Exploring Adversarial Properties of Insider Threat Detection

Duc C. Le and Nur Zincir-Heywood
Faculty of Computer Science, Dalhousie University
Halifax, NS, Canada
Email: lcd@dal.ca, zincir@cs.dal.ca

Abstract—Insider threat represents a major cybersecurity challenge to companies and government agencies. The challenges in insider threat detection include unbalanced data, limited ground truth, and possible user behaviour changes. This research presents an unsupervised machine learning (ML) based anomaly detection approach for insider threat detection. We employ two ML methods with different working principles, specifically autoencoder and isolation forest, and explore various representations of data with temporal information. Evaluation results show that the approach allows learning from unlabelled data under adversarial conditions for insider threat detection with a high detection and a low false positive rate. For example, 60% of malicious insiders are detected under 0.1% investigation budget. Furthermore, we explore the ability of the proposed approach to generalize for detecting unseen anomalous behaviours in different datasets, i.e. robustness. Comparisons with other work in the literature confirm the effectiveness of the proposed approach.

Index Terms—insider threat, anomaly detection, temporal information, data representation

I. INTRODUCTION

Insider threat is one of the most damaging security threats to the safety of data, systems, and intellectual property of companies and organizations. Malicious actions in insider threat are performed by authorized personnel 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, 55% of organizations and 42% of U.S. federal agencies faced insider threat attacks every year [2]. Furthermore, the type of attack has become more frequent recently [3].

Typical threats caused by malicious insiders are trade secrets / intellectual property theft, disclosure of classified information, theft of personal information, and IT system sabotage [4]. A major challenge in detecting insider threats come from the fact that malicious insider is authorized to access the organization’s systems and networks. In addition, data describing insider threat activities is typically rare and poorly documented [5]. Furthermore, storing, processing, and analyzing multiple sources of organizational data – from network traffic, web and file access logs, to email history – for identifying malicious insiders present another challenge in implementing detection solutions.

This work presents our proposed anomaly detection approach, where the focus is on employing unsupervised machine learning methods and different representations of data with temporal information for identifying signs of anomalous behaviours that may indicate insider threats (i.e. initial detection step). In doing so, the contributions are summarized as follows: (i) Different representations of data, namely concatenation, percentile and mean difference, are introduced for ML-based anomaly detection algorithms, where temporal information is encoded to highlight user behaviour changes; (ii) Capabilities and characteristics of two popular ML methods for anomaly detection – Autoencoder and Isolation Forest – are examined under different working conditions, in particular adversarial training, number of users in training data and the duration of training data; (iii) Comprehensive anomaly detection results are presented, per instance and per user; (iv) Case study analysis is performed; 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 presents the proposed anomaly detection approach. Section IV details the experiments and presents the evaluation results and comparisons. Finally, conclusions and future research are discussed in Section V.

II. RELATED WORK

Recently, insider threat detection and mitigation research has become increasingly important to organizations and cybersecurity firms. Different guides 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 [4], [6]. In [4], Collins et al. described 20 practices for organizations to prevent insider threats, as well as case studies of malpractices. Recent survey by Homoliak et al. [1] addresses the definition, taxonomy and categorization of insider threats, and provides an overview of the countermeasures.

Following the successes in ML applications for intrusion detection and anomaly detection tasks [7], [8], ML techniques have been applied in insider threat detection for the 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, including graph-based [9], mixture of models [10]–[12], Hidden Markov Model [13], [14], one-class-SVM [15], and deep learning-based autoencoders and
recurrent networks [16]–[18]. Many of them [10], [11], [18]–[22] are supported by insider threat prevention programs, such as DARPA’s project ADAMS, which aims to identify patterns and anomalies in very large datasets to combat insider threats. 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 [16], evolutionary computation [23], [24] and supervised learning [21], [25] are some examples.

Some works in the literature report the importance of temporal information in dealing with insider threat, which is highly related to the human factors [10], [13], [16], [17], [26], [27]. Notable attempts to leverage temporal information include moving average approach [16], [26], 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-B). This shows advantages over other approaches that build one / many models per user [13], [16], [26] and employ not as scalable learning algorithms [15]–[17].

