CapsITD: Malicious Insider Threat Detection Based on Capsule Neural Network

Haitao Xiao[1,2], Chen Zhang[1], Song Liu[1], Bo Jiang[1,2], Zhigang Lu[1,2], Fei Wang[3], and Yuling Liu[1,2(✉)]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{xiaohaitao,zchen,liusong1106,jiangbo,luzhigang,liuyuling}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
wangfei@ict.ac.cn

Abstract. Insider threat has emerged as the most destructive security threat due to its secrecy and great destructiveness to the core assets. It is very important to detect malicious insiders for protecting the security of enterprises and organizations. Existing detection methods seldom consider correlative information between users and can not learn the extracted features effectively. To address the aforementioned issues, we present CapsITD, a novel user-level insider threat detection method. CapsITD constructs a homogeneous graph that contains the correlative information from users' authentication logs and then employs a graph embedding technique to embed the graph into low-dimensional vectors as structural features. We also design an anomaly detection model using capsule neural network for CapsITD to learn extracted features and identify malicious insiders. Comprehensive experimental results on the CERT dataset clearly demonstrate CapsITD's effectiveness.

Keywords: Insider threat detection · Capsule neural network · Graph embedding

1 Introduction

Nowadays, insider threats are acknowledged as one of the most dangerous cyber threats to an organization's network and data security. Insider threats are harder to detect than external threats since insiders are generally permitted to access internal information systems and are knowledgeable about the organization's structure and security procedures. According to Securonix's insider threat report 2020 [1], 80% of employees who are about to terminate their employment with their company tend to take some sensitive data with them. The insider threat report 2021 issued by Gurucul shows that 98% of organizations feel vulnerable

F. Li et al. (Eds.): SecureComm 2022, LNICST 462, pp. 57–71, 2023.
https://doi.org/10.1007/978-3-031-25538-0_4

to insider attacks and 49% of organizations can not detect insider threats or can only detect them after data has left the organization [2]. Security problems caused by insiders are becoming more and more serious. It is vital to detect insider threats accurately and promptly.

In order to detect malicious insiders, many approaches have been proposed. These approaches can be divided into signature-based approaches and anomaly-based approaches. Signature-based approaches mainly depend on known-bad events' signatures and can not detect unknown threats. Current anomaly-based approaches are mainly focused on user behavior profiles and use machine learning algorithms [3], deep learning algorithms [4,5] to detect malicious insiders. There are two limitations in the previous approaches. Firstly, previous approaches [3,5] have not considered the correlative information between users. The correlative information can reflect the user's aggregation. Users with the same behavior tend to have similar attributes, and such correlative information can help detect malicious insiders. Secondly, previous approaches [4] fail to learn the extracted features effectively, and there is an urgent need to find a more suitable learning method that can adequately learn the extracted features.

To overcome these limitations, we propose CapsITD, a user-level malicious insider threat detection method based on capsule neural network, which leverages graph embedding technique and capsule neural network. First, we extract statistical features based on users' daily activities and communications. Then, we construct a homogeneous graph using the users' authentication logs to represent the correlative information. To efficiently learn correlative information, graph embedding is employed to embed the graph into low-dimensional vectors as structural features. Finally, we design an anomaly detection model using capsule neural network to learn statistical features and structural features adequately. According to the experimental results, CapsITD outperforms both traditional machine learning methods and state-of-the-art deep learning methods.

Our contributions can be summarized as follows:

- We construct a homogeneous graph that contains the correlative information from users' authentication logs and use a graph embedding technique to generate structural features which are helpful for user-level insider threat detection.
- We design a deep learning-based anomaly detection model, which can learn the extracted features effectively and achieve improved performance for detecting malicious insiders.
- We evaluate our method using a universal insider threat dataset (CERT version 4.2). The results show that our method is effective, competitive, and able to achieve state-of-the-art performance.

The remaining part of the paper proceeds as follows. Section 2 reviews the related studies. Section 3 presents the proposed methods and explains the learning algorithm. Section 4 covers the experiments and results analysis. Finally, Sect. 5 contains the conclusions.

