applied sciences

MDPI

Article

Insider Threat Detection Based on Deep Clustering of Multi-Source Behavioral Events

Jiarong Wang, Qianran Sun and Caiqiu Zhou *

Institute of High Energy Physics, Chinese Academy of Sciences (CAS), Beijing 100049, China
* Correspondence: zhoucq@ihep.ac.cn

Abstract: With the continuous advancement of enterprise digitization, insider threats have become one of the primary cybersecurity concerns for organizations. Therefore, it is of great significance to develop an effective insider threat detection mechanism to ensure the security of enterprises. Most methods rely on artificial feature engineering and input the extracted user behavior features into a clustering-based unsupervised machine learning model for insider threat detection. However, feature extraction is independent of clustering-based unsupervised machine learning. As a result, user behavior features are not the most appropriate for clustering-based unsupervised machine learning, and thus, they reduce the insider threat detection accuracy. This paper proposes an insider threat detection method based on the deep clustering of multi-source behavioral events. On the one hand, the proposed method constructs an end-to-end deep clustering network and automatically learns the user behavior feature expression from multi-source behavioral event sequences. On the other hand, a deep clustering objective function is presented to jointly optimize the learning of feature representations and the clustering task for insider threat detection. This optimization can adjust the optimal user behavior features for the clustering model to improve the insider threat detection accuracy. The experimental results show that the proposed end-to-end insider threat detection model can accurately identify insider threats based on abnormal multi-source user behaviors in enterprise networks.

Keywords: insider threat; multi-source user behaviors; deep clustering

check for updates

Citation: Wang, J.; Sun, Q.; Zhou, C. Insider Threat Detection Based on Deep Clustering of Multi-Source Behavioral Events. *Appl. Sci.* **2023**, *13*, 13021. https://doi.org/10.3390/app132413021

Academic Editor: Dimitris Mourtzis

Received: 19 October 2023
Revised: 11 November 2023
Accepted: 28 November 2023
Published: 6 December 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Insiders are current or former employees, contractors, or business partners of an organization who have authorized access to the organization's network, system, or data [1]. In unwanted situations, these insiders lose loyalty to their employers either out of anger or out of greed for personal benefit. This provokes them to behave maliciously and intentionally harm the organization's resources and damage its reputation.

In recent years, several incidents of insider threats have made their way to the public media, one of which was the high-profile insider case involving Edward Snowden, and another was the data leakage cases by Chelsea Manning and Kim. In the insider threat cybersecurity report released in 2023, 74% of the respondents believed that insider attacks have become more frequent in the past 12 months [2]. Unlike external attacks, attacks from insiders are hard to detect because malicious insiders have legitimate access to information on various facilities and are also well informed about the organization and its critical facilities. Meanwhile, it becomes easy for a malicious insider to carry out malicious actions and hide their hacking trail.

Insider threat detection has attracted significant attention over the last decade. Early approaches achieve insider threat detection by analyzing users' behaviors via a single type of audit data, such as system call audit data [3], keyboard and mouse dynamics audit data [4,5], UNIX command execution audit data [6], and file access audit data [7]. However,

these methods are limited in their ability to detect more complex insider attacks owing to their reliance on a single type of user behavior data.

Recent research methods have focused on analyzing user behaviors from multiple types of audit data. Insider threat detection based on multi-source audit logs can combine behavioral analysis ability from multiple data sources, which enhances the capability to detect various types of attacks. To analyze user behaviors across different data sources, it is common to extract the statistical features of user behavior from several data sources and input these features into supervised or unsupervised machine learning algorithms for insider threat detection [8–10]. Compared to supervised machine learning, unsupervised machine learning is more practical due to the lack of labeled malicious samples in the real network environment. However, user behavior feature extraction using the most appropriate unsupervised machine learning model for insider threat detection remains a challenge, especially when using the clustering unsupervised machine learning method. On the one hand, traditional artificial feature engineering requires manual work for feature extraction, which makes it difficult to capture user behavior features efficiently because of the complex and nonlinear nature of user behaviors in cyberspace. On the other hand, feature extraction is independent of clustering-based unsupervised machine learning. It results in suboptimal user behavior feature representations for the clustering algorithm, which leads to the reduced accuracy of insider threat detection.

Given the powerful learning ability of deep neural networks, we utilize deep neural networks to automatically learn feature representations of multi-source user behavior sequences, which are then input into a deep clustering network for insider threat detection. In order to achieve this goal, there are two main challenges: (1) every user behavior event contains multiple entities, and it is necessary to learn the features of such behavior event sequences, which can represent the temporal relationships between events and between entities of events; (2) it is necessary to simultaneously perform the feature mapping of behavior event sequences and the assignment of features to cluster centers based on the deep clustering network.

To overcome the abovementioned two challenges, this paper proposes an insider threat detection method based on the deep clustering of multi-source behavioral events. Firstly, an encoder–decoder model was built to learn the feature representation of user behavior sequences via multi-source behavior audit logs. The encoder involves embedding layers and recurrent neural network (RNN) layers, which transform the multi-source user behavior sequences into fixed-length feature vectors. In addition, a multi-output decoder was constructed to predict the entities of the next user behavior event following a user behavior event sequence. Since estimating the latent generating function of the data has better predictive performance than estimating the data themselves [11], instead of directly predicting the entities of the behavior event itself, we leverage the decoder to estimate the latent generating function of the entities of the behavior event. Subsequently, the deviation between the predicted parameter values of the generating function and the actual parameter values of the generating function is used as the loss to adjust the encoder–decoder network parameters. The proposed encoder–decoder model can obtain the initial user behavior features by capturing temporal relationships between user behavior events and between entities of user behavior events.

Secondly, a deep clustering model of user behavior sequences is proposed to obtain clustering centroids and optimize the user behavior feature representation by using deep neural networks. In this model, we initialize the cluster centers through the application of k-means clustering to the initial feature vectors of user behavior event sequences obtained from the encoder. The objective function for deep clustering is defined based on intra-cluster compactness and inter-cluster separation. Additionally, the stochastic gradient descent algorithm is employed to update both the behavior sequence feature vectors and cluster centers. This iterative process enables the automatic adjustment of the user behavior feature representation suitable for the clustering task, achieving the optimal user behavior features and clustering results simultaneously.

At last, based on the user behavior features and clustering results, sparse clusters and outliers that deviate from the cluster centers can be identified and detected as insider threats.

The main contributions of this paper are as follows.

- A new end-to-end insider threat detection method is proposed for automatically learning user behavior features from multi-source audit logs by using a deep neural network.
- A new deep clustering model for user behavior sequences is proposed to optimize the user behavior features for the clustering task and improve the detection of insider threats.
- Using the CERT benchmark datasets from Carnegie Mellon University, the proposed end-to-end insider threat detection method was evaluated in terms of frequently used metrics such as Recall, ROC curve, and area under the ROC curve (AUC).

2. Related Work

Insider threat detection has attracted significant attention. Most existing approaches achieve insider threat detection by analyzing users' behaviors via audit data, such as host-based data that record activities of users on their own computers, network-based data that are recorded by network equipment, and context data that record users' profile information.