**III. METHODOLOGY**

![Fig. 1. Workflow of the proposed anomaly detection system](image)

An overview of the proposed insider threat detection system is presented in Figure 1. From raw collected log data of user activities, the numerical features are first extracted 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 via unsupervised learning. The employed ML methods are presented in III-B. Post training, an anomaly score is assigned by the detection model to each data example. Additionally, based on a selected investigation budget, a decision threshold can be calculated for further investigation of data samples with high anomaly scores, i.e. exceeding the threshold.

In this work, the investigation budget (IB) is defined as the amount (%) of data that the security analyst can examine for confirmation of malicious behaviours [8], [28]. This represents the available human resources for analyzing the anomaly alerts and performing the necessary actions in response. Post-training, assuming an IB of $\alpha\%$, the analyst is presented with $\alpha\%$ of the data instances with the highest anomaly scores for investigation.

**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 daily or weekly. We select those time periods for data aggregation to summarize a complete view of users' activities over a day/week into each data instance [29]. 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 [29]. However, that may not be beneficial. By lowering the amount of extracted instances, day/week data reduce the required workload to inspect anomaly alerts in the unsupervised anomaly detection setting, where false alerts are unavoidable [8].

Numerical features are then extracted to represent each day or week of a user's activities. Two 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. 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 [29].

1) **Temporal information in data representation**: Given that malicious insiders are essentially regular employees before they start performing malicious actions [4], we propose data representation approaches using temporal information. The goal is to highlight the changes from previously recorded activities of user behaviour.

   a) **Concatenation**: Inspired by the use of shift register and taps for representing time in data for intrusion detection [30], we introduce data examples to anomaly detection algorithms as concatenation of $c$ consecutive data instances of the same user. The idea is to encourage the learning algorithms to construct comparisons/arithmetic operations between each user data instance and its previous records, which may show anomalous changes in behaviours. In this data representation form, a data instance $x_t$ at time $t$ is adjoined with $c - 1$ most recent instances to form a data point for anomaly detection:

   $$x_{t}^{concatenation} = \text{concat}(x_t, x_{t-1}, x_{t-2}, \ldots, x_{t-c+1})$$  

   (1)

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

   b) **Percentile and Mean 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:

**Algorithm 1:** Calculating Percentile and Mean difference representation of data

**Input:** $x_t$ of user $u$, window size $w$

**Output:** $x_t^{output}$

construct a $n \times F$ matrix $X$ of $x_{t-1}, x_{t-2}, \ldots, x_{t-n}$ of the same user $u$, based on $w$; 
\[ // F: \text{# features} \]
\[ x_t^{output} = [ ]; \]

for feature $f$ in $F$ do
\[
\begin{aligned}
  &\text{if percentile} \quad \\
  &\quad f' = \text{findPercentile}(x_t[f], X[:, f]); \\
  &\text{else} \quad \text{mean difference} \\
  &\quad f' = x_t[f] - \bar{E}(X[:, f]); \\
  &x_t^{output}.append(f'); \\
\end{aligned}
\]

Following the algorithm, each arriving data instance is compared with previous data instances of the same user in a time window $w$ to create percentile or mean 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, taking both weekdays and weekends into account.

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 [31], this may be more effective than normalizing all data instances of each user from the beginning.

**B. Unsupervised Machine Learning for Anomaly Detection**

This work employs two popular ML methods for anomaly detection: Autoencoder (AE) and Isolation Forest (IF).

1) **Autoencoder (AE):** AE is a form of multi-layer neural network that compresses and reconstructs the data. Figure 2 depicts an example of an AE with three hidden layers. The input and output layers both have $d$ neurons ($d$: the dimensionality of the data). 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:

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

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 [32], 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 [33] and sigmoid activation functions, respectively.

2) **Isolation Forest (IF):** Differs 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, IF [34] works on the principles that anomaly examples are rare and significantly different in attribute-values from normal data points. IF 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. Each tree in IF works on a subset of training data. 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 $h(x) = \bar{E}(h_i(x))$ – 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 [35]. These characteristics are evaluated in IV-C.