2 Related Work

2.1 Insider Threat Detection

At present, existing insider threat detection methods can mainly be divided into two categories: signature-based methods and anomaly-based methods [6].

Signature-based method is to design a signature for each known insider threat, match incoming user behavior data with existing signatures, and identify users that match the signatures as insiders. Nguyen et al. [7] built a series of rules for exposing unusual system calls relating to the file system and detecting known abnormal actions effectively. However, the signature-based method heavily relies on domain expertise and can not cope with previously unknown insider threats.

Anomaly-based method is to calculate the deviation of current behavior from normal behavior and identify users that have a large deviation as insiders. Most existing anomaly-based methods are based on machine learning or deep learning and generally build an anomaly detection model based on historical user behavior data. Then use the fitted model to determine whether the user is an insider. Le et al. [3] adopted self-organizing map and C4.5 decision tree to detect malicious insiders using the numerical features extracted from user behavior data. Jiang et al. [4] proposed a graph convolutional network based model to identify users with abnormal behavior. Gayathri et al. [5] employed a pre-trained deep convolutional neural network for anomaly detection to identify malicious insiders. These methods either ignore the correlative information between users or can not learn the extracted features effectively.

2.2 Graph Embedding

Graph embedding aims at learning the representative embeddings as low dimensional vectors for each node in a graph. The embedding vectors represent the structure of nodes and can be used in downstream prediction tasks, such as node classification and link prediction.

Graph embedding technique is also used in the anomaly detection field. Wei et al. [8] used graph embedding to capture comprehensive relationships for detecting anomalous logon activities. Bowman et al. [9] built an authentication graph and used graph embedding to learn latent representations of the authenticating entities. Then, they identified low-probability authentication events to detect anomalous users. The above methods improve the learning performance by using the graph embedding technique.

2.3 Capsule Neural Network

Capsule neural network was first proposed by Hinton [10]. Unlike traditional neural network, capsule neural network uses vector instead of scalar as a neuron in traditional neural network and drops the pooling operation to retain feature spatial information. By this means, the capsule neural network has a momentous

improvement compared to the traditional neural network. The capsule neural network is widely utilised in the field of anomaly detection. For example, Zhang et al. [11] presented a capsule neural network based intrusion detection method. Li et al. [12] employed a capsule neural network to detect anomalous images. They have proved the great power of the capsule neural network for feature learning.

3 Methodology

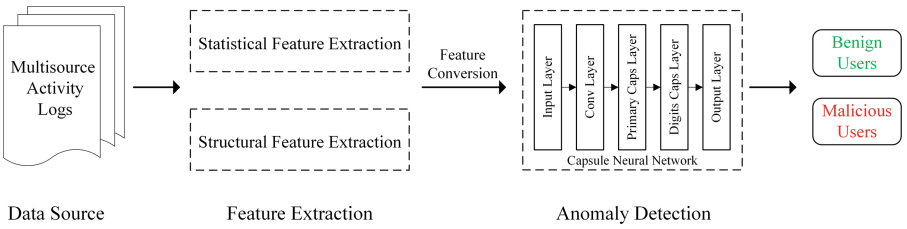


Fig. 1. The framework of CapsITD

Figure 1 illustrates the framework of CapsITD. CapsITD consists of two components: feature extraction module and anomaly detection module.

In the feature extraction module, we first count frequency-based and content-based information about user behavior from users' multisource activity logs as statistical features. Then we construct a homogeneous graph based on users' authentication logs and embed the graph into low-dimensional vectors as structural features. We concatenate and convert statistical features and structural features into the form of feature matrices, which are suitable as input to the neural network for the next module.

In the anomaly detection module, we train the anomaly detection model based on a capsule neural network to detect the users as benign or malicious using the feature matrices generated in the previous module.

3.1 Feature Extraction Module

The goal of the feature extraction module is to collect useful users' characteristics to identify malicious insiders. To obtain more comprehensive and effective features from users' multisource activity logs, as shown in Fig. 2, we extract the features from the statistical aspect and structural aspect respectively.

