Hindawi
Security and Communication Networks
Volume 2021, Article ID 4148441, 11 pages
https://doi.org/10.1155/2021/4148441

*Research Article*

# Detecting Insider Threat from Behavioral Logs Based on Ensemble and Self-Supervised Learning

**Chunrui Zhang**,[1,2] **Shen Wang**,[1] **Dechen Zhan**,[1] **Tingyue Yu**,[1] **Tiangang Wang**,[1] **and Mingyong Yin**[2]

[1]*Faculty of Computing, Harbin Institute of Technology, Harbin 150001, China*
[2]*Institute of Computer Application, China Academy of Engineering Physics, Mianyang 621900, China*

Correspondence should be addressed to Shen Wang; shen.wang@hit.edu.cn

Received 19 April 2021; Revised 29 September 2021; Accepted 5 November 2021; Published 26 November 2021

Academic Editor: Jinwei Wang



Recent studies have highlighted that insider threats are more destructive than external network threats. Despite many research studies on this, the spatial heterogeneity and sample imbalance of input features still limit the effectiveness of existing machine learning-based detection methods. To solve this problem, we proposed a supervised insider threat detection method based on ensemble learning and self-supervised learning. Moreover, we propose an entity representation method based on TF-IDF to improve the detection effect. Experimental results show that the proposed method can effectively detect malicious sessions in CERT4.2 and CERT6.2 datasets, where the AUCs are 99.2% and 95.3% in the best case.

## 1. Introduction

With the rapid development of information security, threats from inside of systems, which are called insider threats, have received more and more attention. Compared with threats from external systems, insider threat can be more harmful to companies and military organizations, as insider employees may take advantage of the loopholes in system implementation and business process to conduct behaviors that threaten system security with ease, including confidential information theft, commercial fraud, or system destruction [1]. Compared with the external attackers who need to find ways to gain access and avoid detection, attacks launched by insiders are often more difficult to defend.

As one of the important defend solutions to insider threat, insider threat detection plays a very important role in recent research studies. In most of the works, insider threat detection is regarded as an anomaly detection problem, in which we try to extract the behavior features of different users from the system log records and then distinguish between the behaviors of normal users and malicious users. Traditional insider threat detection methods are mostly based on user behavior features extracted artificially; thus, their effects often depend on the effectiveness of the extracted features. Besides, these methods are difficult to model user behaviors in long time series. These factors limit the detection effect of insider threats. Therefore, part of the recent works has turned to temporal models, especially those based on deep learning, for insider threat detection [2, 3]. They model user's behavior in a period and get the feature representation of behavior sequences automatically. On this basis, outlier analysis is carried out to identify malicious insiders.

However, based on our observation, existing insider threat detection methods based on deep learning still have some problems that need to be concerned. First of all, the inputs of deep learning models usually focus on the image, video, and natural language tasks, in which most of their inputs have spatial uniformity and continuity. It means that, for these tasks, it is possible to learn a feature representation that is invariant to small input transformations (such as shifting, clipping, and noise) [4]. However, the records in most dimensions of user logs are discrete and unbalanced,

which makes it difficult to learn effective feature representation for user behavior. On the other hand, in log datasets that most researchers focus on [5], the ratio between the number of legitimate users and malicious users is usually very large. It is also a difficult problem to detect abnormalities in datasets with extremely unbalanced positive and negative samples.

To solve existing problems, in this paper, we attempt to establish a framework to distinguish the legitimate behaviors and malicious behaviors of the specific user directly, instead of modeling and representing behavior sequences before anomaly detection. We make the following contributions:

(i) To make the features extracted from user behaviors contain more effective information, we propose an entity embedding method based on TF-IDF, which represents operated entities according to their frequency in different sessions

(ii) We design an insider threat detection method based on ensemble learning, in which we propose a new sampling strategy called Over-Bootstrap, which can effectively alleviate the overfitting caused by sample imbalance

(iii) Based on the detector, we introduce a self-supervised pretext task to identify which user the input behavior sequence belongs to, so as to learn distinctive behavior representation for different users

The experimental results show that the proposed method can detect insider threats effectively, as the AUC of the proposed method is up to 99.2% in the best case in CERT4.2 and up to 95.3% in the best case in CERT6.2, which are both higher than competing methods.

## 2. Related Work

Some of the early works of insider threat detection aim at user command records. They take the user's command sequence in the UNIX system as the analysis object and calculate the probability of occurrence of adjacent command patterns and the matching degree between new commands and historical records, respectively, to identify malicious behaviors [6, 7]. In later works, system audit logs are often used as the analysis object of insider threat detection. For example, Patcha and Park [8] attempt to detect insider threats in the system audit log and attribute the difficulty of insider threat detection to the complex data relationship, the difficulty of attack modeling, and the dynamic change of user behaviors. Recently, more and more works draw their attention to CERT [5], which is a synthetic insider threat dataset using scenarios containing traitor instances and masquerade activities.

In terms of detection methods, machine learning algorithms are introduced in some works, such as the naive Bayes method [9], EM [10], and SVM [11], to recognize the abnormal patterns of user behavior features extracted manually. Lippmann and Cunningham [12] propose an insider threat detection system using keyword selection and a neural network. They calculate the statistics of the number of keywords from the predetermined list to form the input of the neural network and obtain the estimation of the posterior probability of attacks. Eldardiry [13] et al. use the TF/IDF framework to merge users' behaviors in different data domains and get consistent scores, based on which to propose an internal threat detection system. However, their model cannot take into account the differences of different user behavior patterns, which affects the effectiveness of data fusion.

In later works, deep learning models and some other representation learning techniques have been used to extract user behavior features automatically. Rashid et al. [14] propose an insider threat detection method using hidden Markov. They use the hidden Markov model (HMM) to simulate the normal behavior of users and regard the behavior deviating degree from the normal behaviors as abnormal behaviors. However, the time complexity of this method is very high as the computational cost of the hidden Markov model increases with the increase of the number of states. Veeramachanani [15] gathered numerical features in a series of time windows and used autoencoders as a representation learning model. He fed these representations to a series of detectors to calculate the abnormal scores. But this kind of coarse-grained modeling method will lose a lot of information related to abnormal behaviors.

In general, these methods are still difficult to model the temporal relationship between different behaviors of a specific user. Therefore, in recent works, insider threat detection methods based on the deep Recurrent Neural Network and its variants were proposed [2, 3]. Most of the existing insider threat detection methods based on RNN include three steps: behavior modeling, feature representation, and anomaly detection. After aggregating user behaviors or user's behavior sequences from different log files, they obtain the feature representation of user behavior samples by training a DNN-based unsupervised generation model, to approximate the distribution of legitimate user behavior.

