842	0.624
		C3	2.75	0	0	2.69	7.12	0.66	1.42	6.30	20.92	3.64	10.25	16.71	40.06	15.76	26.30	24.08	77.85	66.90	64.72	35.22	0.857	0.827	0.837	0.617
		P30	5.76	0.22	2.47	5.6	10.57	0.44	9.72	9.72	27.63	13.23	31.46	29.59	52.18	46.30	57.56	54.49	81.55	86.46	83.99	88.20	0.874	0.881	0.889	0.897
		P60	7.66	0.16	3.32	7.22	14.34	0.60	13.10	13.07	32.56	13.64	36.58	31.46	55.09	47.41	59.49	56.99	83.67	87.12	86.58	88.58	0.887	0.884	0.900	**0.901**
		E30	6.11	0.25	2.25	5.79	14.94	1.17	9.81	13.01	28.92	12.34	26.65	24.21	46.49	31.84	43.35	36.99	70.79	72.78	67.82	63.58	0.845	0.843	0.833	0.804
		E60	8.39	0.28	2.22	8.32	16.23	1.42	10.09	14.72	29.30	11.46	28.77	24.15	48.48	32.69	48.45	36.58	70	74.08	70.57	64.27	0.849	0.848	0.852	0.824
		M30	3.64	0.32	2.03	3.96	13.16	2.12	10.32	9.94	31.58	17.53	29.46	22.82	47.53	38.16	46.23	36.23	69.75	71.80	70.16	61.04	0.839	0.847	0.840	0.797
		M60	5.03	0.28	3.01	5.44	9.81	2.85	14.08	12.44	29.08	20.22	34.72	25.16	48.51	42.78	51.96	38.01	68.99	76.65	75.13	62.12	0.844	0.862	0.863	0.806
CERT R6.2	day	Org.	26.29	0	8.00	20.00	39.14	6.00	22.57	35.24	67.43	49.71	52.00	38.10	84.29	79.71	88.86	40.00	96.29	90.86	100	41.90	0.952	0.924	0.949	0.751
		C2	24.57	0	2.29	20.95	38.29	4.00	22.00	35.24	74.00	48.29	55.71	37.14	91.14	79.14	93.43	38.10	99.43	92.57	96.86	47.62	0.965	0.923	0.949	0.817
		C3	22.86	0	0.57	22.86	36.86	5.14	16.00	35.24	80.29	49.71	57.71	38.10	95.43	73.71	90.57	38.10	99.14	86.86	94.57	42.86	0.969	0.909	0.942	0.787
		P7	31.14	0.29	16.00	27.62	45.43	21.43	46.29	40.00	81.71	72.00	71.14	69.52	98.57	92.00	86.00	78.10	100	98.86	96.86	91.43	**0.977**	0.958	0.958	0.942
		P30	33.71	0	13.71	37.14	43.43	15.71	42.57	44.76	81.43	82.57	69.71	65.71	96.29	92.29	84.57	83.81	100	97.14	97.14	93.33	0.974	0.960	0.957	0.948
		E7	30.57	0	10.29	27.62	37.71	6.29	31.71	39.05	52.57	48.86	43.14	41.90	64.00	80.86	55.43	49.52	89.14	98.29	74.29	63.81	0.913	0.936	0.878	0.844
		E30	28.86	0	10	31.43	40.00	3.71	33.43	37.14	55.43	52.00	44.00	58.10	67.43	90.57	58.86	76.19	91.43	97.43	77.71	87.62	0.926	0.942	0.892	0.928
		M7	28.00	0.29	9.14	23.81	37.71	12.00	30.57	38.10	53.71	46.29	45.43	50.48	68.86	67.14	54.00	60.00	83.71	80.57	71.71	76.19	0.906	0.890	0.849	0.875
		M30	25.71	0.29	12.29	12.38	37.43	9.43	29.71	40.95	58.00	47.71	44.29	58.10	74.29	64.29	55.71	75.24	92.00	86.00	74.00	91.43	0.935	0.901	0.862	0.934
	week	Org.	38.00	0	0	38.67	63.33	0	8.67	64.00	75.33	10.00	42.00	74.00	94	44.67	82	78.67	98.67	87.33	99.33	86.00	0.973	0.870	0.933	0.925
		C2	33.33	0	0	34.00	57.33	0	4.67	60.67	72.00	14.67	41.33	70.67	91.33	44.67	76	75.33	99.33	82.00	94.67	82.00	0.967	0.857	0.926	0.901
		C3	32.00	0	0	34.00	50.67	0	1.33	53.33	71.33	15.33	40.67	64.00	88.67	34.67	78.67	70.67	94.67	74.00	90.00	78.00	0.959	0.837	0.915	0.881
		P30	40.00	0	9.63	33.33	66.67	0	39.26	66.67	85.93	17.78	74.81	91.85	97.04	52.59	96.30	100	100	85.93	100	100	0.981	0.889	0.970	**0.985**
		P60	40.74	0	2.96	32.59	66.67	0	51.11	66.67	78.52	12.59	74.07	85.93	99.26	42.22	94.81	99.26	100	90.37	100	100	0.979	0.881	0.971	0.981
		E30	30.37	0	0	31.11	63.70	0	10.37	63.70	70.37	6.67	50.37	74.81	82.22	21.48	65.93	83.70	96.30	74.81	86.67	99.26	0.958	0.829	0.909	0.966
		E60	24.44	0	0	25.19	66.67	0	6.67	66.67	71.85	3.70	51.11	75.56	85.19	20.74	70.37	90.37	99.26	77.78	91.85	100	0.966	0.837	0.925	0.972
		M30	33.33	0	0	35.33	60.67	0	9.33	62.67	74.67	10.00	41.33	80.00	85.33	35.33	62.67	92.00	98.00	76.00	88.00	98.67	0.961	0.842	0.908	0.971
		M60	35.33	0	0	38.67	61.33	0.67	7.33	60.00	71.33	13.33	37.33	76.00	82.67	42.00	64.00	92.67	98.67	86.67	97.33	98.67	0.962	0.864	0.913	0.969