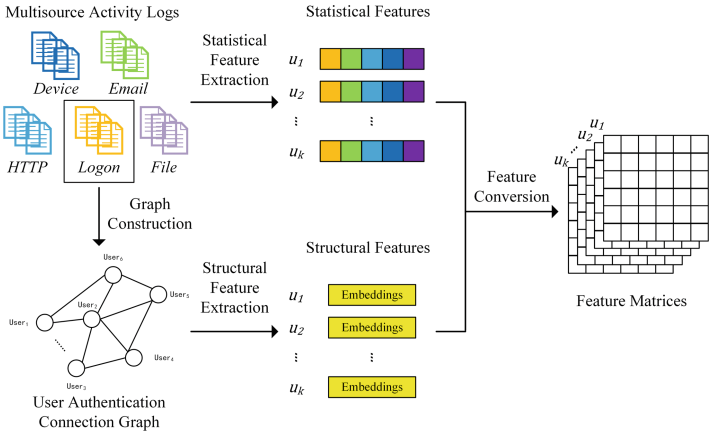


Fig. 2. The process of feature extraction

Statistical Feature Extraction. Statistical features can reflect the behavioral patterns of users and help us identify potential malicious users. In this work, we design 31 statistical features from users' multisource activity logs based on [13,14]. The statistical features can be categorised into frequency-based features and content-based features.

Table 1. Frequency-based features

Data source	Feature name
Logon	Logon/Logoff times, Off-work Logon/Logoff times, Number of PC for Logon/Logoff
Device	Number of Device Connection, Number of Off-work Device Connection, Number of PC for Device Connection
File	Number of Different Files, Number of Total Files, Number of Off-work Files, Number of .exe Files, Number of PC for Files
Email	Number of Sent Emails, Number of Out Organization Emails, Number of In Organization Emails, Average Email Size, Number of Email Attachments, Number of Receivers of Sent Emails, Number of Off-work Sent Emails, Number of PC for Emails
HTTP	Number of Web Pages Browsed, Number of Off-work Web Pages Browsed

On the one hand, we derive frequency-based features from users' daily activities using the frequency information, which can reveal the typical behavioral patterns of users. The frequency-based features are the daily average counts of different types of actions the user performs, such as logon and logoff times, off-work hours logon and logoff times, and the number of PCs for emails. Based on the aggregation of logon and logoff activities, we define the work time as 8:00 to 19:00 [14]. Table 1 lists the detailed frequency-based features.

Table 2. Content-based features

Data source	Feature name
Email	Number of Sentiment-related Emails
HTTP	Number of Wikileak-related, Jobhunting-related, Hacking-related, Cloudstorage-related, Social-related, Sentiment-related Web Pages

On the other hand, we derive content-based features using the contents of users' communication. The content-based features are based on the content of emails and web pages, such as the sentiment tendency of emails [13] and different types of web pages. The detailed content-based features are listed in Table 2.

Structural Feature Extraction. Structural features can reflect the correlative information of users. The correlative information of users is represented by the edges of the graph, and users with similar behaviors tend to be closer together. This information can help us detect malicious insiders. Structural feature extraction can be separated into two steps: graph construction and graph embedding. The first step is to construct a user authentication connection graph based on the users' authentication logs. Then, we use graph embedding to derive latent node representations from the previously constructed graph and embed the nodes into low-dimensional vectors containing the correlative information of users. The low-dimensional vectors generated by graph embedding are used as structural features.

Graph Construction. The user authentication connection graph is defined as a homogeneous graph $G = (V, E)$. This graph has a node type mapping $\phi : V \rightarrow A$ and an edge type mapping $\psi : E \rightarrow R$, where V denotes the node set and E denotes the edge set, $A = \{user\}$ and $R = \{connection\}$. We define two users have a connection only if they log on to the same computer. Specifically, if $user_i$ logs on to the $computer_k$ and $user_j$ logs on to the $computer_k$ too, there is a connection between $user_i$ and $user_j$.

Graph Embedding. Graph embedding is the process of transforming a graph into a low-dimensional space while preserving the graph's information. The goal is to convert each node in the graph G into a low-dimensional vector while retaining relations between nodes.