In [2], they used two different schemes for feature representation and anomaly detection. In the first scheme, the user behavior of the previous moment is used as the input to predict the user behavior of the next moment. The distance between the predicted behavior and the observed behavior is used to calculate the abnormal score of specific user behavior. In another scheme, they use an autoencoder to reconstruct the current sample. At the same time, they modify the training loss, as the positive samples should learn a lower reconstruction error, and the negative samples should learn a higher reconstruction error. In the anomaly detection phase, the reconstruction error of input is used to measure an anomaly score.

In [3], LSTM is introduced to model the dependency relationship on a longer time scale. The behavior of a specific user in one day is aggregated into a single input sample. For one user, the behavior sequence within several days is input into LSTM, and the input sequence is reconstructed by an autoencoder to learn its representation. In the anomaly detection stage, a binary CNN is trained on the feature representation learned by the autoencoder to identify whether an input sample is malicious behavior.

The work of Sharma et al. [16] is similar to the above two methods, while it uses a flexible time window to group data points, so as to improve the performance of the detector. In addition, the work of Le and Zincir-Heywood [17] ensembles different detection models to improve the generalization ability and the effectiveness of the detector. In summary, compared with traditional detecting methods, these methods based on deep learning can extract features from high-dimensional data through a deep neural network, which solves the disadvantage of traditional methods that need to extract features manually. In addition, the use of Recurrent Neural Network can model the temporal characteristics, which improves the detection effect of persistent internal threats. However, they do not consider the representation of entities but simply classify user behaviors and represent them through one-hot coding, which loses a lot of information in user behaviors, affecting the detection effect of internal threats. For example, Tuor et al. [2] combine user attribute features such as user roles, departments, and supervisors in the organization and accumulate counts of 408 activities a user has performed over some fixed time window. Yuan et al. [3] enumerate 64 possible user operations and uses represented by a one-hot vector and combine the user's behavior sequence in a certain period of time into a matrix as the unit of threat detection.

Orthogonal to our work, there have been many recent works that do not focus on the traditional user behavior logs, but on internal threat detection in favor of other scenarios, including network data flow [18], social contact [19], specific single behavior record [20], and trust management in the Internet of Things (IoT) [21]. For example, in order to prevent attackers from disrupting the normal operation of an IoT system, anomaly detection technology is needed to help monitor the network traffic of the system in real time [22, 23].

## 3. Methodology

*3.1. Overview.* The effectiveness of the existing insider threat detection methods based on deep learning depends on the feature representation of user behavior, that is, the probability distribution of the learned legitimate user behavior. This feature representation is obtained by reconstructing the user behavior itself [24] or predicting it the next time [25]. However, we should note that some problems are difficult to solve when directly transplanting these representation learning methods from image, video, or voice to user behavior.

The first problem is how to effectively measure the distance between the output sample and the target sample. Common deep learning tasks focus on image, video, or voice data sets, whose most basic characteristics are spatial uniformity and continuity. This makes it possible to measure the reconstruction error directly by MSE (Euclidean distance between the output sample and the target sample). But in user behavior samples, the semantic information contained in different data dimensions is completely inconsistent. In this case, the validity of using autoencoder to model the distribution of legitimate data is questionable.

Consider the following question: can we effectively detect insider threats by modeling legitimate data and outlier analysis without considering abnormal data? In our view, the answer may be No. The reason lies not only in the difficulty of distance measurement but also in the different behavior patterns of different users. Even if a specific user's behavior seems to be far away from the normal behavior of most users (such as visiting a website that other users have never visited, or logging into the company's computer during off hours), judging whether the behavior is malicious should depend on the definition of abnormal behavior and its comparison with the normal behavior of the user.

Another problem we need to pay attention to is the extreme imbalance between legitimate user behavior data and malicious user behavior data, which means in most cases the number of legitimate users is far more than malicious users. Secondly, for a malicious user, the number of normal behavior sequences is far more than the number of abnormal behavior sequences. As a binary classification problem, when the proportion of one kind of data is much higher than that of the other, the model will produce serious overfitting to the more data. This will have a serious impact on the performance of the detector.

Through these analyses, we conclude that the proposed method should meet the following two requirements. First of all, we need to consider how to alleviate the overfitting problem caused by data imbalance and ensure the performance of the detector under the premise of very few malicious samples. Secondly, we should not rely on unsupervised methods, like data reconstruction or prediction, to extract user behaviors representation but consider how to obtain more distinguishable representation for legitimate samples and malicious samples.

Our work focuses on the CERT4.2 and CERT6.2 datasets [5], which consists of event log records simulating an organization's computer system, including five event sources: login/logoff activities, e-mail traffic, HTTP traffic, removable media device, and file operations. User behaviors from each event source are stored in different log files. We take a session, which is a series of behaviors between one pair of login and logout, as the basic unit to detect insider threat. The overall framework of the proposed method is shown in Figure 1.

(i) We first need to organize the user behavior sequences in the five logs according to the user IDs. Then, the behaviors of every user are sorted chronologically. After these preparations, we first need to extract the features of user behaviors contained in all sessions and transform the sequences of behaviors into feature sequences with the same length. This part will be discussed in Section 3.2

(ii) After obtaining the feature sequence, we use an LSTM-based detector to detect malicious sessions and propose an ensemble strategy to avoid the overfitting problem in the process of training. This part will be discussed in Section 3.3

(iii) For learning distinctive feature representation for different users, to detect malicious behavior