Host-based insider threat detection focuses on developing behavioral anomaly detection techniques with the objective of finding out anomalousness or abnormal changes at the host level activity. For example, user behaviors can be analyzed in terms of the historical occurrence frequency of each subsequence of Unix commands, and then the anomaly can be detected using one-class SVM [6,12] and Naive Bayes classification [13]. In addition, keyboard and mouse-based detection methods create a user profile by extracting a number of features based on the frequency of clicks or movements [14,15], and different key-up and key-down times [4,16], with classifiers used to detect any inconsistent dynamic that indicates the existence of a malicious insider. File-based malicious insider detection methods [7,17] extract multi-dimensional feature vectors from file access paths over a fixed period and input these features into a two-class or one-class classifier.

Network-based insider threat detection methods model the user profile from the network activity for detecting anomalous application-level violations such as Email communication and web browsing usage. Email communication can be structured into a graph in which the nodes represent senders and recipients. Graph structure features are extracted and input into machine learning methods to detect insider threats [18,19]. Some works build the user profile based on a user's web browsing behavior from the perspective of page access frequency and page view time, and abnormal behaviors can be detected by comparing the cosine similarity between current and historical browsing behaviors [20].

Host-based and network-based insider threat detection methods are limited in their ability to detect more complex insider attacks since they analyze only a single type of user behavior data. In recent studies, researchers prefer to combine host-based and network-based data from multi-source audit logs to provide stronger user behavior analysis capabilities and improve insider threat detection.

PRODIGAL (PROactive Detection of Insider threats with Graph Analysis and Learning) [8,9] extracts over 100-dimensional user behavior features from multiple audit logs, including email and proxy server logs. It then employs various machine learning algorithms such as KDE, GMM, kNN, and HMM to build multiple detectors, each using a different subset of features. The Beehive system [10] extracts 15-dimensional user behavioral features for each host daily by analyzing server proxy logs, DHCP, VPN, and LDAP logs in an enterprise network. Principal Component Analysis (PCA) is then used to reduce the dimensionality of the feature vectors. The reduced features are input into the k-means clustering algorithm to identify abnormal user behaviors. Young et al. [21] proposes ensemble-based clustering algorithms for detecting anomalies in multi-source user behavior. This method extracts over 100 dimensions of features from various user behavior logs, and inputs them into ensemble learning algorithms, such as Gaussian Mix-

ture Models, for insider threat detection. However, the above-mentioned methods rely on artificial feature engineering, where domain experts manually extract statistical features from multi-source user behavior data based on their prior knowledge.

To automatically learn user behavior features, some researchers have utilized embedding learning techniques to achieve low-dimensional vector representations of user behavior. Liu et al. [22] introduce the log2vec model, which constructs a graph based on multi-source user behavior. Each behavior event is treated as a vertex in the graph, and different edges are defined using ten rules. Graph embedding learning techniques are then employed to learn vector representations of the vertices (i.e., each behavior event). Finally, the k-means algorithm is used to cluster the vertices, and those in clusters with fewer vertices are identified as anomalies. However, the feature extraction is independent of the clustering method, resulting in suboptimal user behavior feature representations for the clustering algorithm which leads to reduced accuracy in insider threat detection.

To learn the feature representation suitable for the clustering task, Cao et al. [23] propose a community-based anomaly detection model. This method represents every user behavior as an embedding vector, which is then iteratively optimized using k-means clustering. Subsequently, clusters of user behaviors can be obtained as communities and abnormal behavior is evaluated based on the distance between a behavior and other behaviors within the same community. However, this method analyzes static behavior and ignores the temporal relationships among user behaviors in the user behavior sequences.

Table 1 summarizes existing methods in the field of insider threat detection. It can be seen from Table 1 that user behavior feature extraction for multi-source-data-based insider threats detection remains a challenge for two reasons: (1) traditional artificial feature engineering requires manual work for feature extraction, and (2) the feature extraction is independent of the clustering-based unsupervised machine learning, which results in suboptimal user behavior feature representations.

Table 1. Overview of existing methods for insider threat detection.

Study	Type of Data	Source of Data	Features	ML/Statistical Model	Remarks
[6,12,13]	Host-based audit data	UNIX commands execution audit data	Occurrence frequency of each subsequence of the Unix commands	One-class SVM, Naive Bayes	Host-based and network-based insider threat detection methods are limited in their ability to detect more complex insider attacks since they typically analyze only a single type of user behavior data to model the user profile.
[14–17]	Host-based audit data	Keyboard or mouse dynamics audit data	Frequency of clicks or movements, different key-up and key-down times	One-class SVM, KNN, neural networks	
[18,19]	Host-based audit data	File access audit data	File path distance, file access frequency	TreeBagger	
[20,21]	Network-based audit data	Email audit data	Email graph structure	Naive Bayes, Decision Trees, Random Forests and Bagging	
[22]	Network-based audit data	Web browsing audit data	Page access frequency, page view time	Cosine Similarity	
[8–10,23]	Multi-source audit data	Email logs, proxy server logs	File access times, web browsing times	KDE, GMM, kNN, and HMM	These methods rely on artificial feature engineering, where domain experts manually extract statistical features from multi-source user behavior data based on their prior knowledge.
[24]	Multi-source audit data	Email logs, proxy server logs	Feature learning by using graph embedding learning techniques	Graph embedding learning, Kmeans	The feature extraction is independent of the clustering method, resulting in suboptimal user behavior feature representations for the clustering algorithm which leads to reduced accuracy in insider threat detection.
[25]	Multi-source audit data	Network-level event, process-level event	Feature learning by using embedding learning techniques	Embedding learning	This method analyzes static behavior and ignores the temporal relationships among user behaviors in the user behavior sequences.

Some researchers have proposed using deep neural networks for feature representation due to their powerful feature learning capability [24–27]. In addition, these deep neural networks can simultaneously obtain both the feature representation and clustering results, enabling the optimization of feature representations in the clustering space. Compared to traditional step-by-step clustering algorithms, deep clustering provides an end-to-end solution for clustering tasks. Deep clustering algorithms have achieved good performance in the field of image analysis. However, existing deep clustering algorithms cannot be directly applied to user behavior analysis due to the differences between images and multi-source audit logs.

To automatically learn user behavior features, we apply deep neural networks to user behavior analysis and propose a novel end-to-end insider threat detection method based on user behavior sequences from multi-source audit data. Firstly, an encoder–decoder model is built to learn the user behavior features by capturing temporal relationships between user behavior events and between entities of user behavior events. Secondly, the user behavior features can be optimized to adapt to the clustering method by using the proposed deep clustering model of user behavior sequences, thereby improving the detection of insider threats.

3. Proposed Method

This paper proposes an end-to-end deep clustering model for insider threat detection, which learns the feature representation and cluster centers of multi-source user behavior sequences. The objective function of the model includes two main components: one for learning the feature representation of multi-source user behavior sequences and another for deep clustering.

Firstly, an encoder–decoder model is trained. The encoder encodes the feature representation of multi-source behavior sequences, and the decoder predicts the last behavior event in the sequence. The deviation between the predicted value and the true value is used as the objective function to adjust the parameters for learning the feature representation of multi-source user behavior sequences. After training, the decoder is discarded, and the encoder is used to obtain the feature space of multi-source user behavior sequences. The cluster centers for k-means clustering are initialized using this feature space.

Secondly, the objective function for deep clustering is defined, considering intra-cluster compactness and inter-cluster separation. The clustering centers are refined iteratively using stochastic gradient descent, and the feature representation of user behavior sequences is adjusted backward to optimize the clustering objective function. Finally, based on the clustering results, sparse clusters and outliers that deviate from the cluster centers are identified and detected as anomalies.