TABLE IV
INSTANCE-BASED ANOMALY DETECTION RESULTS WITH DIFFERENT INVESTIGATION BUDGETS ON LANL AND TWOS DATASETS. THE UNIT OF DR IS PERCENT (%)

Data	Rep.	DR @ 1% IB				DR @ 5% IB				DR @ 10% IB				DR @ 20% IB				AUC			
		AE	IF	LODA	LOF	AE	IF	LODA	LOF	AE	IF	LODA	LOF	AE	IF	LODA	LOF	AE	IF	LODA	LOF
LANL	Org.	20.28	34.77	25.23	6.59	53.86	54.60	45.00	14.55	71.31	64.15	58.47	24.43	84.83	76.14	73.98	51.59	**0.908**	0.884	0.863	0.776
	C2	17.76	24.59	21.29	8.59	51.18	48.82	38.71	18.82	69.53	59.29	53.29	33.41	85.06	73.88	68.35	56.47	0.896	0.869	0.841	0.787
	C3	13.49	20.60	16.51	7.71	41.45	41.33	33.01	15.90	57.23	54.82	44.58	33.37	76.14	71.33	61.93	59.16	0.862	0.846	0.814	0.789
	P7	11.82	6.54	4.03	5.79	44.78	26.04	19.12	10.19	64.91	44.91	32.45	17.86	85.16	65.28	51.82	32.58	0.897	0.804	0.749	0.689
	E7	17.48	11.45	11.57	5.03	47.67	36.73	30.19	17.99	69.43	53.71	43.02	31.57	83.02	70.44	62.14	57.23	0.883	0.834	0.789	0.780
	M7	17.48	8.68	13.46	4.15	44.40	31.19	26.92	16.98	62.26	47.55	41.26	30.44	79.12	65.66	60.13	49.18	0.864	0.813	0.779	0.763
TWOS	Org.	2.63	0.26	0	0	13.42	18.95	8.95	7.89	23.42	29.47	17.11	15.79	42.89	44.47	34.21	34.21	0.714	0.708	0.577	0.632
	C2	0	0	0	0	5.53	10.53	8.68	10.53	14.21	19.74	13.68	15.79	33.42	37.89	26.58	31.58	0.641	0.673	0.591	0.613
	C3	0.79	0	0	0	5.26	5.79	4.21	10.53	10.26	11.84	9.47	15.79	19.47	28.16	21.32	23.68	0.599	0.634	0.555	0.565
	P1	0	0.26	0.53	0	16.32	12.11	16.32	2.63	40.53	27.89	34.21	2.63	55.26	57.63	54.47	18.42	**0.794**	0.780	0.777	0.578
	E1	7.37	1.32	2.11	2.63	11.05	13.68	13.68	7.89	18.95	22.89	21.84	10.53	40.26	39.21	38.16	31.58	0.709	0.716	0.691	0.669
	M1	4.47	4.74	3.42	5.26	8.16	17.63	13.68	7.89	20.53	36.05	20.26	13.16	39.21	44.21	37.63	36.84	0.708	0.737	0.671	0.692



(a) Instance-based ROCs (b) User-based ROCs

Fig. 5. ROCs of LOF on R6.2 day data with different representations

instance-based and user-based ROCs on R4.2 week data and R6.2 day data with different temporal data representations. Overall, the results achieved using *autoencoder* and *percentile* representation are very promising, given that the results are obtained under unsupervised setting with very limited training data (a small set of only 200 unidentified users in 37 weeks, for CERT data). On CERT R4.2, the approach was able to detect 77% of the malicious users by investigating only 1% of the most anomalous instances (1% IB). Also, at only 5% IB, nearly 100% of 70 malicious insiders are detected. Figures 4 and 5 also show the differences in reporting results based on data instances and users, where the AUC achieved on user-based results could be higher and the differences between temporal data representations are more pronounced.

We note that normal data dominates the distribution in all employed datasets (Table I). Thus, the FPR (normal data wrongly flagged) obtained under each IB is very similar to the IB, e.g. at 1% IB, FPRs ranges from 0.96% to 0.99% on CERT R4.2 week data. Furthermore, as IB represents different human resource levels for investigating anomaly detection output, i.e. different amounts of data flagged, a suitable IB can be selected based on deployment conditions. For example, on CERT R4.2 day data (Table I), 1% / 5% / 10% IBs are equivalent to 3300 / 16500 / 33000 alerts, or approximately 7 / 33 / 66 alerts per day over the dataset's duration.