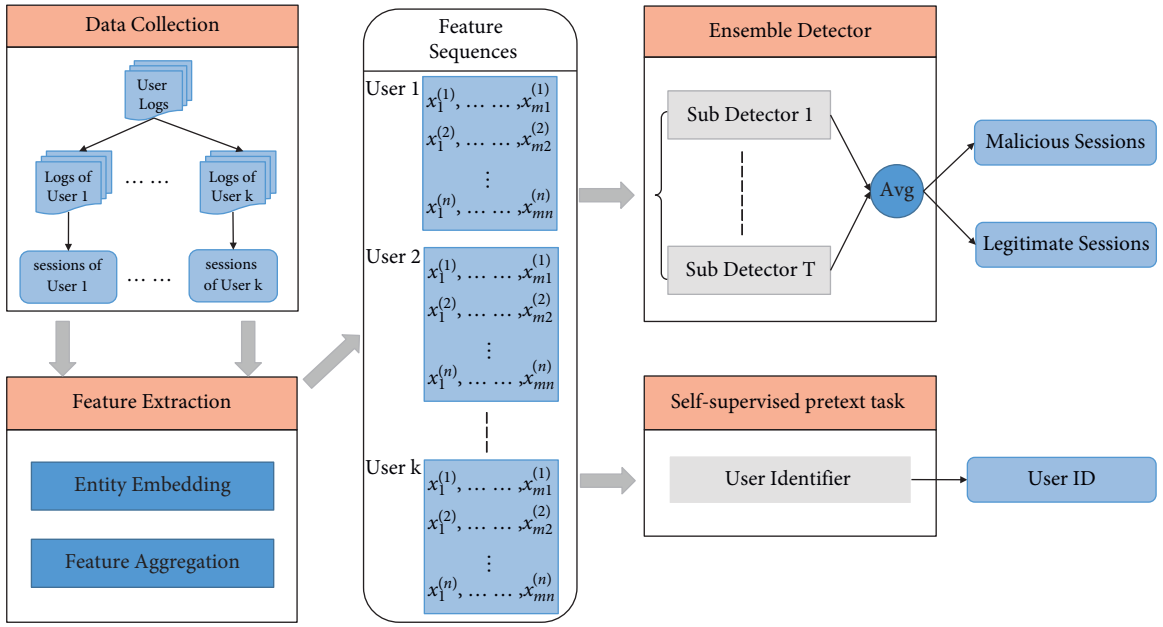


Figure 1: The overall framework of the proposed method.

sequences more effectively, we design a self-supervised pretext task to obtain different behavior patterns among different users. This part will be discussed in Section 3.4

*3.2. Entity Embedding Based on TF-IDF.* At first, to transform the user behavior sequences into a mathematical form that can be understood by detectors, we need to extract the features of user behaviors. As the log records from different sources contain completely different attributes, our first task is to aggregate log records from different sources into a single unified feature. Most of the existing works do not consider the specific operated entities like e-mail addresses and URLs, which lose most of the information for abnormalities. Its main reason is that one-hot coding with finite dimensions is impossible to represent the entities with theoretically infinite space. Therefore, we introduce the TF-IDF metric [26] to score the importance of each entity in different event sources and take the importance score to be its feature.

For an operated entity in a session, its importance in the session increases with the number of times it appears. But at the same time, its importance decreases inversely with its presence in other sessions. For example, a common web page http://www.google.com may be accessed multiple times in a session. But other users are likely to visit it many times, which weakens the amount of information brought by this behavior, while for a URL that is rarely visited by most users, such as http://www.wikileaks.info, its access by a specific user may bring some anomalous signal.

TF stands for Term Frequency, which indicates the frequency of an entity in a specific session; IDF means Inverse Document Frequency, which is the negative logarithm of the proportion of sessions containing the entity in all sessions. TF-IDF score of a specific entity in a specific session is formally defined as follows.

Let the set of all sessions be $S$, $s \in S$. Take $E$ as a specific event source, $RE$ is the set of log records corresponding to $E$, $e^{(i)} \in E$, and $r_j \in RE$. $\mathrm{Ref}(e^{(i)}, r_j)$ is a function that represents whether the entity $e^{(i)}$ appears in the log record $r_j$:

$$\mathrm{Ref}\left(e^{(i)}, r_j\right) = \begin{cases} 1, & \text{where } r_j \text{ refers to } e^{(i)}, \\ 0, & \text{where } r_j \text{ don't refers to } e^{(i)}. \end{cases} \tag{1}$$

For a pair of entity $e^{(i)}$ and session $s$, we define the TF-IDF score as $p_{\mathrm{TI}}(e^{(i)}, s)$:

$$\begin{cases} p_{\mathrm{TI}}\left(e^{(i)}, s\right) = \mathrm{TF}\left(e^{(i)}, s\right) \times \mathrm{IDF}\left(e^{(i)}\right), \\ \mathrm{TF}\left(e^{(i)}, s\right) = \dfrac{\sum_{r_j \in s_k} \mathrm{Ref}\left(e^{(i)}, r_j\right)}{|s|}, \\ \mathrm{IDF}\left(e^{(i)}\right) = -\log \dfrac{|R|}{\left\{s \middle| \sum_{r_j \in S} \mathrm{Ref}\left(e^{(i)}, r_j\right) > 0\right\}}. \end{cases} \tag{2}$$

To ensure that an entity has the same representation in all sessions, we use its average TF-IDF score in all sessions as its feature representation:

$$P_{\mathrm{avg-TI}}\left(e^{(i)}\right) = \frac{\sum_{s \in S} p_{\mathrm{TI}}\left(e^{(i)}, s\right)}{\left\{s \middle| \sum_{r_j \in S} \mathrm{Ref}\left(e^{(i)}, r_j\right) > 0\right\}}. \tag{3}$$

The feature representation of an entity $e^{(i)}$ is positively correlated with its self-information. The higher the average TF-IDF of the entity$e^{(i)}$, the higher probability of an abnormal situation it indicates. For every entity $e^{(i)}$ in source $E$, its avg-TI is normalized to (0, 1).

After the entity feature extraction, we aggregate the features of each user record and express the log records from different event sources in a 10-dimensional vector. The characteristic of each dimension is shown in Table 1.

*3.3. Ensemble Detector.* We try to train a binary classifier to judge whether the input session is malicious or not. As there are many operation activities in a session, these activities are likely to have a long-distance temporal dependence on each other. Not only cannot Convolutional Neural Networks [4] establish this kind of temporal dependence, but even Recurrent Neural Networks [27] cannot learn long-distance dependence. Therefore, we use LSTM [28] model to construct the detector. For a specific session $s$, of which the behavior records have been represented according to the method described in 3.2 as $\hat{s} = (x_1, \ldots, x_m)$, we input these feature vectors into the detector in turn, of which the architecture is shown in Figure 2.

However, as analyzed above, the proportion of legitimate sessions and malicious sessions in the training set is extremely unbalanced, which will make the learned data distribution seriously tilt to the legitimate behaviors. This will bring serious overfitting problems to the model, which makes the model unable to correctly distinguish normal behaviors and malicious behaviors. Therefore, we propose an ensemble learning strategy based on the Bagging algorithm [29] to avoid this problem and alleviate the overfitting problem caused by data imbalance. Specifically, instead of using a single detector, we use the ensemble of multiple weaker detectors to detect malicious sessions. The architecture of the ensemble detector is shown in Figure 3.