3.1. Problem Statement

Consider a multi-source behavior sequence of a user, which contains ℓ consecutive behavior events, denoted as $s = \{e_i\}_{i=1}^{l}$, where each event e_i represents a user behavior and event $e_i = (x_1, \cdots, x_m)$ consists of m distinct entities, such as time, host ID, and operation.

Assuming we have n normal multi-source behavior event sequences $\{s_q \in S\}_{q=1}^{n}$, we first train an encoder–decoder model. The encoder takes as input a multi-source behavior sequence of length ℓ and outputs a fixed-length feature vector z_l for each sequence. The decoder, using the fixed-length feature vector z_l, predicts all entities of the $(\ell + 1)$-th behavior event. The deviation between the predicted value and the true value is used as the loss function to adjust the model parameters. After training, we discard the decoder and keep only the encoder.

Given a set of multi-source behavior event sequences S', we input these sequences into the encoder to obtain the user multi-source behavior sequence feature space z_l. We then initialize the cluster centers $u_j, j = 1, \ldots, k$ using k-means clustering. Next, we use stochastic gradient descent to iteratively update the cluster centers $u_j, j = 1, \ldots, k$, and the user multi-source behavior sequence feature space z_l based on the objective function of

deep clustering. Deep clustering outputs the class labels and class assignment probabilities of the multi-source behavior event sequences in the set S'. Finally, we identify the clusters that are relatively sparse and the user multi-source behavior sequences with low class assignment probabilities as anomalies.

3.2. Deep Clustering Network

Considering n user multi-source behavior event sequences $\{s_q \in S\}_{q=1}^{n}$, the objective of our proposed deep clustering network is to simultaneously learn the feature representation and class labels of user multi-source behavior event sequences. This process consists of two parts: user multi-source behavior sequence feature representation learning and deep clustering. We first pretrain the network to initialize the user multi-source behavior sequence feature space and cluster centers, then train the network to optimize the feature space and cluster centers.

The network structure, shown in Figure 1, consists of an encoder–decoder model and the deep clustering part. We first introduce the training of the encoder–decoder model to learn the feature space of user multi-source behavior event sequences, which is the first stage of learning in the deep clustering network.

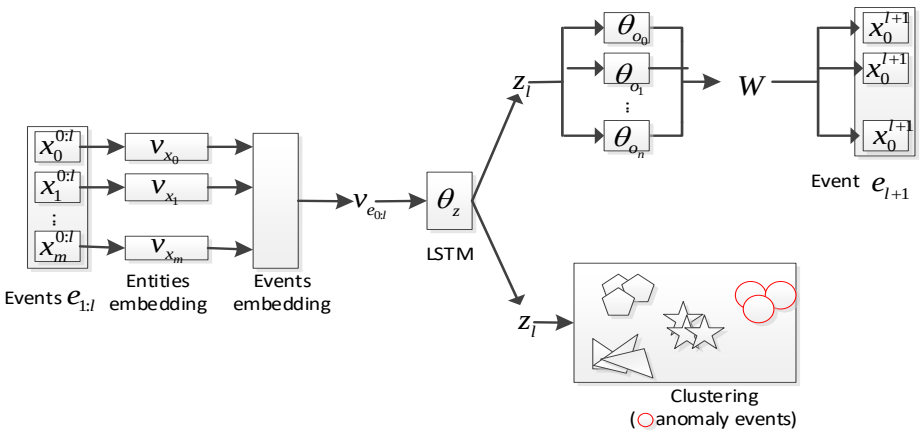


Figure 1. User multi-source behavior deep clustering network framework. (The black symbols in the clustering represent different behavior patterns, and the red circles indicate the abnormal behaviors.)

3.2.1. User Multi-Source Behavior Sequence Feature Representation

For the learning of user multi-source behavior sequence feature representation, the learning objective is an encoder–decoder model. The encoder consists of embedding layers and recurrent neural network layers. The embedding layers learn the feature representation of entities in user behavior events, which are then concatenated to form the feature representations of entities to form the feature representation of behavior events. The feature representations of consecutive user multi-source behavior events are input into the recurrent neural network, which outputs a fixed-length hidden state vector. A multi-output prediction decoder is constructed based on this vector. The deviation between the predicted values and the actual values is used as the objective function to adjust the network parameters. After training, the hidden state vector output by the encoder serves as the feature representation of the user multi-source behavior sequence. The structure of the encoder–decoder model is shown in Figure 2.

For a multi-source behavior sequence s containing ℓ consecutive behavior events, $s = \{e_i\}_{i=1}^{l}$, where each behavior event $e_i = (x_1, \cdots, x_m)$ consists of m different entities. We utilize an embedding layer to learn the embedding vectors of the entities based on their correlation. For any entity $x_p \in e_i, 1 \leq p \leq m$, the embedding layer outputs the embedding vector of the entity as $v_{x_p} = x_p \times V$, where V represents the embedding feature space of the entities. The higher the correlation between entities, the closer the distance between embedding vectors of the entities in the embedding space.

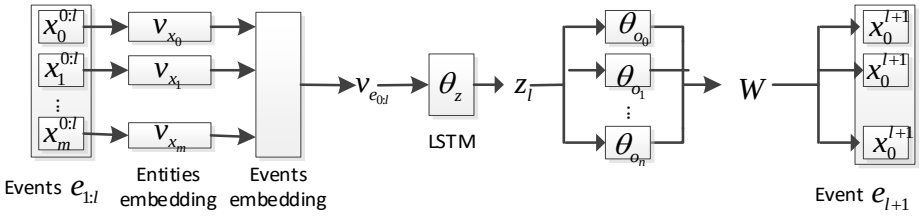


Figure 2. Encoder–decoder network framework for learning user multi-source behavior sequence feature representation.

Then, we concatenate the embedding vectors of the entities contained in the user behavior event as the feature representation of the behavior event. For a behavior event e_i, the feature representation of the event is $v_{e_i} = concatenate(v_{x_1}, \ldots, v_{x_m})$, where v_{x_1} is the embedding vector of entity $x_1 \in e_i$, and v_{x_m} is the embedding vector of entity $x_m \in e_i$. Furthermore, we input the feature vectors of consecutive user behavior events into a RNN to learn the temporal relationships between user behavior events. The RNN outputs a fixed-length hidden state vector as the feature representation of the user behavior event sequence. The feature representation of the user multi-source behavior sequence $s = \{e_i\}_{i=1}^{l}$ can be denoted as:

$$z_l = \mathrm{RNN}(v_{e_1}, \ldots, v_{e_l}) \tag{1}$$

The embedding layer and RNN layer together form the encoder, which encodes the multi-source behavior sequence and outputs a fixed-length hidden state vector. Based on this vector, a multi-output prediction decoder is constructed. It has been found that estimating the underlying generating function of the data has better predictive performance than directly estimating the data itself. Therefore, instead of directly predicting the entities contained in the behavior events, we utilize a multi-output predictor to predict the underlying generating function of the entities. For a behavior sequence s containing ℓ consecutive user multi-source behavior events, $s = \{e_i\}_{i=1}^{l}$, let the underlying generating function of the $\ell + 1$ event $e_{l+1} = (x'_1, \cdots, x'_m)$ be the $x'_{1:m} \sim g(0 : m-1; W)$, where W represents the parameters of the underlying generating function g. Instead of modeling $p((x'_1, \cdots, x'_m) \mid e_1, e_2, \ldots, e_l)$ to predict the entities contained in the behavior event, we model $p(W \mid e_1, e_2, \ldots, e_l)$ to predict the parameters W of the underlying generating function g.