1) Results by learning algorithms: Figure 6 shows a comparison of the algorithms by ROC. Performing Friedman

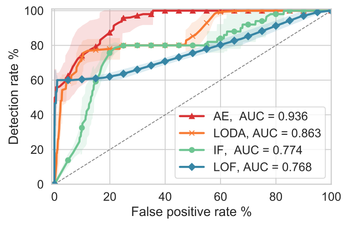
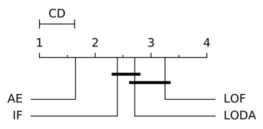
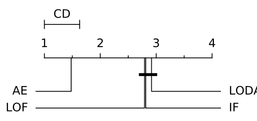


Fig. 6. User-based ROC by learning algorithms on original R6.2 day data



(a) Instance-based results



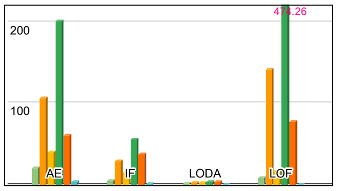
(b) User-based results

Fig. 7. Critical Difference (CD) diagrams of algorithms' results by instance and by user. Average rank of each algorithm is shown on the scale. Two *linked* entries (connected by a horizontal black bar) are not significantly different, i.e. rank difference is less than CD.

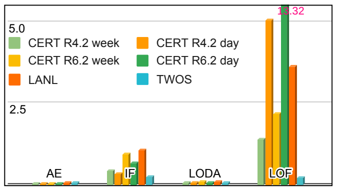
test on both user-based and instance-based anomaly detection performance of the algorithms, the null hypotheses are easily rejected ($\chi_F^2 = 38.78, p = 2 \times 10^{-8}$ and $\chi_F^2 = 40.28, p = 9 \times 10^{-9}$), which means there are significant differences between the algorithms. Figure 7 presents the critical difference diagram obtained using post-hoc test, where average ranking of each algorithm and whether they are significantly different are shown. Additionally, training and prediction times per data instance of each algorithm are presented in Figure 8.

Overall, it is shown that AE achieves the best performance in detecting anomalies representing insider threats, especially at very low FPRs. For example, at only 0.1% IB, AE is able to detect 60% of the malicious insiders from R6.2 week data with P_{30} representation, while IF requires 8% IB to reach a similar UDR in the same setting.

LOF shows interesting results, where it performs well when data counts are lower (R4.2 and R6.2 week) and only on percentile representations. We believe that its ability to outperform in some cases is due to the "local" characteristics of its detected outliers, which may be missed by other algorithms (III-C4). However, LOF suffers from very long training and prediction time. On the remaining two algorithms, LODA achieves very similar results to IF (Table III and Figure 7), and at very low time complexity. This makes it suitable to time critical on-line detection tasks.



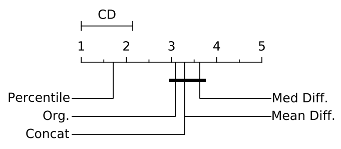
(a) Training time (s)



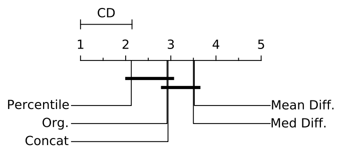
(b) Prediction time per instance (ms)

Fig. 8. Average training time and prediction time per data instance of the algorithms on different data. Out of chart values are noted in red.

Experiments in Section IV-D provide further insights into the detection performance of the algorithms. We note that the datasets characteristics (predominantly normal behaviours – Table I), and experiment settings (Section IV-B) could be partly the reason to AE's outstanding performance in this section. On the other hand, Section IV-D shows that LODA and IF can be more robust to changes in deployment conditions.



(a) Instance-based results



(b) User-based results

Fig. 9. Critical Difference (CD) diagrams of results by data representations.

2) *Results by data representations:* On data representations, Table III and Figure 4 show that percentile (P_w) is the best representation of data for anomaly detection. This is confirmed by Friedman test (hypotheses rejected, $\chi_F^2 = 21.46, p = 0.0003$ and $\chi_F^2 = 12.48, p = 0.01$), and post-hoc tests, as shown in CD diagrams in Figure 9. Percentile representation allows the algorithms to achieve significantly better results than on the original data. Concatenation shows slight improvements in some cases, while mean / median differences are unable to surpass the original data. In some cases, such as R4.2 day data, mean/median difference even deteriorates the AUC.

The observations suggest that percentile representation, although encoding the data change by omitting the absolute values, successfully captures the change in user behaviours while avoiding noises in the data. At the same time, maintaining the absolute values of changes as in mean and median difference representation seems to create noise and decreases the detection performance (Figure 4). Finally, on concatenated representation, the results show that it is hard to facilitate meaningful automatic comparisons between data related to different points in time.

D. Results on Different Conditions for Training Anomaly Detection Algorithms

In the following, we assume P_{30} data representation and analyze ML algorithms on CERT R4.2 data types under different sizes of training data and conditions.

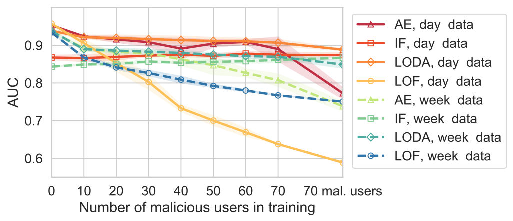


Fig. 10. UAUC by number of malicious users in R4.2 training data

1) *Anomaly detection performance under training data poisoning conditions:* In this experiment, instead of using data

from 200 randomly selected users, we deliberately introduce malicious users' data during training. The number of malicious users included varies from 0 (pure normal training data) to all 70 malicious users of CERT R4.2 (35% of training users are malicious). In an extreme case, we use only data of the 70 malicious insiders for training the algorithms. This is to analyze how the anomaly methods respond to data poisoning, where malicious data is presented at high density in training data, which may corrupt the ML models into mislabelling malicious as normal [54], [55].