For a specific session $s$ described as $\hat{s}$, it is input into $k$ subdetectors and its malicious scores are calculated in each detector, respectively. Then, the $k$ malicious scores are averaged to get the final malicious score of $\hat{s}$:

$$D_\theta(\hat{s}) = \frac{1}{T}\sum_{i=1}^{T} D_{\theta_i}(\hat{s}), \tag{4}$$

where $D_\theta(\hat{s})$ is a float ranged on (0, 1). When $D_\theta(\hat{s})$ is close to 1, the session $s$ is considered more likely to be malicious. Thus, for an input session $\hat{s}$ with a ground true label $y$, the loss function of the insider threat detector $D_\theta(\cdot)$ is as follows:

$$L_{\text{Mal}}(\hat{s}, y; \theta) = \text{CE}\big(D_\theta(\hat{s}), y\big) = \text{CE}\left(\frac{1}{T}\sum_{i=1}^{T} D_{\theta_i}(\hat{s}), y\right), \tag{5}$$

where CE stands for Cross-Entropy loss function.

In the training phase of each submodel, the bootstrap strategy used by the bagging algorithm requires each submodel to randomly sample the data without putting it back. In theory, the above training scheme is equivalent to using $N$ original data sets sampled from the same distribution, which alleviates the dependence of the detector on a single data set and reduces the variance of the ensemble model. However, as legitimate samples are the absolute majority in the training set, the bootstrap strategy is likely to lead to some trivial submodels, which have never seen any malicious samples in the training phase, making it completely loses the ability to distinguish malicious samples.

Therefore, we propose a new sampling strategy, which is called the Over-Bootstrap strategy, to sample the training data in the training phase. The training process with the Over-Bootstrap strategy is shown in Algorithm 1. If a submodel requires $m$ samples from the training set in a minibatch, we randomly sample $m/2$ legitimate sessions from the legitimate set and $m/2$ malicious sessions from the malicious set. This method can ensure that each submodel can see enough malicious samples in the training phase, to fully learn the difference between legitimate behaviors and malicious behaviors.

*3.4. Self-Supervised Learning for Users.* We notice that different users often have different behavior patterns due to their positions, ranks, and personality traits. It is important to distinguish the behavior characteristics between malicious users and legitimate users of which the behavior habits are different from most other users. Therefore, it will be helpful to learn distinctive behavior patterns among different users when detecting insider threats, which remain to be a difficult problem. Some works suggest training RNNs for every single user to learn the distribution of his behaviors [2]. However, due to a large number of users, it is not feasible to train RNNs for thousands of users.

To learn distinctive feature representation for different users and detect malicious behavior sequences more effectively, we append a self-supervised pretext task into the insider threat detector. Based on the feature representation extracted by the ensemble detector, we append a user identification network to judge which user the current input session belongs to. The architecture of the self-supervised network is shown in Figure 4.

In theory, there are some differences between our proposed self-supervised task and classical self-supervised learning. The input of classical self-supervised learning is usually unsupervised data, while based on the structure of the data itself [30] or artificial transformation[31], pretext tasks are constructed. With the help of pretext labels, the self-supervised network can be trained like supervised learning. But the pretext task we designed relies on an existing data field, which is just ignored in the training phase. In another view, our self-supervised pretext task can also be considered as a regularization item of detector loss, which can force LSTM to extract distinctive representations between different users.

Thus, for an input session $s$, of which the user ID is $u$, and the feature vector sequence is $\hat{s}$, the loss function of self-supervised user identifier $F_\varphi(\cdot)$ is as follows:

$$L_{\text{ID}}(\hat{s}, y; \varphi) = \text{CE}\big(F_\varphi(\hat{s}), u\big). \tag{6}$$

The training process of the combination model is as follows. As mentioned above, the proposed model has two independent parts, including an insider threat detector $D_\theta(\hat{s})$ and the self-supervised user identifier $F_\varphi(\cdot)$. They

Table 1: Characteristic of each dimension for the extracted feature.

| Dimension | Substance | Annotation |
|---|---|---|
| 0 | Whether use computer belongs to oneself | 1/0 stands for true/false |
| 1 | Time of behavior, represented by avg-TI | Normalized to (0, 1) |
| 2 | E-mail address sent to, represented by avg-TI | Normalized to [0, 1); 0 means do not send e-mail |
| 3 | E-mail address received from, represented by avg-TI | Normalized to [0, 1); 0 means do not receive e-mail |
| 4 | Name of operated file, represented by avg-TI | Normalized to [0, 1); 0 means do not operate files |
| 5 | URL of the accessed website, represented by avg-TI | Normalized to [0, 1); 0 means do not access website |
| 6 | Whether a log-on activity | 1/0 stands for true/false |
| 7 | Whether a log-off activity | 1/0 stands for true/false |
| 8 | Whether removable device connecting activity | 1/0 stands for true/false |