The encoder takes as input a user multi-source behavior event sequence $s = \{e_i\}_{i=1}^{l}$ of length ℓ and generates a hidden state vector z_ℓ for the multi-source behavior sequence s. The decoder then utilizes the hidden state vector z_ℓ to predict the underlying generating function g for the $\ell + 1$ event $e_{\ell+1}$. Assuming that g is a polynomial function of degree r, the function g contains $r + 1$ real-valued parameters, denoted as $g(0 : m-1; W) = w_0 + w_1 l + \cdots + w_r l^r$. The decoder predicts the real-valued parameters of the polynomial function g as:

$$w_t = o_t(z_l; \theta_{o_t}), t \in [0, r] \tag{2}$$

In the given context, the hidden state vector z_ℓ can be obtained by using Formula (1). θ_{o_t} is the neural network parameter. o_t is the prediction function of the decoder.

For each multi-source behavior event sequence containing ℓ consecutive user behavior events, the decoder predicts a set of parameters $W = \{w_t\}$ that represent the underlying generating function g. This allows the function g to approximate the joint distribution of the entities in the $\ell + 1$ event $e_{l+1} = (x'_1, \cdots, x'_m)$ optimally. The decoding process involves polynomial function prediction, wherein the prediction of the underlying generating function enables the learning of the joint distribution correlation among the entities in the behavior events.

To train the encoder–decoder model, we calculate the true values of the polynomial fitting parameters for the entities $(x'_1, \cdots, x'_m)$ contained in the user behavior event $e_{\ell+1}$. These true values serve as the ground truth for the parameters W of the polynomial function

g predicted by the decoder. The deviation between the predicted and true parameter values and the true function parameter values is used as the loss function to adjust the model parameters. Once the model training is complete, we discard the decoder and extract the encoder. The hidden state vector of the user multi-source behavior sequence, which is outputted by the encoder, serves as the initial values for the features of the user multi-source behavior sequence. Using these initial feature vectors, we apply the k-means clustering algorithm to perform clustering and obtain k initial cluster centers $\{u_j \in Z\}_{j=1}^{k}$.

3.2.2. Deep Clustering of User Multi-Source Behavior Sequences

Given the initial values z_ℓ of the user multi-source behavior sequence feature vectors and k initial cluster centers $\{u_j\}_{j=1}^{k}$, we perform deep clustering of user multi-source behavior sequences. Firstly, we define the objective function for deep clustering, and then utilize the stochastic gradient descent (SGD) algorithm to iteratively refine the cluster centers and adjust the user behavior sequence feature representations. This process jointly optimizes the learning of user behavior sequence feature representations and the clustering task. Compared to traditional methods, it can learn user behavior sequence feature representations that are suitable for clustering without requiring additional supervision. It can simultaneously learn the optimal user behavior sequence feature representations and clustering results. The structure of deep clustering is illustrated in Figure 3.

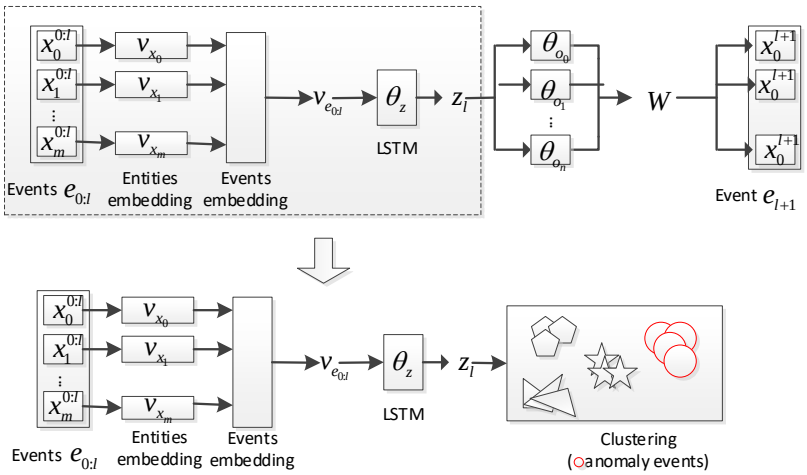


Figure 3. Deep clustering network for multi-source user behavior sequences. (The black symbols in the clustering represent different behavior patterns, and the red circles indicate the abnormal behaviors.)

First, we define the objective function for deep clustering based on the clustering objectives of intra-cluster compactness and inter-cluster separation. The objective function for deep clustering is defined as follows:

$$O = \sum_{q=1}^{n}\sum_{j=1}^{k} p_{jq}||z_l^q - u_j||^2 - \sum_{q=1}^{n}\sum_{j=1}^{k}\sum_{j'=1}^{k} ||u_j - u_{j'}||^2 \quad (3)$$

In the given context, n represents the number of user multi-source behavior sequences, k represents the number of cluster centers, z_l^q represents the feature vector of the user multi-source behavior sequence $s_q \in S$, and $\{u_j\}_{j=1}^{k}$ represents the cluster centers. p_{jq} represents the probability that the q-th multi-source behavior sequence belongs to cluster j. The objective function for deep clustering consists of two terms. The first term calculates the distance between any multi-source behavior sequence and the cluster centers. Minimizing the first term helps to maintain intra-cluster compactness, ensuring that the sequences within each cluster are similar. The second term calculates the distance between cluster

centers. Maximizing the second term helps to maintain inter-cluster separation, ensuring that different clusters are distinct.

To evaluate the similarity between the feature vector z_l^q of a user multi-source behavior sequence and the cluster center u_j, we can use Student's t-distribution as a kernel. The allocation probability p_{jq} of a multi-source behavior sequence to a cluster center is defined as:

$$p_{jq} = \frac{\left(1 + ||z_l^q - u_j||^2/\alpha\right)^{-\frac{\alpha+1}{2}}}{\sum_{j'}\left(1 + ||z_l^q - u_{j'}||^2/\alpha\right)^{-\frac{\alpha+1}{2}}} \tag{4}$$

The parameter α is set to a value of 1 in the experiments.

According to the clustering objective function O, we can update the cluster centers $\{u_j\}_{j=1}^k$ and the feature vectors z_l^q of the multi-source behavior sequences using the SGD algorithm. On one hand, the gradient update for the feature vectors z_l^q of the multi-source behavior sequences in the objective function O is given by:

$$\frac{\partial O}{\partial z_l^q} = \frac{\alpha+1}{\alpha}\sum_j\left(1 + \frac{||z_l^q - u_j||^2}{\alpha}\right)^{-1} \times ||z_l^q - u_j||^3 + 2p_{jq}\left(z_l^q - u_j\right) \tag{5}$$

In the given context, the allocation probability p_{jq} can be obtained by using Formula (4).