The process of user authentication connection graph embedding is divided into two steps: First, we sample graph G using fixed-length random walks. Specifically, we explore r fixed-length random walks for any node in the graph $G = (V, E)$ and generate node sequences set $S = \{s_1, s_2, ..., s_m\}$, where the ith random walk sequence is denoted as s_i and the total number of sequences is

denoted as m. Then, we learn a d-dimensioal representation for each unique user u. In this step, the skip-gram model is utilised to acquire the embedding vector of each node over the sequence set S by maximizing the objective function f as Eq. 1.

$$f = \sum_{u \in W} logP(N(u) \mid u) \tag{1}$$

where W is the vocabulary of unique nodes representing users. $N(u)$ is the set of neighborhoods of node $u \in V$. The probability of observing nodes $P(N(u)|u)$ is defined by a softmax unit:

$$P\left(N(u) \mid u\right) = \prod_{n_i \in N(u)} \frac{\exp\left(v'^{T}_{n_i} v_u\right)}{\sum_{w=1}^{|W|} \exp\left(v'^{T}_{w} v_u\right)} \tag{2}$$

where v and v' are two vectors which represent of the node u. And v is the ultimately embedding vector of node u. If two nodes have similar authentication patterns, their embedding vectors will commonly occur together after the convergence of the skip-gram model.

Feature Conversion. Feature conversion is to transform the extracted features into a form that is suitable for the input of the capsule neural network based anomaly detection model. We directly concatenate the statistical feature vectors with structural feature vectors as concatenating features. Then the concatenating features will be normalized by min-max normalization, which limits the value range to $[0, 1]$. The normalization can eliminate the magnitude differences between features and avoid the impact on the detection model. Finally, we convert normalized concatenating features to 6×6 two-dimensional feature matrices as the input of the anomaly detection model.

3.2 Anomaly Detection Module

The goal of the anomaly detection module is to detect anomalous users and identify malicious insiders. A strong learning model is required for learning the extracted features in the previous module. Neural networks generally have excellent feature learning capabilities over other algorithms. The convolutional neural network is one of the most classic algorithms in neural networks. As described in Sect. 2, capsule neural network can make up for the shortage of convolutional neural network. So we choose the capsule neural network as our anomaly detection model. The model architecture is made up of convolutional layer, primary capsule layer, and digit capsule layer as shown in Fig. 3.

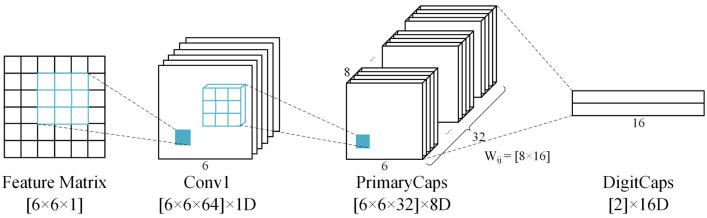


Fig. 3. Structure of capsule neural network

In the convolutional layer(Conv1), Conv1 has 64 channels each make up of 3×3 filters with a stride of 1, a padding of 1, and ReLu activation applied to a $6 \times 6 \times 1$ matrix which represents the concatenating features. Conv1 outputs 64 channel matrices of features with the size 6×6. This layer transforms pixel intensities into local feature detector activity, which is subsequently sent into the primary capsules.

The primary capsule layer is a convolutional capsule layer. This layer is made up of 32 channels of convolutional 8D capsules. Each primary capsule has eight convolutional units, each having a 3×3 kernel and a stride of 1. We execute primary capsule operation on 64 channel matrices of features and generate $6 \times 6 \times 32$ outputs. Each capsule in the 6×6 grid shares its weights with the others.