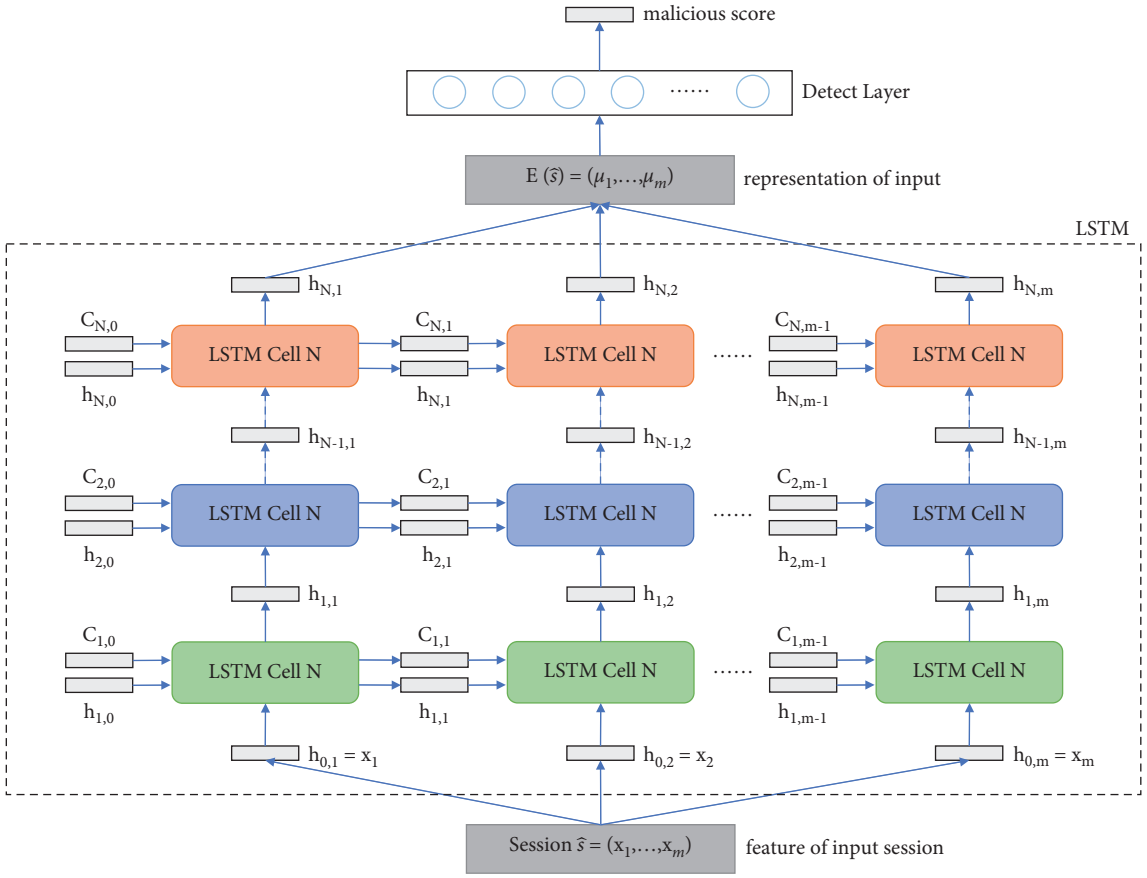


Figure 2: The architecture of the detector based on LSTM.

share an LSTM model for session representation. Since the output of the user identifier is a regularization term of the insider threat detector, we need to train the detector and the user identifier synchronously. Therefore, the training loss of the combination model also includes two independent parts. For an input session of which the feature vector sequence is $\hat{s}$, its ground true label is $y$ and its user ID is $u$. Then, its training loss function is formally described as follows:

$$L(\hat{s}, y, u; \theta) = (1 - \alpha)L_{\text{Mal}}(\hat{s}, y; \theta) + \alpha L_{\text{ID}}(\hat{s}, u; \varphi), \quad (7)$$

where $\alpha$ is a float ranged on [0, 1], which is used to balance the two different tasks. The training process of our combining model is shown in Algorithm 2.

## 4. Experiment

*4.1. Setup.* Experiments are conducted on the widely used CERT4.2 and CERT6.2 insider threat datasets [32] to evaluate the effectiveness of the proposed method. CERT4.2 contains 32770227 activity records generated by 1000 users, among which there are 70 users which contain 7327 malicious behaviors manually injected by experts in the field. CERT6.2 contains 135117169 activity records generated by 4000 users, among which there are 5 malicious users and 428 malicious behaviors. Obviously, the CERT6.2 dataset is more unbalanced, which means that detecting insider threats is more difficult. We use a single session as a base unit to detect insider threats and judge whether a given session is

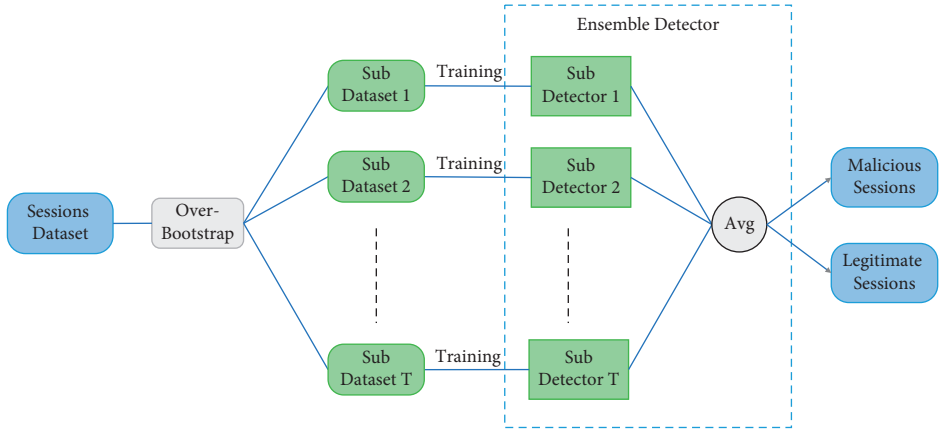


Figure 3: Architecture of ensemble detector based on bagging.

**input**: Set to legitimate sessions $S_{\text{teg}}$; Set of legitimate sessions $S_{\text{mal}}$; T: Number of subdetectors; iter:Iterations number; m:Batch size
**Output**: Ensemble detectors $D_{\theta(\cdot)}$
(1) : **Initialize**:**for** $t = 1, 2, \ldots T$ **do** rand_init $(\theta_t)$
(2) : **for** $n = 1, 2, \ldots$ iter **do**
(3) : **for** $t = 1, 2, \ldots T$ **do**
(4) Over-Bootstrap sampling:
(5) $\text{Rand\_sample}(S_{\text{leg}}) \longrightarrow S_{\text{legt}} = \{x_{\text{legi},0}\}_{i=1}^{m/2}$
(6) $\text{Rand\_sample}(S_{\text{mal}}) \longrightarrow S_{\text{malt}} = \{x_{\text{mali},0}\}_{i=1}^{m/2}$
(7) $\text{Merge}(S_{\text{legt}}, S_{\text{malt}}) \longrightarrow S_{n-t} = \{x_i, y_i\}_{i=1}^{m}$
(8) Train for a single subdetector
(9) $\text{Train}(D_{\theta_t}(\cdot)\text{: } S_{n-t}) \longrightarrow \theta_t$
(10) **end for**
(11) **end for**

Algorithm 1: Training process for ensemble detector with Over-Bootstrap strategy.

malicious or not. We use 90% of the log records to train the insider threat detector and the remaining 10% to evaluate the detection effect.

The architecture of the proposed model is shown in Table 2 As described in Section 3.4, a target session is first input to the Representation Part to get its feature representation. The feature representation is then input to the detector part and the user identifier part, respectively, of which the outputs are obtained for training or testing.

The hyperparameters used in training are shown in Table 3.

*4.2. Evaluation Scheme.* Like many existing works, we use the area under Receiver Operating Characteristic Curve (ROC) to evaluate the detection effect [28]. The horizontal axis of the ROC curve is a false positive rate (FPR), and the vertical axis is a true positive rate (TPR), or Detection Rate (DR). When given an input session, the detector outputs an exception score. AUC is equal to the probability that the malicious samples have higher exception scores than the legitimate samples. Therefore, the detector with 50% AUC is at a random level, while the detector with 100% AUC is perfect.

At the same time, to ensure that the experimental results are not affected by the partition of the dataset, we randomly divide the legitimate sample set and the malicious sample set in each experiment. For every setup, experiments have to be repeated 10 times, and the average AUC value of each experiment result is used to measure the effectiveness of the proposed method and methods used for comparison. In addition, we report DRs in some specific FPRs and the corresponding F1 score to evaluate the performance.