Figure 10 shows the user-based AUC by the algorithms on R4.2 data under different number of malicious users in training. Overall, it is clear that ensemble-based algorithms, IF and LODA, are very robust to the data poisoning attack. Using IF, AUC even increases slightly with the presence of malicious data in training. This can be explained through its properties, where small amount of contamination in training data allows trained IF trees to better model the anomalies that may appear in the data [40].

On the other hand, performance of AE and LOF deteriorates as the amount of malicious users in training increases. It seems that with high malicious data presence in training set, AE may incorporate some malicious actions as normal in its trained model through the encoding-decoding process. Thus, it is unable to detect those types of behaviours in testing. Similarly, in the case of LOF, high amount of poisoning data injected into training may increase the local density of malicious data points, which may trick LOF to assign lower anomaly scores to those points.

Nevertheless, AE was able to maintain a better performance than other algorithms, up to 30 malicious users in training data (15%). We note that in practice, the amount of malicious users in training data for insider threat detection approaches is typically very small [6], hence the use of AE is still preferred. Moreover, LODA shows great balance between detection performance and robustness, making it a prime candidate in severe poisoning condition.

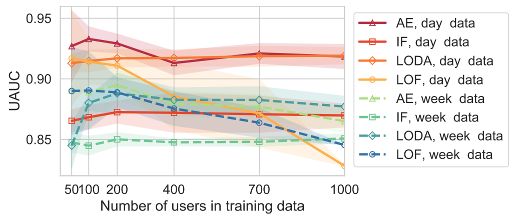


Fig. 11. UAUC by number of users in R4.2 training data

2) Effects of the number of users in training data: Without having to collect groundtruth for training data, the unsupervised learning approach permits the use of as many users in training data as possible, at the cost of a higher computational cost. In this experiment, we vary the number of (randomly selected) users to include in training data from 50 to 1000 (maximum amount) users in CERT R4.2. User-based AUCs are presented in Figure 11. Results show that except LOF, in most cases, the performance is largely unchanged. However, results varies more (i.e. unstable), when less data (less number of users) are used in training. LODA and AE's UAUC increase slightly to 200 users in training data, but AE's performance decreases slowly as the number of users increase in training.

Behaviours of AE and LOF can be explained through results in IV-D1, where a larger number of training users creates a higher chance of malicious users to be included in training data, hence lowering their effectiveness. This shows that maintaining a relatively small number of users (200) in training data not only reduces the computational cost but also potentially gives more robust results.

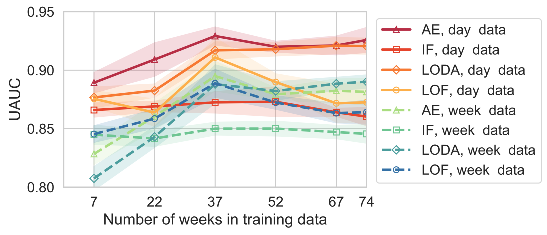


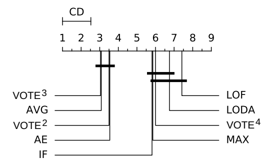
Fig. 12. UAUC by number of weeks in R4.2 training data

3) Effects of training data duration: Similar to the number of users in training data, the number of weeks can be adjusted, too. This experiment varies the parameter from 7 (10% of data time range) to 74 (100%). This setting also simulates online learning conditions, where the algorithms are retrained / enhanced with the arrival of new data. Figure 12 shows user-based AUCs on CERT R4.2 data types. As in the previous experiments, IF's performance is maintained through different number of weeks used in training data. On the other hand, detection performance of AE and LODA rises until about 50% of the data duration is used in training (37 weeks), then remains largely unchanged. LOF shows similar improvements in the first half of data duration, but quickly deteriorates after that. In fact, more malicious insider activities appear in the second half of CERT data than in the first half [44]. Hence, it can be concluded that for AE and LOF, more training data may help to improve results, but only to a point where the improvements are negated by the introduction of malicious samples in training data (IV-D1). This experiment shows the advantage of online learning methods, such as LODA, where results can be progressively improved over time with more training data.

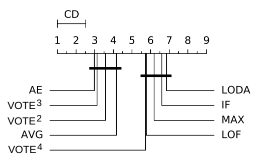
E. Ensembles of Anomaly Detection Models

As shown in previous sections, four anomaly detection algorithms show various effectiveness on the datasets, especially under different training conditions. This section presents the results by the ensemble schemes described in III-D. Combining the anomaly scores to create ensembles, the best results (measured by AUC) by individual unsupervised ML algorithms are maintained in almost all cases, by VOTE$^{\nu}$ and AVG. In some cases, ensembles increase the detection performance. For example, VOTE^3 achieves AUCs of 0.909 and 0.907 on CERT R4.2 day and week data, respectively, which improves over the individual components (Table III). Performing Friedman test, hypotheses rejected on both AUC and UAUC comparisons between the learning algorithms and ensemble schemes ($\chi_F^2 = 148, p = 4 \times 10^{-28}$ and $\chi_F^2 =$

$120, p = 2 \times 10^{-22}$). Figure 13 shows critical difference diagrams for the post-hoc tests on instance-based and user-based results. On the other hand, as shown in the figure, there are no significant differences detected between AVG, VOTE^{2,3}, and AE, and these methods all significantly outperform the remaining algorithms (VOTE^4, MAX, IF, LODA, LOF).