The third layer is digit capsule layer. This layer contains one 16D capsule for each digit class, and each of these capsules gets input from every capsule in the layer beneath it. As shown in Eq. 3, the overall input to capsule s_j is a weighted sum over all prediction vector $\hat{\mathbf{u}}_{j|i}$ from the capsules in the layer beneath it and the coupling coefficients c_{ij} determined by the iterative dynamic routing process. The $\hat{\mathbf{u}}_{j|i}$ is produced by the Eq. 4, where $\mathbf{W}_{ij}$ is a weight matrix.

$$\mathbf{s}_j = \sum_i c_{ij} \hat{\mathbf{u}}_{j|i} \tag{3}$$

$$\hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij} \mathbf{u}_i \tag{4}$$

The coupling coefficients c_{ij} is produced by the Eq. 5, where b_{ij} is the log prior probabilities that capsule i coupled to capsule j.

$$c_{ij} = \frac{\exp\left(b_{ij}\right)}{\sum_k \exp\left(b_{ik}\right)} \tag{5}$$

The probability of an entity is represented by the length of the output vector of a capsule. Thus, we utilise a non-linear squashing function to ensure that short vectors get shrunk to almost zero length and long vectors get shrunk to a length slightly below 1. Equation 6 shows the squashing function, where $\mathbf{v}_j$ is the capsule j 's vector output and $\mathbf{s}_j$ is its overall input.

$$\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1 + \|\mathbf{s}_j\|^2} \frac{\mathbf{s}_j}{\|\mathbf{s}_j\|} \tag{6}$$

To update the parameters of the entire network, we use the margin loss function L_k for each digit capsule, k:

$$L_k = T_k \max\left(0, m^+ - \|\mathbf{v}_k\|\right)^2 + \lambda\left(1 - T_k\right)\max\left(0, \|\mathbf{v}_k\| - m^-\right)^2 \tag{7}$$

where $T_k = 1$ if the class k is present, otherwise $T_k = 0$. The two hyperparameters, m^+ and m^- are set to 0.9 and 0.1. λ is a regularization parameter and is set to 0.5. To achieve better accuracy, we add reconstruct loss into the final loss as well.

4 Experiments and Results

This section is devoted to evaluating the performance of CapsITD. First, we introduce the dataset utilised for evaluation. Then, we describe the evaluation metrics and experiment setup specifically. The experiment results are then discussed and compared to three classical machine learning algorithms (logistic regression, support vector machine, and random forest) as well as two deep learning algorithms (convolutional neural network and graph convolutional network [4]). We also set a comparative experiment between statistical features and concatenated features to evaluate the effort of correlative information and test the influence of different anomalous sample ratios to verify the robustness of CapsITD.

4.1 Dataset

We use a universal insider threat dataset (CERT version 4.2) from Carnegie Mellon University [15]. This dataset consists of multisource activity logs of 1000 users and 1003 computers over a period of 17 months from January 2010 to May 2011. The multisource activity logs contain user login, removable device usage, file access, email communication, and web browsing history. The dataset covers 70 malicious insiders and three insider threat scenarios, which are data exfiltration, intellectual property theft, and IT sabotage.

4.2 Evaluation Metrics

In the experiments, four well-known metrics are adopted for our evaluation: Accuracy, F-measure, AUC (Area Under the ROC Curve), and Recall. Accuracy is the ratio of correctly predicted observations to total observations. F-measure is the weighted average of precision and recall. AUC is a metric for evaluating the pros and cons of a binary classification model. Recall is the ratio of correctly predicted positive observations to all observations in the true class.

4.3 Experiment Setup

Our experiments are performed on a PC with Intel Core i5-10500 CPU @ 3.10GHz, 16GB RAM, and 64-bit Windows 10 Professional OS. We use Sklearn 0.24.1 to implement the machine learning algorithm and PyTorch 1.7.1 to implement the deep learning algorithm.

In the CERT dataset, we process the multisource activity logs of users and derive statistical features from these logs as demonstrated in Sect. 3. The dimension of statistical features is 31. Then we construct the user authentication connection graph using 1000 users' authentication logs and embed the nodes into low-dimensional vectors. The hyperparameters of graph embedding are as follows: We set the walk length $l = 30$ and the number of walks per node $r = 200$ in random walk. We set the vector's dimension $d = 5$ and the window size to 10 in skip-gram. After that, we concatenate and transform the extracted features into a 6×6 feature matrix as the input to the capsule neural network for each user. The hyperparameters of the capsule neural network are set as follows: The training epoch is set to 50, and the learning rate is set at 0.001.