We select two recently published insider threat detection methods based on deep learning for comparison [2, 3]. As the feature extraction methods used in [2, 3] are almost the same, both of which are based on one-hot coding to represent a specific behavior, and take the counting in a period as the feature of a behavior sequence. So, the method to extract features in competing works is called Vanilla.

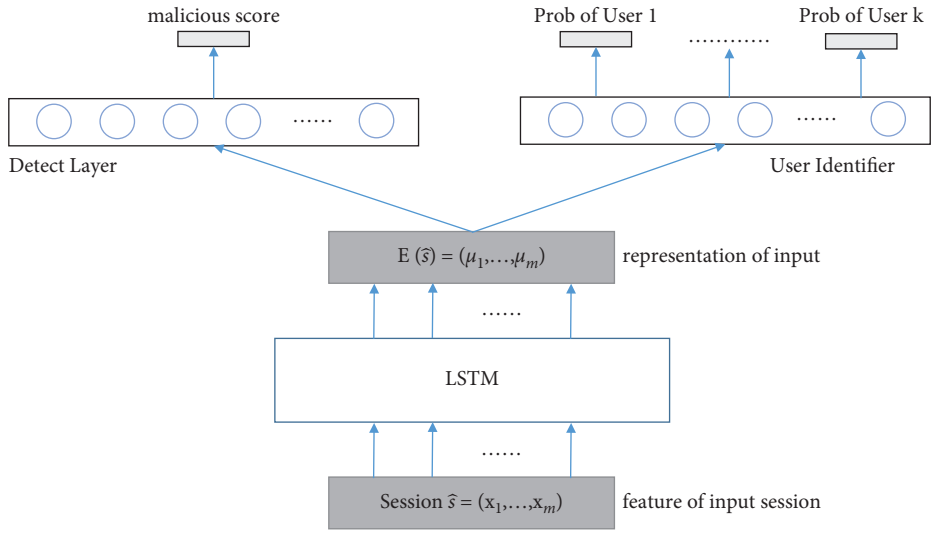


Figure 4: Architecture of network for proposed self-supervised pretext task.

**Input**: Set to legitimate sessions $S_{teg}$; Set of legitimate sessions$S_{mal}$; T: Number of subdetectors; iter:Iterations number; m:Batch size; $T_1, T_2$:learning rate of Detectors and User Recognizer; ID(·):User ID of input session

**Output**: Combined model $D_\theta(\cdot)$ and $F_\phi(\cdot)$

**initialize: for** $t = 1, 2, \ldots T$ do rand_init $(\theta_i)$
  **for** $n = 1, 2, \ldots$ iter **do**
    **for** $t = 1, 2, \ldots T$ **do**
      $L_D = 0; L_F = 0$
      Over-Bootstrap$(S_{leg}, S_{mal}) \longrightarrow S_{n-t} = \{x_i, y_i\}_{i=1}^{m}$
      **for** $t = 1, 2, \ldots m$ **do**
        $u_i = I\ D(x_i)$
        $L_F += CE(F_\phi(x_i), u_i)$
        $L_D += (1-\alpha) \cdot CE(D_\phi(x_i), y_i) + \alpha \cdot CE(F\ \phi(x_i), u_i)$
      **end for**
      $\phi \leftarrow \phi - T_2 \cdot \Delta L_F / \Delta\phi$
      $\theta_t \leftarrow \theta_t - T_2 \cdot \Delta L_D / \Delta\theta_t$
    **end for**
  **end for**
$D_\theta = (\widehat{S}) = 1/Tt- = 1 \sum_{t=1}^{T} D_\theta(\widehat{S}) \longrightarrow D_\theta(\cdot)$

Algorithm 2: Training algorithm for combined model.

Table 2: Architecture of proposed combining model.

| Part | Layer | Type | Parameter | Activation |
|---|---|---|---|---|
| Representation | LSTM | LSTM cell | (−, 100) | ReLU |
| | FC _1 | Fully connected | (100, 256) | ReLU |
| Detector | Drop1 | Dropout | 0.5 | — |
| | FC _2 | Fully connected | (256, 1) | Sigmoid |
| | FC_3 | Fully connected | (256, 512) | ReLU |
| User recognizer | Drop2 | Dropout | 0.5 | — |
| | FC_4 | Fully connected | (512, 1000) | — |

Table 3: Hyperparameters for training of proposed model.

| Hyperparameter | Value |
|---|---|
| Optimizer | SGD |
| Learning rate for detector | 0.01 |
| Learning rate for user identifier | 0.001 |
| Momentum | 0.9 |
| Batch size | 20 |
| $\alpha$ | 0.001 |
| Training steps | 5000 |

*4.3. Experimental Result.* We first study the impact of the number of subdetectors on the detection effect. We make the number of ROC detectors $T = 1$ (without ensemble), 5, and 10 and draw the ROC curve, respectively, as shown in Figure 5.

According to each ROC curve, the corresponding AUC values are 91.9%, 99.2%, and 95.1%, respectively.

It can be found that when $T = 5$, the detector has a much better effect than the scheme without ensemble. However, as the number of subdetectors further increased to $T = 10$, the

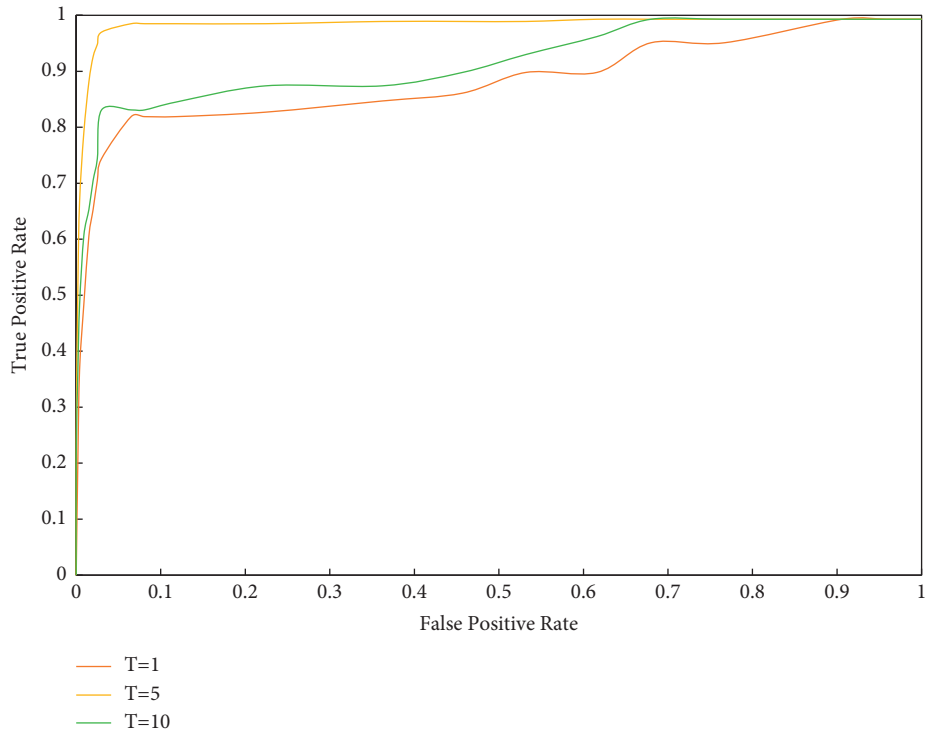


Figure 5: ROC curve for proposed method in $T = 1, 5, 10$.