**IV. EXPERIMENTS AND RESULTS**

In this section, we present the evaluations and results using the proposed approach for insider threat detection on the CERT insider threat datasets [36], [37]. Specifically, sub-section IV-A summarizes the datasets and data pre-processing methods (with temporal representation). Experiment settings, quantitative results, comparisons, and further discussions are presented in sub-sections IV-B, IV-C, and IV-D, respectively.

In this work, the insider threat detection performances are 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 (different investigation budgets), and AUC (Area Under the Curve) summarizes ROC in a single numerical metric for comparison between models.

\[
DR = \frac{TruePositive}{TruePositive + FalseNegative} \tag{3}
\]

\[
FPR = \frac{FalsePositive}{FalsePositive + TrueNegative} \tag{4}
\]

We also present DRs and FPRs at critical IBs (see §III) for better understanding of the performances 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 [29], [38]. Specifically, a normal insider (user) is classified as malicious 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).

A. Datasets

The CERT insider threat datasets are publicly available for development and testing of insider threat mitigation approaches [36], [37]. 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. Due to the space limitation, readers are referred to [36], [37] for details regarding the threat scenarios.

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-A1. Specifically, for concatenation representation, \(c = 2\) or \(c = 3\) most recent data instances of each user are joined (abbreviated as \(C2\) and \(C3\)). In percentile and mean difference representations, the window size value \(w\) is set to 7 days and 30 days for daily data, and 30 days and 60 days for weekly data (4 weeks and 8 weeks). Thus, the data representations are abbreviated as \(P7, P30, P60\) for percentile, and \(M7, M30, M60\) for mean difference representations, respectively. 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 [36]. Table I shows the statistics of the employed data in each type and the number of normal and malicious users.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Feature Count</th>
<th>Class Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>CERT R4.2</td>
<td>day 500</td>
<td>Normal 85, Sc 1 861</td>
</tr>
<tr>
<td></td>
<td>week 661</td>
<td>Sc 2 254, Sc 3 10</td>
</tr>
<tr>
<td>CERT R6.2</td>
<td>day 888</td>
<td>Normal 3, Sc 1 20</td>
</tr>
<tr>
<td></td>
<td>week 1176</td>
<td>Sc 2 4, Sc 3 7</td>
</tr>
</tbody>
</table>

B. Experiment Settings and ML Training

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 data of randomly selected \(n_u = 200\) users in the first \(n_w = 37\) weeks (50% of dataset duration) for training. 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.

We implemented the data pre-processing and analysis steps using Python 3. AE s are implemented using Keras [39] with Tensorflow back-end [40]. 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 [41] for 100 epoch each. Implementation from Scikit-learn [42] is used for IF. With insights from [34], the number of trees in IF is set to 200.

C. Anomaly Detection Results

Instance-based anomaly detection results with different investigation budgets are presented in Table II, while Table III shows user-based results. Figure 3 shows instance-based and user-based ROCs in the case of R4.2 week data with different 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). 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 (Table III).
### Table II