4.4 Experimental Results

To evaluate the proposed method comprehensively, we first compare CapsITD with three classical machine learning methods and two deep learning methods. Then we expand the experiment to evaluate the effectiveness of correlative information and the robustness of CapsITD.

Table 3. Comparison results with other methods

Method	Accuracy	F-measure	AUC	Recall
LR	0.910	0.794	0.755	0.533
SVM	0.920	0.811	0.761	0.533
RF	0.920	0.803	0.747	0.500
CNN	0.965	0.926	0.897	0.800
GCN	0.945	–	–	0.833
CapsITD	**0.980**	**0.958**	**0.933**	**0.867**

Comparison with Other Methods. We compare the experimental results of the CapsITD with three classical machine learning algorithms such as Logistic Regression (LR), Support Vector Machine (SVM), Random Forest (RF), and two deep learning algorithms such as Convolutional Neural Network (CNN), and Graph Convolutional Network (GCN) [4]. However, since the GCN method does not describe its algorithm in detail, we only refer to the detection results of the CERT dataset as shown in [4] and set up the same dataset partition. The training set and test set are selected the same way as the [4]. The training set contains

160 normal users and 40 abnormal users, and the test set contains 170 normal users and 30 abnormal users. The comparison results between CapsITD and other methods on the CERT dataset are shown in Table 3. We can observe that CapsITD outperforms the other five competing algorithms. The deep learning algorithms outperform the other three machine learning algorithms. Compared with the GCN model proposed by [4], CapsITD is higher 3.5% accuracy and 3.4% recall than the GCN model. CapsITD is competent for insider threat detection since the performance of CapsITD outperforms other existing models.

Table 4. Comparison between statistical features and concatenated features

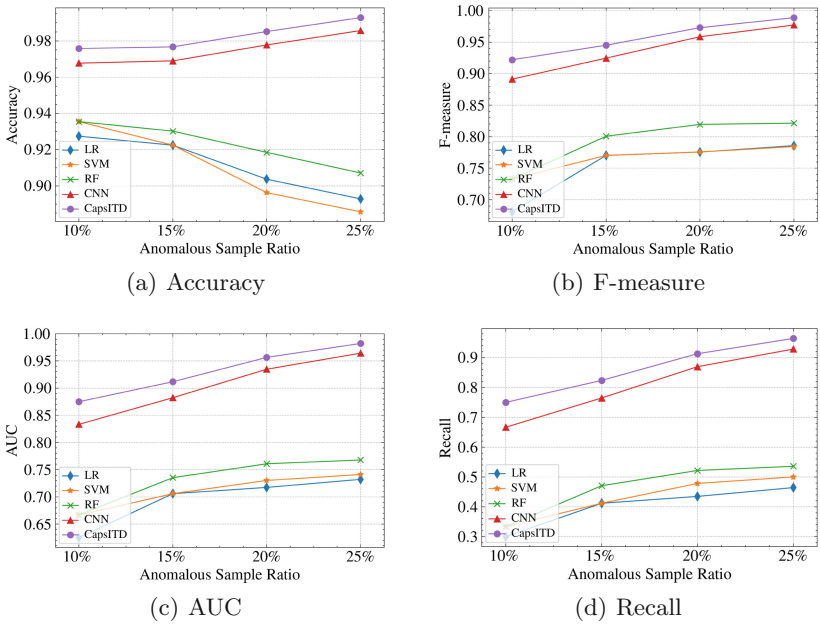
Method	Accuracy	F-measure	AUC	Recall
LR_{stat}	0.895	0.764	0.732	0.500
LR_{conc}	**0.910**	**0.794**	**0.755**	**0.533**
SVM_{stat}	0.905	0.779	0.738	0.500
SVM_{conc}	**0.920**	**0.811**	**0.761**	**0.533**
RF_{stat}	0.895	0.715	0.664	0.333
RF_{conc}	**0.920**	**0.803**	**0.747**	**0.500**
CNN_{stat}	0.950	0.889	0.847	0.700
CNN_{conc}	**0.965**	**0.926**	**0.897**	**0.800**
$CapsITD_{stat}$	0.970	0.936	0.900	0.800
$CapsITD_{conc}$	**0.980**	**0.958**	**0.933**	**0.867**