The gradient update $\frac{\partial OO}{\partial z_l^q}$ for the feature vector z_l^q of the multi-source behavior sequences is backpropagated to the encoder, adjusting the embedding representation of the behavior event entities and the feature representation of the behavior event sequences from the recurrent neural network. On the other hand, the gradient update for the cluster centers u_j in the objective function O is given by:

$$\frac{\partial O}{\partial u_j} = -\frac{\alpha+1}{\alpha}\sum_j\left(1 + \frac{||z_l^q - u_j||^2}{\alpha}\right)^{-1} \times ||z_l^q - u_j||^3 - 2p_{jq}\left(z_l^q - u_j\right) - 2\sum_{j'} u_j - u_{j'} \tag{6}$$

Based on the gradient $\frac{\partial O}{\partial u_j}$, we iteratively refine the cluster centers. The gradient descent process stops when the cluster centers no longer change between two consecutive iterations. By simultaneously updating the cluster centers and the feature representations of the multi-source user behavior event sequences, we can obtain the optimal feature space for clustering and the clustering results.

The process of deep clustering of multi-source user behavior sequences can be summarized as Algorithm 1.

Algorithm 1 Algorithm for Deep Clustering of Multi-source User Behavior Sequences

Input: User multi-source behavior sequence training set S, collection of user multi-source behavior sequences for detection S', each behavior sequence contains l consecutive behavioral events.

Output: Class labels and class assignment probabilities for sequences in the user multi-source behavior sequence collection S'.

1: for a sequence $s = \{e_i\}_{i=1}^l$, $s \in S$ do
2: To compute the feature vector z_ℓ of a multi-source behavior sequence s using encoder based on Formula (1).
3: To predict the reference values of the polynomial generating function for the entity contained in the $(\ell+1)$-th behavior event using the decoder based on Formula (2).
4: The true values of the polynomial fitting for the entity in the $(\ell+1)$-th behavior event as the parameters of the generating function.
5: The deviation between the true values and predicted values of the parameters of the generating function as the loss function to adjust the encoder and decoder.
6: end for
7: To extract the feature vector z_l' of a sequence from the user's multi-source behavior sequence collection S' generated by the encoder.
8: Using k-means clustering to initialize the class centers u_j based on the collection S'.
9: for a sequence $s' = \{e'_i\}_{i=1}^l$, $s' \in S'$ do
10: To update the feature vector z_l' of a multi-source behavior sequence according to Formula (5).
11: To update the cluster centers u_j based on Formula (6).
12: end for

3.3. Anomaly Detection

For the given set of multi-source behavior sequences S', the deep clustering network outputs the class labels and class assignment probabilities for the user behavior event sequences in the set S'. Based on the clustering results, we consider two types of anomalies: cluster anomalies and outlier anomalies.

Assuming that the behavior event sequences in the user's multi-source behavior sequence collection S' are clustered into k classes. To begin, calculate the standard deviation of the number of behavior event sequences for each class label:

$$\sigma = \sqrt{\frac{1}{k}\sum_{j=1}^{k}(x_j - \mu)^2} \tag{7}$$

In the given context, x_j represents the number of behavior event sequences for the j-th class label, while μ represents the average number of behavior event sequences for all class labels.

The standard deviation of the quantity of behavior event sequences for class labels reflects the balance of the distribution of behavior sequences in the clustering results. A larger standard deviation indicates an imbalanced distribution of behavior sequences in the clustering results. In this case, clusters with fewer behavior sequences are identified as anomaly clusters, and all behavior sequences within those clusters are considered anomalies. Conversely, a smaller standard deviation indicates a balanced distribution of behavior sequences in the clustering results. In this scenario, if there are behavior sequences with low class assignment probabilities to their corresponding clusters, those sequences are classified as outliers.

4. Experiment

4.1. Datasets

To validate the effectiveness of our method, we conducted experiments using the CMU-CERT insider threat dataset r4.2 [28]. This dataset consists of audit records of five types of behaviors of 1000 employees: host logins, file accesses, email communications, web browsing, and mobile device connections, spanning 17 months from January 2010 to May 2011 within an enterprise. By detecting anomalous user behaviors in this dataset, we evaluated the insider detection performance of the algorithm.

The dataset includes three types of anomalous user behaviors involving 70 malicious insiders. The first type involves 30 insiders who typically do not work after office hours. However, one day they suddenly logged into the host after office hours and used a USB device to copy data, later uploading it to WikiLeaks. The second type includes 30 insiders who started browsing a large number of job search websites and, before leaving the company, increased their frequency of using USB devices to steal data. The third type includes 10 insiders who are system administrators. These individuals downloaded a keylogger and copied it to the host of a department leader using a USB device. The following day, they used the collected keylogs to log into the department leader's host and sent a large number of emails, causing panic within the company.

To preprocess the different types of behavior records for users, each behavior record was transformed into an event, with the type of behavior becoming the type of the event. After preprocessing, we obtain a collection of events, each including the following five entities:

- Time entities: extracted from the hour field of the time field in the user behavior records, and the values for time entities range from [0, 1, 2, . . ., 23], representing the 24-h clock.
- Host entities: extracted from the host number field in the user behavior records, with the host number serving as the identifier for the host entity.
- User entities: extracted from the user number field in the user behavior records, with the user number serving as the identifier for the user entity.

- Action entities: Extracted from the action field in the user behavior records, with specific actions serving as the identifier for the action entity. There are seven action entities: login (host), logout (host), connect (mobile device USB), disconnect (mobile device USB), browse (web), send (email), and access (file).
- Action magnitude: Derived by counting the number of repetitions for each action and discretizing the counts into different levels of action magnitude. The action magnitude entity reflects the current level of repetition for that action entity.

Given a fixed length l of the multi-source user behavior sequence, the user behavior event collection was divided into multiple behavior event sequences, forming a multi-source user behavior sequence dataset. The first 70% of the multi-source behavior event sequences in the multi-source user behavior sequence dataset were used as the training dataset to train the encoding–decoding model and learn the feature representation of the event sequences, containing only normal samples. The remaining 30% of the multi-source behavior event sequences in the dataset were used as the detection dataset for detecting abnormal behavior event sequences.

4.2. Experimental Setup

In the experiment, the proposed detection method was compared with the following existing insider threat detection methods.

BAIT [29] proposed a semi-supervised classification method for detecting abnormal user behavior. This method defines 28-dimensional user behavior features, with 16 dimensions being basic features that calculate the frequency of user behavioral operations. The remaining 12 dimensions are combinations of these basic features, such as the ratio between one basic feature and another. The feature vector is then input into a support vector machine and Naive Bayes algorithms for classifying abnormal user behavior.

Isolation Forest [30] was proposed as an unsupervised multi-source user behavior anomaly detection method. This method extracts 42-dimensional features from multiple sources of user behavior, including email usage features, email content features, login/logout features, application software usage features, and web browsing usage features. The average, weighted average, and weighted difference of the feature values for each day are calculated and concatenated as the feature vector. The Isolation Forest algorithm is then used to detect anomalies.

Scenario-Based [31] was proposed as a semi-supervised multi-source user behavior anomaly detection method. Initially, 20-dimensional features are extracted from multiple sources of user behavior, such as login counts, file download counts, and email sending counts. A cost-sensitive approach is employed to undersample normal behavior samples to balance the number of normal and abnormal samples. Random Forest and Deep Autoencoder are then utilized to classify abnormal user behavior.

In the experiment, the proposed method set the dimension of the embedding vector for event entities to 10. The recurrent neural network layer in the network was set as the Gate Recurrent Unit (GRU). The deep clustering algorithm was set to have 10 clusters. The length of the multi-source user behavior sequences was set to 8.