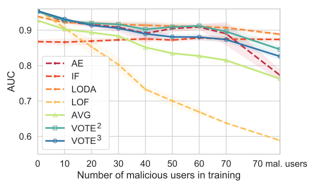


(a) Instance-based results

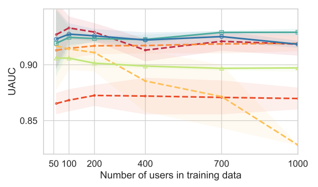


(b) User-based results

Fig. 13. Critical Difference diagrams of results by learning algorithms and ensembles



(a) UAUCs under training data poisoning



(b) UAUCs with different number of users in training data.

Fig. 14. UAUC of learning algorithms and ensembles under different training conditions on CERT R4.2 day data.

Furthermore, we explore the effects of ensemble schemes under different training conditions as in IV-D. Figure 14 shows results (in UAUC) of ML algorithms and ensembles on CERT R4.2 day data under different training conditions. It is apparent that voting schemes, especially VOTE^2, achieve the best or near best detection performance in almost all cases. Moreover, with more training data (Figure 14b), VOTE^2 is able to outperform all other algorithms. On the other hand, while AVG shows similar results to voting-based ensembles, Figure 14 shows that combination by averaging is not favoured under adverse learning conditions.

On time requirement consideration, it is noteworthy that while the computation cost (and hence time) of combining scores by the individual algorithms is insignificant, in order to create an ensemble, all components need to be trained. Hence, the ensembles are restricted by the slowest algorithm (e.g. LOF) in both training and predicting. In the particular cases of the datasets employed in this work, the time required to train and evaluate detection models is reasonable (Fig. 8), hence permitting their use in the current form. In other real-world applications, lightweight components can be selected to create ensembles to avoid time and computation cost burdens.

V. Discussions and Comparisons

In this part, CERT R6.2 is employed for testing purposes, as it represents more malicious insider threat cases and better mimics real-world conditions (only 5 malicious insiders). We study anomaly detection results given by the proposed system under specific scenarios and show how security analysts may use these to further investigate and identify malicious behaviours. Results on each insider threat scenario and comparisons with other works in the literature are also presented.

A. *Case study of anomaly alerts*

Using a unique `id` for each data instance used in anomaly detection process, the corresponding course of original user actions can be quickly examined, once an anomaly alert is raised. A true anomaly alert example on CERT R6.2 is associated with actions of user `PLJ1771` – an IT administrator – on August 12, 2010. Using AE and P_{30} representation, the data instance was assigned an anomaly alert with 99.99% confidence (i.e. the data instance has anomaly score higher than 99.99% of CERT R6.2 data). By studying the action sequence of the user on the day, his/her malicious behaviour can quickly be confirmed: The user visits several sites providing computer monitoring software, downloads a keylogger and puts it on a USB. Later in the day, they log onto `PC-3999`, which belongs to their supervisor – `HIS1706`, and start keylogging on the PC. This corresponds to the behaviours of a "disgruntled system administrator" in the CERT dataset [44].

Another true anomaly alert is raised with 99.93% confidence for activities of user `CDE1846` on March 22, 2011, in which the user logged in after work hours to `PC-5014`, which belongs to another user. Then, he/she opens and emails multiple documents to his/her personal email.

On the other hand, several false alarms generated by the anomaly detection system are worth investigating as well. For example, false alarms are raised for user `YNW2855` on September 24, 2010 and user `RRH3057` on November 03, 2010 with confidence of 99.90% and 99.99%. Investigating the original user activities on both days reveals multiple actions (file accesses, website visits) very late after work hours (around 10 PM). While these examples may not depict malicious intentions (as per dataset's groundtruth), their anomalous nature needs to be inspected to ensure the safety of the system and data.

These case studies show how a system administrator may leverage the anomaly detection system's output to identify the true nature of alerts as well as perform appropriate response, with reference to the reorganized course of actions (by user, time) in log files. Furthermore, in manually investigating the original user's activities corresponding to each alert, the analyst may have access to more restricted information that was not incorporated in the ML system's training, such as email content, to make informed decisions.

B. *Detection performance on insider threat scenarios*

As mentioned in IV-A, there are five malicious insiders in CERT R6.2, each depicts a unique threat scenario. This part examines the detection results on the scenarios.

Table V presents the malicious insiders and detection results using AE and week data with P_{60} representation of CERT R6.2. Detection delays (at 10% IB), which is the time between the first malicious action and when the malicious user is detected, are also presented in the table. As the table shows, scenarios 1, 3, and 4 can be detected very easily using the proposed system with only 0% to 0.04% FPR (or 0.04 to 0.15% normal users flagged wrongly). All malicious instances of those users are detected with less than half a percent (0.32%) FPR. Scenarios 1 and 3 can also be detected very quickly.

On the other hand, threat scenarios 2 and 5 are much harder to detect, resulting in FPRs of 3.07% and 8.36%, respectively. At a UFPR of 26.46%, a system analyst will need to inspect more than 1000 users to identify the malicious user `MBG3183`. The descriptions of these scenarios show much less intrusive malicious behaviours than the other three scenarios [44]. For example, in scenario 5, "a member of a group decimated by layoffs uploads documents to Dropbox, planning to use them for personal gain". This explains the lower detection performance on these two scenarios, as they are easy to be mistaken as normal activities.

TABLE V
DETECTION PERFORMANCE ON SPECIFIC INSIDER THREAT SCENARIOS. DD: DETECTION DELAY.