The Effectiveness of Correlative Information. To evaluate the effectiveness of correlative information, we compare the models with statistical features and concatenated features. The experimental results are shown in Table 4, where the subscripts are *stat* for only statistical features and *conc* for concatenated features. The concatenated features, which incorporate correlative information, are improved in all three machine learning methods and two deep learning methods compared to only statistical features. The most significant improvement is the random forest, where the random forest model using concatenated features improves accuracy by 2.5% compared to the random forest model using only statistical features. CapsITD with concatenated features outperforms CapsITD with only statistical features in terms of 1% accuracy, 2.2% F-measure, 3.3% AUC, and 6.7% recall. In general, correlative information can further improve the performance of detecting malicious insiders.

Table 5. Different anomalous sample ratios

Anomalous sample ratio	Normal	Abnormal
10%	280	28
15%	280	42
20%	280	56
25%	280	70

Performance on Different Anomalous Sample Ratios. To evaluate the robustness of CapsITD, we compare the models with different anomalous sample ratios. Table 5 shows the data with varying ratios of anomalous samples. Since the total number of anomalous samples in the dataset is 70, we set the number of normal samples at 280 to ensure that different anomalous sample ratios can be selected. In this experiment, we test the models on different ratios of anomalous samples. The proportion of training set to test set is 6:4.



(a) Accuracy (b) F-measure

(c) AUC (d) Recall

Fig. 4. Performance on different anomalous sample ratios

As shown in Fig. 4, they are the performance of three machine learning algorithms: Logistic Regression (LR), Support Vector Machine (SVM), Random Forest (RF), and two deep learning algorithms: Convolutional Neural Networks (CNN) and Our Proposed Method (CapsITD). We make the following observations: CapsITD is always higher than CNN of the four metrics. And deep

learning-based approaches have significant improvements over machine learning-based approaches. According to Fig. 4(d), the ability of all models to detect malicious insiders increases as the ratio of anomalous samples rises. However, Fig. 4(a) shows the accuracy of machine learning-based approaches decreases as the ratio of anomalous samples rises. This is due to the poor ability to identify anomalous samples by machine learning-based approaches.

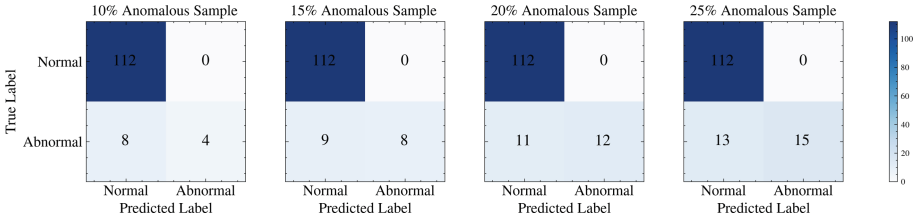


Fig. 5. The confusion matrix of RF on different anomolous sample ratios

As shown in Fig. 5, they are the confusion matrices of the random forest model under different anomalous sample ratios. It can be seen that the random forest can detect all normal samples under different anomalous sample ratios, but has a poor ability to identify anomalous samples. As the ratio of anomalous samples increases, the random forest model can gradually distinguish anomalous samples, but the accuracy tends to slightly decrease due to the large base of normal samples. The situation for the other machine learning-based approaches is the same as random forest. It demonstrates that machine learning-based approaches are unable to learn from the extracted features to classify normal and abnormal samples effectively. Meanwhile, deep learning-based approaches show an increasing trend in all metrics as the ratio of anomalous samples rises. It means that deep learning-based approaches can learn more from the extracted features to classify normal and abnormal samples precisely. In summary, CapsITD still outperforms other methods on different anomalous sample ratios. This experiment shows the robustness of CapsITD.