Table 4: The results of the competing experiment and ablation experiment for the proposed method on CERT4.2.

| Method | DR (FPR = 0.05) | DR (FPR = 0.1) | DR (FPR = 0.5) | AUC (%) |
|---|---|---|---|---|
| Vanilla1 + LSTM-Diag [2] | 91.7% (93.3%) | 92.5% (91.2%) | 94.5% (65.4%) | 93.3 |
| Vanilla1 + DNN-Diag [2] | 92.1% (93.5%) | 92.7% (91.3%) | 94.4% (65.4%) | 93.6 |
| Vanilla2 + LSTM-CNN [3] | 92.9% (93.9%) | 93.8% (91.9%) | 95.7% (65.7%) | 94.5 |
| TF-IDF + OB + SS (ours) | 97.9% (96.4%) | 98.5% (94.1%) | 98.9% (66.4%) | 99.2 |
| Vanilla1 + LR | 85.8% (90.2%) | 86.9% (88.4%) | 92.1% (64.8%) | 87.6 |
| TF-IDF (+LSTM) | 88.1% (91.4%) | 89.5% (89.7%) | 93.5% (68.0%) | 89.9 |
| TF-IDF + SS | 89.4% (92.1%) | 91.1% (90.5%) | 92.6% (64.9%) | 91.9 |
| TF-IDF + Bagging | 93.8% (94.4%) | 94.5% (92.2%) | 95.9% (65.7%) | 95.4 |
| TF-IDF + Bagging + SS | 95.1% (95.0%) | 95.8% (92.8%) | 96.2% (65.8%) | 97.1 |
| TF-IDF + OB + SS (ours) | 97.9% (96.4%) | 98.5% (94.1%) | 98.9% (66.4%) | 99.2 |

SS: self-supervised; OB: Over-Bootstrap. The values within parentheses are F1 values corresponding to DR and specific FPR.

detection effect decreased obviously. With the increase of the number of subdetectors, many malicious samples are repeatedly sampled many times, which may cause the detector to bias to malicious samples instead, thus introducing a new overfitting problem. Therefore, in the following experiments, we choose $T = 5$ as the number of subdetectors.

Competing experiments: the results of the competing experiment on CERT4.2 are shown in the upper part of Table 4. It can be found that the DRs and F1 values of the proposed method are higher than those of the competing method under different specific FPRs. In addition, the AUC value of the proposed method is 99.2%, which is 4.7% higher than that of the best competing method.

The results on CERT6.2 are shown in the upper part of Table 5. Similar to the results on CERT 4.2, the DR, F1, and AUC values of the proposed method are higher than those of all competing methods. The most representative is that the AUC value of the proposed method is 95.3%, which is 2.6% higher than the best competing result.

It can be noted that the experimental results of the proposed method on CERT6.2 are lower compared with results in CERT4.2, which can be attributed to the fact that CERT6.2 contains much less abnormal samples than CERT4.2. Although our method considers the imbalance of abnormal samples, the detection effect is still not ideal under the extreme imbalance case in CERT6.2. Therefore, our future work will focus on extending our work to unsupervised or a few shot scenarios to further improve the detection effect.

At the same time, we conduct ablation experiments to verify our proposed techniques, including feature extraction method based on TF-IDF, Over-Bootstrap sampling

Table 5: The results of the competing experiment and ablation experiment for the proposed method on CERT6.2.

| Method | DR (FPR = 0.05) | DR (FPR = 0.1) | DR (FPR = 0.5) | AUC (%) |
|---|---|---|---|---|
| Vanilla1 + LSTM-Diag[2] | 84.7% (89.6%) | 86.1% (88.0%) | 88.3% (63.8%) | 90.4 |
| Vanilla1 + DNN-Diag[2] | 84.5% (89.4%) | 86.4% (88.2%) | 88.5% (63.9%) | 90.8 |
| Vanilla2 + LSTM-CNN[3] | 88.2% (91.5%) | 89.9% (89.9%) | 90.7% (64.5%) | 92.7 |
| TF-IDF + OB + SS (ours) | 92.4% (93.7%) | 93.7% (91.8%) | 94.8% (65.5%) | 95.3 |
| Vanilla1 + LR | 79.4% (86.5%) | 80.1% (84.8%) | 83.2% (62.5%) | 84.2 |
| TF-IDF (+LSTM) | 84.2% (89.3%) | 85.1% (87.5%) | 88.5% (63.9%) | 88.7 |
| TF-IDF + SS | 86.5% (90.6%) | 88.7% (89.3%) | 90.4% (64.4%) | 91.5 |
| TF-IDF + Bagging | 86.7% (90.7%) | 88.5% (89.2%) | 89.9% (64.3%) | 91.1 |
| TF-IDF + Bagging + SS | 91.8% (93.4%) | 93.1% (91.5%) | 94.2% (65.3%) | 94.4 |
| TF-IDF + OB + SS (ours) | 92.4% (93.7%) | 93.7% (91.8%) | 94.8% (65.5%) | 95.3 |

SS: self-supervised; OB: Over-Bootstrap. The values within parentheses are F1 values corresponding to DR and specific FPR.

strategy, and user identification self-supervised pretext task. A classifier based on Linear Regression is used as a baseline. The results of the ablation experiment are shown in Tables 4 and 5.

We can note that when TF-IDF and classical bagging algorithm are used, our detection effect has already exceeded [2, 3]. It shows that the use of TF-IDF-based feature extractor and ensemble learning will bring great improvement together. Besides, the rest of the results in the ablation experiment show that the self-supervised pretext task and Over-Bootstrap sampling strategy can further improve the detection effect.