Threat Scen.	Username	Min. FPR to detect	FPR to detect *all* malicious instances	UFPR	DD - week data (days)	DD - day data (days)
1	`ACM2278`	0.02%	0.06%	0.12%	3.22	0.22
2	`CMP2946`	3.07%	6.97%	13.00%	5.74	0.74
3	`PLJ1771`	0.00%	0.00%	0.04%	2.79	0.79
4	`CDE1846`	0.04%	0.32%	0.15%	5.68	3.07
5	`MBG3183`	8.36%	8.36%	26.46%	4.57	0.57

C. Robustness of the trained models

For this analysis, we use an anomaly detection model trained on one CERT dataset (R4.2) to detect new anomalies on another one (R6.2). As CERT R6.2 is a newer version with changed generative models and a larger size [44], this experiment can be seen as applying anomaly detection model of a company for a different one. User-based AUCs on CERT R6.2 week data by AE models trained using the original and P_{30} data representations are shown in Figure 15. The figure shows that anomaly detection model trained using CERT R4.2 data with P_{30} representation can achieve very good AUC when tested on CERT R6.2 (UAUC=0.908). The result is vastly improved over a model trained using R4.2 via the original data representation (UAUC=0.511). This demonstrates the robustness of the proposed system when percentile data representation is used. The result suggests that modelling user data points in percentile representation brings in the temporal information of the user's previous data instances and therefore allows the model to generalize better.

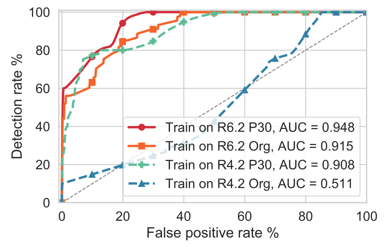


Fig. 15. UAUC of models trained on CERT R4.2 and R6.2 data when tested on R6.2

D. Comparison against time series data extraction

In this section, we perform comparison between the temporal data representation and time series feature extraction. Time series features are extracted from CERT R4.2 day data with a rolling window size of 30. `tsfel` package [56] is employed with comprehensive extraction settings. Due to computation overhead of the time series feature extraction process, a sample set of 200 randomly selected users in CERT R4.2 is used. Each feature in the original data is treated as a time series to extract 132 time series features, using tsfel. Given LODA's low time complexity and good performance, as shown in Section IV, we apply it for anomaly detection on the time series extracted features.

Results obtained show an AUC of 0.78 using time series extracted features. In comparison, using original data and percentile representation generate AUCs of 0.81 and 0.87, respectively, under the same conditions. This shows the advantage of our approach to traditional time series extraction approaches for temporal data in this application. We believe that by focusing on using the temporal window to define a baseline comparison for each new data instance, changes in user behaviours are easier to detect than from time series data via time windows, where all data points in the window contributes similarly to the output.

E. Comparative study

The proposed system shows clear advantages in both detection performance and the ability to generalize when compared to other works in the literature employing unsupervised anomaly detection methods for insider threat detection on the CERT datasets [11], [13]–[18], [57].

On CERT R4.2, our proposed approach obtained AUC of 0.907 and 0.909 on week and day data (Section IV-E), outperforming previous works [13]–[15] that used HMM and OneClass-SVM, which achieved AUC of 0.83 and 0.89, respectively. On CERT R6.2 data, our approach achieved AUC of 0.977 and 0.981 on day and week data. In comparison, recent best AUCs achieved on R6.2 day data were 0.814 (Matterer *et al.* [17]), and 0.956 (Liu *et al.* [57], on only 3 malicious insiders). This demonstrates the advantage of our approach in embedding temporal information in data representation, as opposed to using a learner with temporal learning capabilities such as Long Short-Term Memory [17] and Markov models [13]. On R6.2 week data, recently, [18] achieved AUC of 0.999. However, they only tested on 500 users and 1 easy-to-detect malicious user (`ACM2278`, see V-B). Under the same malicious user consideration, our approach posts an AUC of 0.9996. Similarly, log2vec [11] achieved AUC of 0.93 with only 6 malicious users and 12 normal users in CERT R6.2 included in evaluation, while our result (higher AUC) is obtained on the full dataset. Furthermore, to the best of our knowledge, no other work has been able to show the ability of the anomaly detection solutions to generalize (robustness) on other datasets as illustrated in V-C.

Finally, on LANL, our approach achieves comparable results to unsupervised approaches in the literature [11], [30]. Note that other recent works on the datasets achieved higher

AUCs, but they used supervised learning, as in [31], or presented results by log lines [24], which significantly increase the amount of alerts.

VI. Conclusions and Future Work

In this research, an unsupervised ML based anomaly detection approach for insider threat detection is presented. To this end, four different anomaly detection algorithms with different working principles are employed. The methods are studied using different representations of data with temporal information, including concatenation, percentile and mean or median difference. In doing so, the aim is to describe the changes in user activities that could highlight the detection of anomalous behaviours. Experiments under different constrained conditions are performed on publicly available datasets and comprehensive results are reported. Results show that Autoencoder using percentile representation of data is the best combination for anomaly detection. Temporal data representation in percentile format achieves significant improvements over original extracted data, which enables effective insider threat detection under very low investigation budgets and generalizes well on new data. Moreover, experiments demonstrate the robustness of LODA, which may suggest its use under extreme conditions and for low time complexity on-line learning and prediction. Furthermore, when training resources permit, voting-based ensemble of anomaly detection can be used to improve detection performance and robustness. Comparing with the existing literature, our approach shows clear advantage in detection performance and ability to generalize to work under different environments.