5 Conclusion

In this paper, we present CapsITD, a novel user-level approach for insider threat detection. This approach can effectively detect malicious insiders from users' multisource activity logs. Firstly, we extract statistical features based on users' daily activities and communications. Secondly, we construct a homogeneous graph based on users' authentication logs. The constructed graph can represent the correlative information between users. We then use a graph embedding technique to embed the graph into low-dimensional vectors as structural features. Thirdly, we design an anomaly detection model using capsule neural network to learn statistical features and structural features adequately. The results of

the comparative experiments reveal that our proposed approach has superior performance compared with other existing approaches. In our expansive experiments, we verified the effectiveness of correlative information and the robustness of CapsITD.

Acknowledgment. This work is supported by National Key Research and Development Program of China (No.2021YFF0307203, No.2019QY1300), and NSFC (No. 61902376), Youth Innovation Promotion Association CAS (No.2021156), the Strategic Priority Research Program of Chinese Academy of Sciences (No. XDC02040100). This work is also supported by the Program of Key Laboratory of Network Assessment Technology, the Chinese Academy of Sciences, Program of Beijing Key Laboratory of Network Security and Protection Technology.

References

1. 2020 Securonix Insider Threat Report. https://www.securonix.com/resources/2020-insider-threat-report/. (Accessed 29 Dec 2021)
2. 2021 Insider threat report. https://gurucul.com/2021-insider-threat-report. (Accessed 29 Dec 2021)
3. Le, D.C., Zincir-Heywood, A.N.: Evaluating insider threat detection workflow using supervised and unsupervised learning. In: 2018 IEEE Security and Privacy Workshops (SPW), pp. 270–275. IEEE (2018)
4. Jiang, J., et al.: Anomaly detection with graph convolutional networks for insider threat and fraud detection. In: MILCOM 2019–2019 IEEE Military Communications Conference (MILCOM), pp. 109–114. IEEE (2019)
5. Gayathri, R., Sajjanhar, A., Xiang, Y.: Image-based feature representation for insider threat classification. Appl. Sci. **10**(14), 4945 (2020)
6. Liu, L., De Vel, O., Han, Q.L., Zhang, J., Xiang, Y.: Detecting and preventing cyber insider threats: A survey. IEEE Commun. Surv. Tutorials **20**(2), 1397–1417 (2018)
7. Nguyen, N., Reiher, P., Kuenning, G.H.: Detecting insider threats by monitoring system call activity. In: IEEE Systems, Man and Cybernetics Society Information Assurance Workshop, vol. 2003, pp. 45–52. IEEE (2003)
8. Wei, R., Cai, L., Yu, A., Meng, D.: Age: authentication graph embedding for detecting anomalous login activities. In: Zhou, J., Luo, X., Shen, Q., Xu, Z. (eds.) ICICS 2019. LNCS, vol. 11999, pp. 341–356. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-41579-2_20
9. Bowman, B., Laprade, C., Ji, Y., Huang, H.H.: Detecting lateral movement in enterprise computer networks with unsupervised graph ai. In: 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020), pp. 257–268 (2020)
10. Hinton, G.E., Krizhevsky, A., Wang, S.D.: Transforming auto-encoders. In: Honkela, T., Duch, W., Girolami, M., Kaski, S. (eds.) ICANN 2011. LNCS, vol. 6791, pp. 44–51. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21735-7_6
11. Zhang, X., Yin, S.: Intrusion detection model of random attention capsule network based on variable fusion. J. Commun. **41**(11), 160 (2020)
12. Li, X.: Anomaly Detection Based on Disentangled Representation Learning. Ph.D. thesis, Université d'Ottawa/University of Ottawa (2020)

13. Jiang, J., et al.: Prediction and detection of malicious insiders' motivation based on sentiment profile on webpages and emails. In: MILCOM 2018–2018 IEEE Military Communications Conference (MILCOM), pp. 1–6. IEEE (2018)
14. Chattopadhyay, P., Wang, L., Tan, Y.P.: Scenario-based insider threat detection from cyber activities. IEEE Trans. Comput. Soc. Syst. **5**(3), 660–675 (2018)
15. Glasser, J., Lindauer, B.: Bridging the gap: A pragmatic approach to generating insider threat data. In: 2013 IEEE Security and Privacy Workshops, pp. 98–104. IEEE (2013)