## 5. Conclusion

In this paper, a supervised insider threat detection method based on ensemble learning and self-supervised learning is proposed, which is used to determine whether the sessions in the user log are malicious. Firstly, we introduce the TF-IDF algorithm from NLP to express operated entities from different event sources with their importance and aggregate user behaviors into short vectors. On this basis, we design an ensemble binary classifier to detect the input session and propose a new sampling strategy called Over-Bootstrap to avoid the waste of subdetectors. To better distinguish malicious samples from outlier samples, we design a self-supervised pretext task to enable detectors to generate distinctive feature representations for different users. The CERT v4.2 dataset is used to evaluate the proposed method. Experimental results show that the proposed method can effectively detect malicious sessions in CERT4.2 and CERT6.2 datasets, where the AUCs are 99.2% and 95.3% in the best case.

## Data Availability

The CERT4.2 and CERT6.2 datasets used to support the findings of this study have been deposited in the Insider Threat Test Dataset repository (DOI: https://doi.org/10.1184/R1/12841247.v1).

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This work was supported by the Defense Industrial Technology Development Program (JCKY2018603B006) and CAEP Foundation (CX2019040).

## References

[1] 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*, vol. 20, no. 2, pp. 1397–1417, 2018.

[2] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," 2017, https://arxiv.org/abs/1710.00811.

[3] F. Yuan, Y. Cao, Y. Shang, Y. Liu, J. Tan, and B. Fang, "Insider threat detection with deep neural network," *Lecture Notes in Computer Science*, in *Proceedings of the International Conference on Computational Science*, pp. 43–54, Springer, Wuxi, China, June 2018.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[5] J. Glasser and B. Lindauer, "Bridging the gap: a pragmatic approach to generating insider threat data," in *Proceedings of the 2013 IEEE Security and Privacy Workshops*, pp. 98–104, IEEE, San Francisco, CA, USA, May 2013.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[7] M. Ahmed, A. Naser Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[8] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, pp. 3448–3470, 2007.

[9] S. Shakya and S. Sigdel, "An approach to develop a hybrid algorithm based on support vector machine and Naive Bayes for anomaly detection," in *Proceedings of the 2017 International Conference on Computing, Communication and Automation (ICCCA)*, pp. 323–327, IEEE, Greater Noida, India, May 2017.

[10] I. Syarif, A. Prugel-Bennett, and G. Wills, "Unsupervised clustering approach for network anomaly detection," in *Proceedings of the International Conference on Networked Digital Technologies*, pp. 135–145, Springer, Dubai, UAE, April 2012.

[11] H. Saeedi Emadi and S. M. Mazinani, "A novel anomaly detection algorithm using DBSCAN and SVM in wireless sensor networks," *Wireless Personal Communications*, vol. 98, no. 2, pp. 2025–2035, 2018.

[12] R. P. Lippmann and R. K. Cunningham, "Improving intrusion detection performance using keyword selection and neural networks," *Computer Networks*, vol. 34, no. 4, pp. 597–603, 2000.

[13] H. Eldardiry, B. Evgeniy, L. Juan, H. John, P. Bob, and B. Oliver, "Multi-domain information fusion for insider threat detection," in *Proceedings of the 2013 IEEE Security and Privacy Workshops*, IEEE, San Francisco, CA, USA, May 2013.

[14] T. Rashid, I. Agrafiotis, and J. R. C. Nurse, "A new take on detecting insider threats: exploring the use of hidden Markov models," in *Proceedings of the 8th ACM CCS International Workshop on Managing Insider Security Threats*, Vienna, Austria, October, 2016.

[15] K. Veeramachaneni, "AI2: training a big data machine to defend," in *Proceedings of the IEEE International Conference on High Performance & Smart Computing & IEEE International Conference on IEEE International Conference on Big Data Security on Cloud IEEE*, San Jose, CA, USA, April 2016.

[16] B. Sharma, P. Pokharel, and B. Joshi, "User behavior analytics for anomaly detection using LSTM autoencoder-insider threat detection," in *Proceedings of the 11th International Conference on Advances in Information Technology*, Bangkok, Thailand, July 2020.

[17] D. C. Le and N. Zincir-Heywood, "Anomaly detection for insider threats using unsupervised ensembles," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1152–1164, 2021.

[18] Z. Li, X. Cheng, L. Sun, J. Zhang, and B. Chen, "A hierarchical approach for advanced persistent threat detection with attention-based graph neural networks," *Security and Communication Networks*, vol. 2021, Article ID 9961342, 14 pages, 2021.

[19] P. Wanda and J. Jie, "DeepProfile: finding fake profile in online social network using dynamic CNN," *Journal of Information Security and Applications*, vol. 52, Article ID 102465, 2020.

[20] T. Hu, W. Niu, and X. Zhang, "An insider threat detection approach based on mouse dynamics and deep learning," *Security and Communication Networks*, vol. 2019, Article ID 3898951, 12 pages, 2019.

[21] N. Narang and S. Kar, "A hybrid trust management framework for a multi-service social IoT network," *Computer Communications*, vol. 171, pp. 61–79, 2021.

[22] W. Meng, W. Li, and J. Zhou, "Enhancing the security of blockchain-based software defined networking through trust-based traffic fusion and filtration," *Information Fusion*, vol. 70, pp. 60–71, 2021.

[23] Z. Ma, L. Liu, and W. Meng, "Towards multiple-mix-attack detection via consensus-based trust management in IoT networks," *Computers & Security*, vol. 96, Article ID 101898, 2020.

[24] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, https://arxiv.org/abs/1312.6114.

[25] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," *Proceedings. Presses universitaires de Louvain*, vol. 89, pp. 89–94, 2015.

[26] D. Kim, D. Seo, S. Cho, and P. Kang, "Multi-co-training for document classification using various document representations: TF-IDF, LDA, and Doc2Vec," *Information Sciences*, vol. 477, pp. 15–29, 2019.

[27] W. Luo, W. Liu, and S. Gao, "A revisit of sparse coding based anomaly detection in stacked rnn framework," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 341–349, Venice, Italy, October 2017.

[28] T. Ergen and S. S. Kozat, "Unsupervised anomaly detection with LSTM neural networks," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 8, pp. 3127–3141, 2019.

[29] X. Tao, Y. Peng, F. Zhao, S. Wang, and Z. Liu, "An improved parallel network traffic anomaly detection method based on bagging and GRU," in *Proceedings of the International Conference on Wireless Algorithms, Systems, and Applications*, pp. 420–431, Springer, Qingdao, China, September 2020.

[30] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, https://arxiv.org/abs/1301.3781.

[31] H. Lee, S. J. Hwang, and J. Shin, "Self-supervised label augmentation via input transformations," in *Proceedings of the International Conference on Machine Learning*, pp. 5714–5724, PMLR, Virtual Event, July 2020.

[32] B. Lindauer, *Insider Threat Test Dataset*, Carnegie Mellon University, Pittsburgh, PA, USA, 2020.