4.3. Evaluation Metrics

When evaluating a detection system, the results can be categorized as either correct or incorrect, with all possible outcomes falling into the following four scenarios:

(1) True positive (*TP*): the sub-activities of abnormal behavior are detected as abnormal;
(2) True negative (*TN*): the sub-activities of normal behavior are detected as normal;
(3) False positive (*FP*): the sub-activities of normal behavior are detected as abnormal, resulting in a false alarm;
(4) False negative (*FN*): the sub-activities of abnormal behavior are detected as normal, resulting in a missed detection.

If TP, TN, FP, and FN represent the number of occurrences of the aforementioned scenarios, the following metrics can be defined:

$$\textit{False Positive Rate } (FPR) = \frac{FP}{TN + FP}$$

$$\textit{True Positive Rate } (TPR) = \frac{TP}{TP + FN}$$

For each threshold set, the corresponding values of FPR and TPR are obtained. By varying the threshold from 0 to the maximum, initially every behavior sub-activity is predicted as an anomaly. As the threshold increases, the number of predicted anomalous sub-activities decreases until none are left. During this process, FPR and TPR values are calculated at each step, and a graph is plotted with FPR as the x-axis and TPR as the y-axis, known as the "ROC curve". By plotting the ROC curves of different detectors on the same coordinate system, it becomes possible to visually evaluate their performance. Detectors with ROC curves closer to the top-left corner represent higher accuracy.

If two ROC curves do not intersect, the curve closer to the top-left corner represents a better-performing detector. However, in practical tasks, the situation is often more complex, and if the ROC curves intersect, it is challenging to make a general assertion about superiority. In such cases, if a comparison must be made, the Area Under the ROC Curve (AUC) is a reasonable criterion for comparison. AUC represents the overall performance of the detection model, with values closer to 1 indicating better performance and values closer to 0 indicating poorer performance. The calculation of AUC accounts for the detector's classification ability for both anomalous and normal behaviors, even in situations with imbalanced normal and anomalous samples. AUC is not sensitive to the balance of sample categories, which is why it is commonly used to evaluate the performance of anomaly detection models.

4.4. Results for Insider Threat Detection

Firstly, the proposed method was used to detect three types of abnormal behaviors in the dataset. We refer to these three types of abnormal behaviors as type 1, type 2, and type 3. Figure 4 shows the ROC curves and AUC values of the proposed method for detecting type 1, type 2, and type 3 abnormal behaviors. From the graph, it can be observed that the ROC curve of the proposed method for type 2 abnormal behavior is closest to the upper-left corner, indicating that the proposed method is most effective in detecting type 2 abnormal behavior. Furthermore, based on the AUC values for detecting all three types of abnormal behaviors, it can be seen that the proposed method achieves AUC values exceed 90% for all three types, indicating good detection results for all three types of abnormal behaviors.

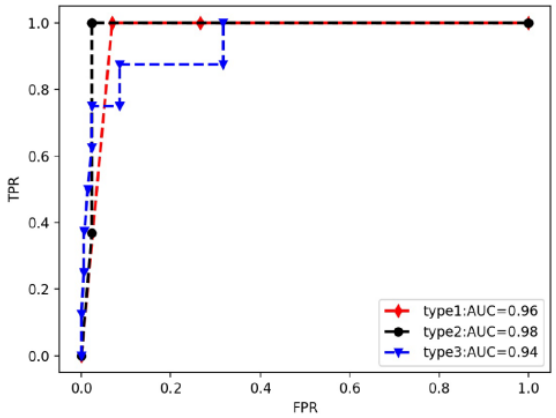


Figure 4. The ROC curves and corresponding AUC values for the detection of type 1, type 2, and type 3 abnormal behaviors.

Next, the proposed method was compared with existing detection methods, including BAIT, Isolation Forest, Random Forest, and Deep Autoencoder. These methods fuse user statistical features at the behavioral level across multiple detection domains and extract multi-source behavioral features for inputting into supervised or unsupervised machine learning algorithms for anomaly detection. BAIT, Random Forest, and Deep Autoencoder are supervised anomaly detection algorithms, while Isolation Forest is an unsupervised anomaly detection algorithm. The proposed method fuses user behaviors in different detection domains based on the temporal nature of behavioral events and detects anomalies in the user's multi-source behavioral event sequences. The proposed method is an unsupervised anomaly detection method. Since Random Forest and Deep Autoencoder in the comparison methods use *Recall* for performance evaluation, this section also adopts *Recall* as the evaluation metric for all the compared methods in the experimental comparison. *Recall* represents the percentage of correctly predicted abnormal behavior samples in the user behavior samples.

$$Recall = \frac{TP}{TP + FN}$$

In the given context, *TP* represents the number of abnormal behavior samples correctly detected as abnormal, while *FN* represents the number of abnormal behavior samples incorrectly detected as normal.

Table 2 presents the *Recall* rates of all comparison methods for detecting type 1, type 2, and type 3 abnormal behaviors. The bolded sections indicate the best detection results for each type of abnormal behavior. From the first three columns of Table 2, it can be observed that the proposed method outperforms other comparison methods in terms of *Recall* rates for type 1 and type 2 abnormal behaviors. However, the Deep Autoencoder achieves the highest Recall rate for type 3 abnormal behavior. The last column of Table 2 represents the average *Recall* rate of the detection methods for the three types of abnormal behaviors, reflecting the comprehensive detection capability of the methods for abnormal behaviors. According to the last column of Table 2, it can be observed that the *Recall* rate of Random Forest is higher than BAIT and Isolation Forest. There may be two reasons for this: (1) The Random Forest method extracts more effective user behavior features based on specific attack scenarios. (2) Random Forest utilizes both normal and abnormal behavior samples to train the classifier, resulting in a more accurate classification model. It is a supervised anomaly detection method, while Isolation Forest is an unsupervised anomaly detection method. In addition, Deep Autoencoder and Random Forest extract the same user behavior features, but Deep Autoencoder achieves a higher *Recall* rate than Random Forest. This may be because deep neural networks have better data learning capabilities. Furthermore, Deep Autoencoder and the proposed method achieve similar *Recall* rates. However, the proposed method is an unsupervised anomaly detection method that does not rely on prior knowledge from domain experts to design and extract user behavior feature vectors.

Table 2. *Recall* rates of existing comparative methods for detecting three types of abnormal behaviors.

Method	**Anomaly Type**			
	Type 1	**Type 2**	**Type 3**	**Avg**
BAIT	51.29	54.60	49.04	51.64
Isolation Forest	82.09	95.68	71.08	82.95
Random Forest	89.58	85.94	96.35	90.62
Deep Autoencoder	90.25	99.48	**94.10**	94.61
Proposed method	**98.01**	**99.84**	87.50	**95.11**

Based on the experiments conducted, it can be concluded that the proposed method achieves favorable ROC curves and AUC values for all three types of abnormal behaviors.

This validates the capability of the proposed method to simultaneously detect multiple types of abnormal behaviors without relying on prior knowledge from domain experts to extract user behavior feature vectors. Furthermore, when comparing the experimental results with existing detection methods, both the proposed method and the Deep Autoencoder demonstrate commendable *Recall* rates. This finding confirms that deep neural networks possess superior learning capabilities when contrasted with traditional machine learning methods. However, the detection method of the Deep Autoencoder requires prior knowledge for designing and extracting feature vectors, as well as supervised training of the classifier using samples of abnormal behavior. In contrast, the proposed method utilizes the constructed encoder–decoder model to automatically learn user behavior feature representations, while performing clustering and anomaly detection. This unsupervised anomaly detection approach is more suitable for practical detection environments.