Future work will investigate other ML approaches, such as semi-supervised and adversarial techniques, and data availability for anomaly detection. Finally, informed attackers' actions and adversarial attacks can also be introduced to further examine the performance under more adverse conditions.

Acknowledgement

This research has been enabled in part by support provided by Natural Science and Engineering Research Council of Canada (NSERC) and Compute Canada (www.computecanada.ca). Duc C. Le gratefully acknowledges the support by the Killam Trusts and the province of Nova Scotia. The research is conducted as part of the Dalhousie NIMS Lab at: https://projects.cs.dal.ca/projectx/.

References

[1] I. Homoliak, F. Toffalini, J. Guarnizo, Y. Elovici, and M. Ochoa, "Insight into insiders and IT: A survey of insider threat taxonomies, analysis, modeling, and countermeasures," *ACM Computing Surveys*, vol. 52, no. 2, pp. 30:1–30:40, Apr. 2019.

[2] CSO, CERT Division of SEI-CMU, U.S. Secret Service, and KnowBe4, "The 2018 U.S. state of cybercrime survey," IDG, Tech. Rep., 2018. [Online]. Available: https://www.idg.com/tools-for-marketers/2018-u-s-state-of-cybercrime

[3] Crowd Research Partners, "2018 insider threat report," CA Technologies, Tech. Rep., 2018, https://crowdresearchpartners.com/insider-threat-report.

[4] R. Campagna, "Enterprise insider threats on the rise," https://www.cyberdefensemagazine.com/enterprise-insider-threats-on-the-rise/.

[5] M. L. Collins *et al.*, "Common sense guide to mitigating insider threats, fifth edition," The CERT Insider Threat Center, Tech. Rep., 2016, CMU/SEI-2015-TR-010.

[6] A. Azaria, A. Richardson, S. Kraus, and V. S. Subrahmanian, "Behavioral analysis of insider threat: A survey and bootstrapped prediction in imbalanced data," *IEEE Transactions on Computational Social Systems*, vol. 1, no. 2, pp. 135–155, Jun. 2014.

[7] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[8] National Cybersecurity and Communications Integration Center, "Combating the insider threat," 2014, https://www.us-cert.gov/security-publications/Combating-Insider-Threat.

[9] L. Liu, O. De Vel, Q.-L. Han, J. Zhang, and Y. Xiang, "Detecting and Preventing Cyber Insider Threats: A Survey," *IEEE Communications Surveys & Tutorials*, 2018.

[10] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[11] F. Liu *et al.*, "Log2vec: A Heterogeneous Graph Embedding Based Approach for Detecting Cyber Threats within Enterprise," in *ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[12] P. Parveen, J. Evans, B. Thuraisingham, K. W. Hamlen, and L. Khan, "Insider threat detection using stream mining and graph mining," in *IEEE Third International Conference on Privacy, Security, Risk and Trust*, 2011, pp. 1102–1110.

[13] T. Rashid, I. Agrafiotis, and J. R. Nurse, "A new take on detecting insider threats," in *Int. Workshop on Managing Insider Security Threats*, 2016.

[14] D. C. Le and N. Zincir-Heywood, "Evaluating insider threat detection workflow using supervised and unsupervised learning," in *IEEE Security and Privacy Workshops (SPW)*, 2018.

[15] M. Aldairi, L. Karimi, and J. Joshi, "A trust aware unsupervised learning approach for insider threat detection," in *IEEE Int. Conf. on Information Reuse and Integration for Data Science*, 2019, pp. 89–98.

[16] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," in *AAAI Workshop on Artificial Intelligence for Cyber Security*, 2017.

[17] J. Matterer and D. Lejeune, "Peer group metadata-informed LSTM ensembles for insider threat detection," *International Florida Artificial Intelligence Research Society Conference*, pp. 62–67, 2018.

[18] L. Liu, C. Chen, J. Zhang, O. De Vel, and Y. Xiang, "Unsupervised insider detection through neural feature learning and model optimisation," in *Lecture Notes in Comp. Sci.*, vol. 11928 LNCS. Springer, 2019.

[19] H. Eldardiry *et al.*, "Multi-domain information fusion for insider threat detection," in *IEEE SPW*, 2013.

[20] P. Parveen and B. Thuraisingham, "Unsupervised incremental sequence learning for insider threat detection," in *The IEEE International Conference on Intelligence and Security Informatics*, 2012.

[21] 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*, vol. 11, no. 2, 2017.

[22] T. E. Senator *et al.*, "Detecting insider threats in a real corporate database of computer usage activity," in *ACM SIGKDD Conf. (KDD)*, 2013.

[23] B. Bose, 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.

[24] A. Brown, A. Tuor, B. Hutchinson, and N. Nichols, "Recurrent neural network attention mechanisms for interpretable system log anomaly detection," in *Proceedings of the First Workshop on Machine Learning for Computing Systems*, 2018, pp. 1–8.

[25] D. C. Le, S. Khanchi, N. Zincir-Heywood, and M. I. Heywood, "Benchmarking evolutionary computation approaches to insider threat detection," in *ACM Genetic and Evolutionary Computation Conf.*, 2018.

[26] D. C. Le, N. Zincir-Heywood, and M. I. Heywood, "Dynamic insider threat detection based on adaptable genetic programming," in *IEEE Symposium Series on Computational Intelligence*, 2019.

[27] G. Gavai *et al.*, "Supervised and unsupervised methods to detect insider threat from enterprise social and online activity data," *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl. (JoWUA)*, vol. 6, no. 4, 2015.