**Instance-based anomaly detection results with different investigation budgets (IB). The unit of DR is percent (%)**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Temp. rep.</th>
<th>DR @ 0.1% IB</th>
<th>DR @ 1% IB</th>
<th>DR @ 5% IB</th>
<th>DR @ 10% IB</th>
<th>DR @ 20% IB</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cert R4.2 day</td>
<td>Org.</td>
<td>2.45</td>
<td>8.50</td>
<td>1.60</td>
<td>26.20</td>
<td>13.72</td>
<td>47.41</td>
</tr>
<tr>
<td></td>
<td>C2</td>
<td>2.68</td>
<td>9.34</td>
<td>0.84</td>
<td>24.68</td>
<td>11.89</td>
<td>46.07</td>
</tr>
<tr>
<td></td>
<td>C3</td>
<td>2.23</td>
<td>8.81</td>
<td>0.45</td>
<td>21.92</td>
<td>11.39</td>
<td>43.37</td>
</tr>
<tr>
<td></td>
<td>P7</td>
<td>7.85</td>
<td>3.59</td>
<td>0.45</td>
<td>21.92</td>
<td>11.39</td>
<td>43.37</td>
</tr>
<tr>
<td></td>
<td>P30</td>
<td>2.31</td>
<td>12.08</td>
<td>3.66</td>
<td>36.15</td>
<td>23.82</td>
<td>62.39</td>
</tr>
<tr>
<td></td>
<td>M7</td>
<td>3.48</td>
<td>10.57</td>
<td>3.44</td>
<td>20.86</td>
<td>15.97</td>
<td>33.55</td>
</tr>
<tr>
<td></td>
<td>P30</td>
<td>2.31</td>
<td>12.08</td>
<td>3.66</td>
<td>36.15</td>
<td>23.82</td>
<td>62.39</td>
</tr>
<tr>
<td></td>
<td>M30</td>
<td>3.41</td>
<td>10.82</td>
<td>3.31</td>
<td>21.15</td>
<td>10.90</td>
<td>40.13</td>
</tr>
</tbody>
</table>

| Certification R6.2 week | Org. | 2.53 | 9.97 | 0.85 | 22.41 | 6.71 | 40.13 | AE, AUC = 0.847 |
| | C2 | 2.88 | 9.21 | 0.66 | 22.15 | 4.78 | 41.65 | AE, AUC = 0.858 |
| | C3 | 2.75 | 7.12 | 0.66 | 20.92 | 3.64 | 40.06 | AE, AUC = 0.857 |
| | P30 | 5.76 | 10.57 | 0.44 | 27.63 | 13.23 | 52.18 | AE, AUC = 0.874 |
| | M7 | 3.41 | 10.82 | 3.31 | 21.15 | 10.90 | 40.13 | AE, AUC = 0.797 |
| | P30 | 5.76 | 10.57 | 0.44 | 27.63 | 13.23 | 52.18 | AE, AUC = 0.874 |
| | M30 | 3.41 | 10.82 | 3.31 | 21.15 | 10.90 | 40.13 | AE, AUC = 0.797 |

![ROC plots](image)

On the learning algorithms, it is clear that Autoencoder is better than Isolation Forest for detecting anomalies representing insider threats, especially at very low FPRs. Figure 4 shows the difference in the case of original R6.2 week data. The AUCs and detection rates at almost all investigation budgets achieved by AE are greater than IF (Table II). For example, at only 0.1% IB, AE is able to detect 60% of the malicious insiders from R6.2 week data with P30 representation, while IF requires 8% IB to reach a similar UDR in the same setting.

On data representations, Table II, III and Figure 3 show that percentile (P7, P30, P60) is the best representation of data for anomaly detection. It allows both algorithms to significantly...
outperform the original data representation. Concatenation or mean difference on the other hand are unable to show improvements over results achieved using the original representation. In some cases, such as R4.2 day data, mean difference even deteriorates the AUC. The observations suggest that percentile difference representation (M7, M30, M60) seems to create noises and decreases the detection performance (Figure 3).

Finally, on concatenated representation, the results with largely no improvements over original representation show that it is no longer able to capture changes in the data. At the same time, maintaining the absolute values of changes as in mean difference representation (M7, M30, M60) seems to create noises and decreases the detection performance (Figure 3). Finally, on concatenated representation, the results with largely no improvements over original representation show that it is hard to facilitate meaningful automatic comparisons between data of different points in time.

On day and week data types, it appears that UAUCs achieved on day data is better than on week data (Table III). This comes at a cost of higher amount of alerts, as day data is about five folds of week data (Table I). Specifically, 1% IB on day data generates an equivalent amount of instance alerts as 5% IB on week data. In the following, we assume P30 data representation and analyze ML algorithms on CERT R4.2 data types under different sizes of training data and adversarial conditions.

### TABLE III

User-based anomaly detection results with different investigation budgets. The unit of UDR and UFPR is percent (%).