4.5. Results for Different Parameters

In the process of anomaly detection of user behavior based on multi-source behavior deep clustering, the deep clustering model includes three main hyperparameters: the number of class labels k for user multi-source behavior sequences, the number of consecutive behavioral events l contained within the user multi-source behavior sequences, and the dimension h of the embedded feature vectors of the behavioral events' entities. In order to investigate the impact of different hyperparameters on deep clustering anomaly detection, this section adjusts the parameter values of different parameters and compares the anomaly detection results of different parameter values. In this section, AUC was used as the evaluation metric for all parameter values in the comparative experiments. In the experiment, the default parameter values for the three hyperparameters were set to $k = 10$, $l = 8$, and $h = 10$. When adjusting the parameter value of one parameter, the remaining parameters were kept at their default values.

Figure 5 illustrates the impact of the length l of user multi-source behavior sequences on the deep clustering anomaly detection method. As the parameter value of l increases, the AUC values for detecting the first type (type 1) and the second type (type 2) of abnormal behaviors remain relatively stable, while the AUC value for detecting the third type (type 3) of abnormal behavior shows an increase.

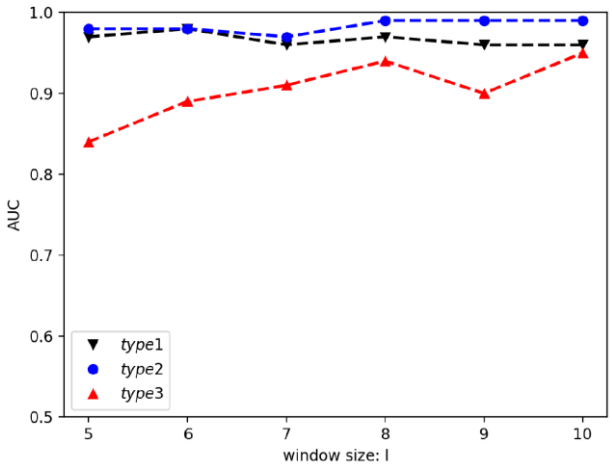


Figure 5. Different detection results when adjusting the length l of user multi-source behavior sequences.

Figure 6 illustrates the impact of the number of class labels k in user multi-source behavior sequences on the deep clustering anomaly detection method. With an increase in the value of k, the AUC value for detecting the second type (type 2) of abnormal behavior remains relatively stable. However, the AUC values for detecting the first type (type 1) and the third type (type 3) of abnormal behaviors initially rise and then fall. The maximum AUC value for detection is achieved when the number of class labels k ranges between 10 and 12.

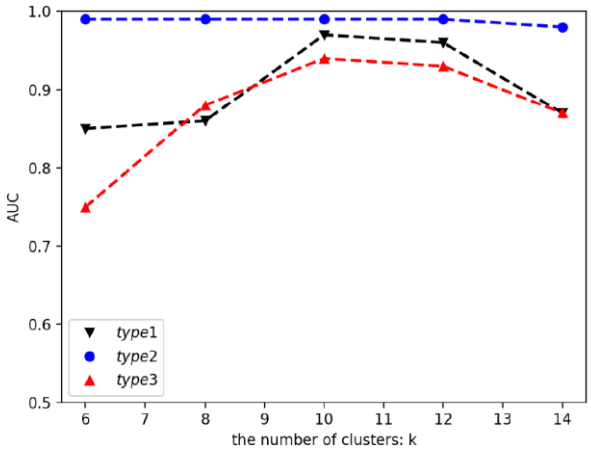


Figure 6. Different detection results when adjusting the number of class labels k of user multi-source behavior sequences.

Figure 7 illustrates the impact of the dimension h of embedded feature vectors of user behavior events' entities on the deep clustering anomaly detection method. As the parameter value of h increases, the AUC value for detecting the second type (type 2) of abnormal behavior remains relatively stable. However, the AUC values for detecting the first type (type 1) and the third type (type 3) of abnormal behaviors fluctuate slightly.

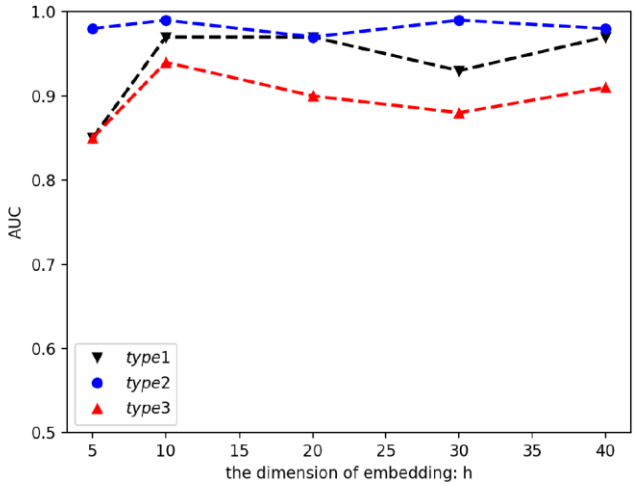


Figure 7. Different detection results when adjusting the dimensionality h of user behavior event entity embedding feature vectors.

Based on the above experiments, we can conclude that among the three hyperparameters, the length l of user multi-source behavior sequences and the dimension h of the embedded feature vectors of user behavior event entities have a relatively minor impact on deep clustering anomaly detection. Conversely, the number of class labels k in user multi-source behavior sequences significantly affects the performance of deep clustering anomaly detection.

5. Conclusions

This paper proposes an end-to-end insider threat detection method based on deep neural networks. The proposed method firstly employs the encoder–decoder model to effectively learn feature representation of user behavior sequences. Subsequently, a deep clustering network is constructed, which iteratively refines cluster centers using a clustering objective function and adjusts the parameters of the deep neural network through the stochastic gradient descent (SGD) algorithm. Abnormal user behaviors are then identified

based on the degree of behavior deviation from the cluster centers. The proposed method improves insider threat detection by jointly optimizing user behavior clustering and feature representation, thus addressing the challenge of learning an optimal behavior sequence representation suitable for clustering. The experimental evaluation, conducted on the publicly available CMU-CERT insider threat dataset, shows that the proposed method achieved 98% AUC and 99.8% *Recall* for the second type of malicious insiders in the dataset. The overall comparison with the existing detection methods demonstrates that the proposed method obtains superior detection performance.

In this study, the proposed method can be applied to any organization in which a substantial amount of user behavior data can be collected. It is particularly effective for datasets with diverse types of user behavior logs, including host-based and network-based behavior logs. A significant factor influencing detection accuracy is the predetermined number of clusters required by the deep clustering algorithm. Future research will focus on developing a method to automatically select the optimal number of clusters based on the characteristics of the dataset in use.