[28] D. C. Le and N. Zincir-Heywood, "Machine learning based insider threat modelling and detection," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management*, 2019.

[29] W. Meng, K. R. Choo, S. Furnell, A. V. Vasilakos, and C. W. Probst, "Towards bayesian-based trust management for insider attacks in healthcare

software-defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 2, pp. 761–773, Jun. 2018.

[30] S. Zhao, R. Wei, L. Cai, A. Yu, and D. Meng, "Ctlmd: Continuous-temporal lateral movement detection using graph embedding," in *International Conference on Information and Communications Security*. Springer, 2019, pp. 181–196.

[31] H. Bian, T. Bai, M. A. Salahuddin, N. Limam, A. Abou Daya, and R. Boutaba, "Uncovering lateral movement using authentication logs," *IEEE Transactions on Network and Service Management*, 2021.

[32] D. C. Le and N. Zincir-Heywood, "Exploring adversarial properties of insider threat detection," in *IEEE Conference on Communications and Network Security (CNS)*, 2020.

[33] ——, "Big data in network anomaly detection," in *Encyclopedia of Big Data Technologies,* , S. Sakr and A. Zomaya, Eds. Springer International Publishing, 2018, pp. 1–9.

[34] ——, "Exploring anomalous behaviour detection and classification for insider threat identification," *International Journal of Network Management*, Mar. 2020.

[35] D. C. Le, N. Zincir-Heywood, and M. I. Heywood, "Analyzing data granularity levels for insider threat detection using machine learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 30–44, 2020.

[36] M. I. Heywood, "Evolutionary model building under streaming data for classification tasks: opportunities and challenges," *Genetic Programming and Evolvable Machines*, vol. 16, no. 3, pp. 283–326, 2015.

[37] H. G. Kayacik, N. Zincir-Heywood, and M. I. Heywood, "On the capability of an som based intrusion detection system," in *International Joint Conference on Neural Networks*, Jul. 2003, pp. 1808–1813.

[38] C. C. Aggarwal, *Outlier Analysis*, 2nd ed. Springer Publishing, 2016.

[39] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Int. Conf. on Artificial Intelligence and Statistics*, 2011.

[40] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discov. Data*, vol. 6, no. 1, Mar. 2012.

[41] T. Pevný, "Loda: Lightweight on-line detector of anomalies," *Machine Learning*, vol. 102, pp. 275–304, 2016.

[42] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," *SIGMOD Rec.*, vol. 29, no. 2, p. 93–104, May 2000.

[43] G. O. Campos *et al.*, "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study," *Data mining and knowledge discovery*, vol. 30, no. 4, pp. 891–927, 2016.

[44] CERT and ExactData, LLC, "Insider Threat Test Dataset," https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099.

[45] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *IEEE SPW*, 2013.

[46] A. D. Kent, "Cybersecurity Data Sources for Dynamic Network Research," in *Dynamic Networks in Cybersecurity*. Imperial College Press, Jun. 2015. [Online]. Available: https://csr.lanl.gov/data/cyber1/

[47] A. Harilal *et al.*, "The wolf of sutd (twos): A dataset of malicious insider threat behavior based on a gamified competition." *JoWUA*, vol. 9, no. 1, pp. 54–85, 2018. [Online]. Available: https://github.com/ivan-homoliak-sutd/twos

[48] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, https://www.tensorflow.org/.

[49] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[50] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, 2011.

[51] Y. Zhao, Z. Nasrullah, and Z. Li, "Pyod: A python toolbox for scalable outlier detection," *Journal of Machine Learning Research*, vol. 20, no. 96, pp. 1–7, 2019.

[52] H. Debar and A. Wespi, "Aggregation and correlation of intrusion-detection alerts," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2001, pp. 85–103.

[53] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[54] D. C. Le and N. Zincir-Heywood, "A frontier: Dependable, reliable and secure machine learning for network/system management," *Journal of Network and Systems Management*, Jan. 2020.

[55] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *ACM Symposium on Information, Computer and Communications Security*, vol. 2006, 2006, pp. 16–25.

[56] M. Barandas, D. Folgado, L. Fernandes, S. Santos, M. Abreu, P. Bota, H. Liu, T. Schultz, and H. Gamboa, "Tsfel: Time series feature extraction library," *SoftwareX*, vol. 11, p. 100456, 2020.

[57] L. Liu, C. Chen, J. Zhang, O. De Vel, and Y. Xiang, "Insider threat identification using the simultaneous neural learning of multi-source logs," *IEEE Access*, vol. 7, pp. 183 162–183 176, 2019.

Duc C. Le is a Ph.D. student at Dalhousie University, Halifax, Canada. He received the Master degree in computer science from the same university in 2017, and the B. Eng. degree in electronics and telecommunications engineering from Posts and Telecommunications Institute of Technology, Ha Noi, Vietnam, in 2015. His research focuses on machine learning and its applications in computer and network security and analysis.

Nur Zincir-Heywood is a Full Professor of Computer Science with Dalhousie University, Canada. Her research interests include machine learning and data mining for networks, services and cybersecurity. She has published over 200 fully reviewed papers and has been a recipient of several best paper awards. She is an Associate Editor of the IEEE Transaction on Network and Service Management and the International Journal of Network Management. She has recently served as the Program Co-Chair and the General Co-Chair for the IEEE/IFIP International Conference on Network and Service Management. She is a member of the IEEE and the ACM and a recipient of the 2017 DNS Women Leaders in the Digital Economy Award.