<table>
<thead>
<tr>
<th>Data type</th>
<th>Temp. rep.</th>
<th>0.1% IB</th>
<th>1% IB</th>
<th>5% IB</th>
<th>10% IB</th>
<th>UAUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CERT R4.2</td>
<td>Day</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Org.</td>
<td>2.56</td>
<td>24.43</td>
<td>2.52</td>
<td>1.43</td>
<td>16.48</td>
<td>58.29</td>
</tr>
<tr>
<td>C2</td>
<td>1.71</td>
<td>21.43</td>
<td>1.41</td>
<td>0.43</td>
<td>12.09</td>
<td>54.71</td>
</tr>
<tr>
<td>C3</td>
<td>1.33</td>
<td>17.71</td>
<td>1.27</td>
<td>0.29</td>
<td>10.27</td>
<td>51.86</td>
</tr>
<tr>
<td>P7</td>
<td>2.52</td>
<td>21.57</td>
<td>3.97</td>
<td>6.86</td>
<td>11.70</td>
<td>96.57</td>
</tr>
<tr>
<td>M7</td>
<td>3.27</td>
<td>32.57</td>
<td>6.30</td>
<td>7.00</td>
<td>18.75</td>
<td>70.29</td>
</tr>
<tr>
<td>P30</td>
<td>1.89</td>
<td>21.14</td>
<td>2.27</td>
<td>6.43</td>
<td>11.96</td>
<td>73.73</td>
</tr>
<tr>
<td>M50</td>
<td>2.06</td>
<td>25.71</td>
<td>4.65</td>
<td>4.43</td>
<td>19.41</td>
<td>69.57</td>
</tr>
<tr>
<td>CERT R4.2</td>
<td>Week</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Org.</td>
<td>0.63</td>
<td>10.57</td>
<td>0.47</td>
<td>0.14</td>
<td>7.43</td>
<td>36.29</td>
</tr>
<tr>
<td>C2</td>
<td>0.48</td>
<td>9.86</td>
<td>0.44</td>
<td>0.14</td>
<td>4.60</td>
<td>32.43</td>
</tr>
<tr>
<td>C3</td>
<td>0.40</td>
<td>9.29</td>
<td>0.40</td>
<td>0</td>
<td>3.79</td>
<td>26.29</td>
</tr>
<tr>
<td>P30</td>
<td>0.53</td>
<td>19.86</td>
<td>1.33</td>
<td>1.00</td>
<td>5.00</td>
<td>35.57</td>
</tr>
<tr>
<td>M30</td>
<td>0.63</td>
<td>19.00</td>
<td>1.44</td>
<td>1.14</td>
<td>13.32</td>
<td>50.43</td>
</tr>
<tr>
<td>P60</td>
<td>0.35</td>
<td>27.14</td>
<td>1.17</td>
<td>0.71</td>
<td>4.43</td>
<td>46.14</td>
</tr>
<tr>
<td>M60</td>
<td>0.51</td>
<td>24.86</td>
<td>1.56</td>
<td>1.29</td>
<td>14.51</td>
<td>50.43</td>
</tr>
</tbody>
</table>

Fig. 5. UAUC by number of malicious users in R4.2 training data.

1) Anomaly detection performance under adversarial 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). This is to analyze how the anomaly detection approaches respond to adversarial conditions, where malicious
data is presented at high density in training data (i.e. data poisoning), which may corrupt the ML models into mislabelling malicious as normal [43], [44].

Figure 5 shows the user-based AUC by AE and IF on R4.2 data types under different number of malicious users in training. Overall, it is clear that IF is very robust to the data poisoning attack, with AUC even increasing slightly with the presence of malicious data in training. This can be explained through the properties of IF, where small amount of contamination in training data allows trained IF trees to better model the anomalies that may appear in the data [35]. On the other hand, AE’s performance 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. Nevertheless, AE was able to maintain a better performance than IF on R4.2 day data, and on week data with up to 40 malicious users in training data (20%). We note that in practice, the amount of malicious users in training data for insider threat detection approaches is typically very small [5], hence the use of AE is still preferred.