Author Contributions: J.W. and C.Z.: methodology, formal analysis, writing—original draft, writing—review and editing. Q.S., J.W. and C.Z.: software, English spelling check. J.W. and C.Z.: conceptualization, validation, data curation. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61931019 and the Xiejialin Project of Institute of High Energy Physics under Grant no. E25467U2.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cappelli, D.M.; Moore, A.P.; Trzeciak, R.F. *The CERT Guide to Insider Threats: How to Prevent, Detect, and Respond to Information Technology Crimes (Theft, Sabotage, Fraud)*; Addison-Wesley: Boston, MA, USA, 2012.
2. Insider Threat Report [EB/OL]. 2023. Available online: https://www.cybersecurity-insiders.com/portfolio/2023-insider-threat-report-gurucul/ (accessed on 19 October 2023).
3. Parveen, P.; Evans, J.; Thuraisingham, B.; Hamlen, K.W.; Khan, L. Insider threat detection using stream mining and graph mining. In Proceedings of the 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing, Boston, MA, USA, 9–11 October 2011; pp. 1102–1110.
4. Morales, A.; Fierrez, J.; Ortega-Garcia, J. Towards predicting good users for biometric recognition based on keystroke dynamics. In Proceedings of the Computer Vision-ECCV 2014 Workshops, Zurich, Switzerland, 6–7 and 12 September 2014; Part II 13; Springer International Publishing: Berlin/Heidelberg, Germany, 2015; pp. 711–724.
5. Hu, T.; Niu, W.; Zhang, X.; Liu, X.; Lu, J.; Liu, Y. An insider threat detection approach based on mouse dynamics and deep learning. *Secur. Commun. Netw.* **2019**, *2019*, 3898951. [CrossRef]
6. Salem, M.B.; Stolfo, S.J. A comparison of one-class bag-of-words user behavior modeling techniques for masquerade detection. *Secur. Commun. Netw.* **2012**, *5*, 863–872. [CrossRef]
7. Camiña, J.B.; Monroy, R.; Trejo, L.A.; Medina-Pérez, M.A. Temporal and spatial locality: An abstraction for masquerade detection. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 2036–2051. [CrossRef]
8. Senator, T.E.; Goldberg, H.G.; Memory, A.; Young, W.T.; Rees, B.; Pierce, R.; Huang, D.; Reardon, M.; Bader, D.A.; Chow, E.; et al. insider threats in a real corporate database of computer usage activity. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, IL, USA, 11 August 2013; pp. 1393–1401.
9. Young, W.T.; Goldberg, H.G.; Memory, A.; Sartain, J.F.; Senator, T.E. Use of domain knowledge to detect insider threats in computer activities. In Proceedings of the 2013 IEEE Security and Privacy Workshops, San Francisco, CA, USA, 19–22 May 2013; pp. 60–67.
10. Yen, T.F.; Oprea, A.; Onarlioglu, K.; Leetham, T.; Robertson, W.; Juels, A.; Kirda, E. Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In Proceedings of the 29th Annual Computer Security Applications Conference, New Orleans, LA, USA, 9 December 2013; pp. 199–208.

11. Fox, I.; Ang, L.; Jaiswal, M.; Pop-Busui, R.; Wiens, J. Deep multi-output forecasting: Learning to accurately predict blood glucose trajectories. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 23 August 2018; pp. 1387–1395.
12. Salem, M.B.; Stolfo, S.J. Detecting Masqueraders: A Comparison of One-Class Bag-of-Words User Behavior Modeling Techniques. *J. Wirel. Mob. Netw. Ubiquitous Comput. Dependable Appl.* **2010**, *1*, 3–13.
13. Maxion, R.A.; Townsend, T.N. Masquerade detection using truncated command lines. In Proceedings of the International Conference on Dependable Systems and Networks, Washington, DC, USA, 23–26 June 2002; pp. 219–228.
14. Ahmed, A.A.E.; Traore, I. A new biometric technology based on mouse dynamics. *IEEE Trans. Dependable Secur. Comput.* **2007**, *4*, 165–179. [CrossRef]
15. Shen, C.; Cai, Z.; Guan, X.; Du, Y.; Maxion, R.A. User authentication through mouse dynamics. *IEEE Trans. Inf. Forensics Secur.* **2012**, *8*, 16–30. [CrossRef]
16. Ahmed, A.A.; Traore, I. Biometric recognition based on free-text keystroke dynamics. *IEEE Trans. Cybern.* **2013**, *44*, 458–472. [CrossRef]
17. Camiña, B.; Monroy, R.; Trejo, L.A.; Sánchez, E. Towards building a masquerade detection method based on user file system navigation. In *Advances in Artificial Intelligence: Proceedings of the 10th Mexican International Conference on Artificial Intelligence, MICAI 2011, Puebla, Mexico, 26 November–4 December 2011*; Part I 10; Springer: Berlin/Heidelberg, Germany, 2011; pp. 174–186.
18. Eberle, W.; Graves, J.; Holder, L. Insider threat detection using a graph-based approach. *J. Appl. Secur. Res.* **2010**, *6*, 32–81. [CrossRef]
19. Patil, A.; Liu, J.; Shen, J.; Brdiczka, O.; Gao, J.; Hanley, J. Modeling attrition in organizations from email communication. In Proceedings of the 2013 International Conference on Social Computing, Alexandria, VA, USA, 8–14 September 2013; pp. 331–338.
20. Yang, Y.C. Web user behavioral profiling for user identification. *Decis. Support Syst.* **2010**, *49*, 261–271. [CrossRef]
21. Young, W.T.; Memory, A.; Goldberg, H.G.; Senator, T.E. Detecting unknown insider threat scenarios. In Proceedings of the 2014 IEEE Security and Privacy Workshops, San Jose, CA, USA, 17–18 May 2014; pp. 277–288.
22. Liu, F.; Wen, Y.; Zhang, D.; Jiang, X.; Xing, X.; Meng, D. Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, UK, 11–15 November 2019; pp. 1777–1794.
23. Cao, C.; Chen, Z.; Caverlee, J.; Tang, L.A.; Luo, C.; Li, Z. Behavior-based community detection: Application to host assessment in enterprise information networks. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino, Italy, 22–26 October 2018; pp. 1977–1985.
24. Xie, J.; Girshick, R.; Farhadi, A. Unsupervised deep embedding for clustering analysis. In *Proceedings of the International Conference on Machine Learning*; PMLR: London, UK, 2016; pp. 478–487.
25. Yang, J.; Parikh, D.; Batra, D. Joint unsupervised learning of deep representations and image clusters. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 5147–5156.
26. Ghasedi Dizaji, K.; Herandi, A.; Deng, C.; Cai, W.; Huang, H. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 5736–5745.
27. Caron, M.; Bojanowski, P.; Joulin, A.; Douze, M. Deep clustering for unsupervised learning of visual features. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 132–149.
28. Software Engineering Institute, Carnegie Mellon University. [n. d.]. Insider Threat Test Dataset. Available online: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099 (accessed on 27 January 2021).
29. Azaria, A.; Richardson, A.; Kraus, S.; Subrahmanian, V.S. Behavioral analysis of insider threat: A survey and bootstrapped prediction in imbalanced data. *IEEE Trans. Comput. Soc. Syst.* **2014**, *1*, 135–155. [CrossRef]
30. Gavai, G.; Sricharan, K.; Gunning, D.; Rolleston, R.; Hanley, J.; Singhal, M. Detecting insider threat from enterprise social and online activity data. In Proceedings of the 7th ACM CCS International Workshop on Managing Insider Security Threats, Denver, CO, USA, 16 October 2015; pp. 13–20.
31. Chattopadhyay, P.; Wang, L.; Tan, Y.P. Scenario-based insider threat detection from cyber activities. *IEEE Trans. Comput. Soc. Syst.* **2018**, *5*, 660–675. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.