On the other hand, AE’s performance deteriorates as the number 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. Nevertheless, AE was able to maintain a better performance than IF on R4.2 day data, and on week data with up to 40 malicious users in training data (20%). We note that in practice, the amount of malicious users in training data for insider threat detection approaches is typically very small [5], hence the use of AE is still preferred.

2) Effects of the number of users in 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 maximum 1000 users in CERT R4.2. User-based AUCs are presented in Figure 6. Results show that while IF’s performances are largely unchanged, AE’s UAUC increases slightly to 200 users in training data before decreasing slowly with more training users. This can be explained through results in IV-C1, where a larger number of training users creates a higher chance of malicious users to be included in training data, hence lowering AE’s performance. This shows that maintaining a relatively small number of users in training data not only reduces the computational cost but also potentially gives better results.

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%). Figure 7 shows user-based AUCs of AE and IF on CERT R4.2 data types. As in the previous experiments, IF’s performances are maintained through different number of weeks used in training data. On the other hand, AE’s performances rise until about 50% of the data duration is used in training (37 weeks), then remains stable. In fact, more malicious insider activities appear in the second half of CERT data than in the first half [36]. Hence, it can be concluded that for AE, 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-C1).

D. Discussions and Comparisons

In this part, CERT R6.2 is employed for testing purposes. 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.

1) 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 alarm example on CERT R6.2 is associated with actions of user PLJ1771 – an IT administrator – on August 12, 2010. Using AE and P30 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, he/she logs onto PC-3999, which belongs to his supervisor – HIS1706, and starts keylogging on the PC. This corresponds to the behaviours of a “disgruntled system administrator” in the CERT dataset [36].

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. Given the link between anomaly alarms and the corresponding action sequence as presented, a security analyst can easily examine the system alerts, identify and respond to the malicious actions, if any.
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, their anomalous nature needs to be inspected to ensure the safety of the system and data.

2) Detection performances on insider threat scenarios: As mentioned in IV-A, there are 5 malicious insiders in CERT R6.2, each depicts a unique threat scenario. Table IV presents the malicious insiders and detection results using AE and week data with P60 representation of CERT R6.2. 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.

On the other hand, threat scenarios 2 and 5 are much harder to detect, resulting in FPRs of 3.07% and 8.36%, respectively. The descriptions of these scenarios show much less intrusive malicious behaviours than the other three scenarios [36]. 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 performance on these two scenarios.

3) Robustness of the trained models: For this analysis, we use an anomaly detection model trained on one CERT dataset (R4.2) for detecting anomalies on another one (R6.2). As CERT R6.2 is a newer version with changed generative models and much larger size [36], 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 P30 data representations are shown in Figure 8. The figure shows that anomaly detection model trained using CERT R4.2 data with P30 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 ability of the proposed system to generalize (robustness) well when percentile data representation is used. This 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. 8. UAUC of models trained on CERT R4.2 and R6.2 data when tested on R6.2](image)

4) 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 [13]–[18], [22]. On CERT R4.2, our proposed approach obtained AUC of 0.887 and 0.903 on week and day data, outperforming previous works [13]–[15] that used HMM and one-class-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. [22], 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 IV-D2). Under the same malicious user consideration, our approach posts an AUC of 0.9996. 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 IV-D3.

V. CONCLUSIONS AND FUTURE WORK

In this research, an unsupervised ML based anomaly detection approach for insider threat detection is presented. To this end, Autoencoder and Isolation Forest are used as the ML methods for anomaly detection. Furthermore, both methods are studied using different representations of data with temporal information. These representations include concatenation, percentile and mean difference. In doing so, the aim is to describe the changes in user activities that could highlight 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. This combination enables effective insider...
threat detection under very low investigation budgets. On the other hand, experiments demonstrate the robustness of Isolation Forest, which may suggest its use under extremely adversarial conditions. 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, other data representations and data availability for anomaly detection. Furthermore, informed attackers’ actions can also be introduced to further examine the performance under more adverse conditions.

ACKNOWLEDGMENT

This research was 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 supports of the Killam Trusts and the province of Nova Scotia.

REFERENCES