A graph empowered insider threat detection framework based on daily activities

ISA Transactions 141 (2023) 84–92

Contents lists available at ScienceDirect

ISA Transactions

journal homepage: www.elsevier.com/locate/isatrans

Research article

A graph empowered insider threat detection framework based on daily activities

Wei Hong [a], Jiao Yin [b,*], Mingshan You [b], Hua Wang [b], Jinli Cao [c], Jianxin Li [d], Ming Liu [d], Chengyuan Man [e]

[a] School of Artificial Intelligence, Chongqing University of Arts and Sciences, Chongqing, 402160, China
[b] Institute for Sustainable Industries and Liveable Cities, Victoria University, Melbourne, VIC, 3011, Australia
[c] Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC, 3086, Australia
[d] School of Information Technology, Deakin University, Melbourne, VIC, 3125, Australia
[e] Async Working Pty Ltd, Melbourne, VIC, 3149, Australia

ARTICLE INFO

Article history:
Received 30 November 2022
Received in revised form 27 June 2023
Accepted 28 June 2023
Available online 4 July 2023

Keywords:
Sequential activity
Graph neural networks
Insider threat
LSTM auto-encoder

ABSTRACT

While threats from outsiders are easier to alleviate, effective ways seldom exist to handle threats from insiders. The key to managing insider threats lies in engineering behavioral features efficiently and classifying them correctly. To handle challenges in feature engineering, we propose an integrated feature engineering solution based on daily activities, combining manually-selected features and automatically-extracted features together. Particularly, an LSTM auto-encoder is introduced for automatic feature engineering from sequential activities. To improve detection, a residual hybrid network (ResHybnet) containing GNN and CNN components is also proposed along with an organizational graph, taking a user-day combination as a node. Experimental results show that the proposed LSTM auto-encoder could extract hidden patterns from sequential activities efficiently, improving F1 score by 0.56%. Additionally, with the designed residual link, our ResHybnet model works well to boost performance and has outperformed the best of other models by 1.97% on the same features. We published our code on GitHub: https://github.com/Wayne-on-the-road/ResHybnet.

1. Introduction

Since the information boom from the early 90 s of last century, cybersecurity has been a rising concern all the way to nowadays [1–3]. According to one of the latest cyber-security reports from Accenture in 2021,[1] the cyber attacks experienced by each respondent increased 31% compared with previous year, reaching an eye-catching high number of 270 times. Out of all those cyber attacks, although threats from insiders account for only a small portion of the total, organizations generally consider them to pose far more risks to their information security [4].

According to the seventh edition of the CERT Common Sense Guide to Mitigating Insider Threats published by Carnegie Mellon University [5], insider refers to an individual who has or had authorized access to an organization's critical assets. Therefore, on the one hand, once insiders initiate attacks, they could bring significant losses for the organization due to their access to essential assets. On the other hand, due to legitimate authorization, insider threats are often hard to contain, even with comprehensive access control solutions [6–8].

To mitigate the threats posed by insiders, the key is to find hidden patterns in an insider's malicious behavior and distinguish them from benign ones. Typically, domain experts would first preset specific rules for feature extraction, and then craft suitable classification models to discover malicious activities [9]. Since different domain experts see individual behavior from different perspectives, the feature engineering approach often varies depending on how the problem is defined. At the same time, applied detection methods have also evolved from simple statistical approaches to machine learning ones such as linear regression (LR), Gaussian naive Bayes (GNB), support vector machine (SVM), and all the way to deep learning approaches such as convolutional neural networks (CNNs) [10–12].

Regarding activity feature engineering, although domain knowledge could help to select key features efficiently, they are more related to the standalone properties of certain activities [13]. Some latent features in the sequential order of activities could still exist during a specific time interval, and it has the potential to improve detection performance [14].

* Corresponding author.
E-mail address: jiao.yin@vu.edu.au (J. Yin).

[1] https://www.accenture.com/us-en/insights/security/invest-cyber-resilience

https://doi.org/10.1016/j.isatra.2023.06.030

In terms of classification models, on the one hand, traditional models such as artificial neural networks (ANNs) and convolutional neural networks (CNNs) only see user behavior in an isolated way, trying to discover the patterns behind a large amount of individual data separately [15–17]. On the other hand, when taking the approach of graph intelligence such as graph neural networks (GNNs), individual characteristics would suffer downgrade to a certain degree due to the over-emphasized connections between individuals [18–20].

In order to offer more solutions in insider threat detection regarding feature engineering, and considering that an employee's workload would most likely show pattern according to calendar day, we first propose a daily activity based feature engineering approach. Furthermore, to better discover the behavior similarity between employees from organizational connection, we propose a residual hybrid network (ResHybnet) model that can combine GNN model and CNN model together.

Specifically, for daily activity feature engineering, we first employ domain knowledge to manually select features from standalone activities, and then use a long short-term memory (LSTM) model based auto-encoder to extract features from sequential activities automatically. After combining extracted features, we feed them into the proposed ResHybnet, and along with the information from constructed organizational graph, the insiders conducted malicious behaviors during specific day could be found out.

To summarize, this paper makes contributions from three aspects listed below:

- We proposed an integrated feature engineering method based on user's daily behaviors. Firstly, we manually extract domain knowledge-based features from standalone activities. Secondly, we define sequential activities on a daily basis and use an LSTM auto-encoder model to extract latent features from sequential activities automatically. Experimental results show that the features extracted from sequential activity indeed can improve the detection performance in the final stage.
- We designed a residual hybrid network (ResHybnet) model consisting of GNN and CNN layers, and a residual link from node attributes to the CNN component to enhance the performance of insider threat detection. ResHybnet achieves the best results for the following three reasons. Firstly, the GNN component can discover the topological connections of daily behavior (represented by node attributes) from constructed organizational graph. Secondly, the residual link can compensate node attributes that have been weakened by the GNN component during propagation. Lastly, the CNN component is a sound model for pattern discovering in non-topological context. When integrated together with designed residual link, the best part of two classical models could be made use of.
- We conducted a comparative study to see the impact of length choice on the final detection performance to better understand how the input sequence length of LSTM auto-encoder will affect feature engineering of sequential activities. Experimental results show longer sequence setting for LSTM auto-encoder facilitate better performance, and the statistics of sequence length could contribute to proper length selection.

In the rest of this paper, we will first review the related works on insider threat detection and the corresponding feature engineering approaches in Section 2, followed by Section 3 consisting of framework overview and LSTM auto-encoder architecture introduction. Section 4 implement the proposed methodology on an open-source insider threat dataset as a use case. Section 5 presents experiment results on detection performance and ablation study findings related to sequential activity feature engineering. Section 6 concludes the paper with limitation discussion and future works.

2. Related works

2.1. Insider threat detection

For long, the terms insider and insider threat have never been easy to define. Nowadays, the most widely recognized definitions in this field are dynamically updated by the CERT research team of the Software Engineering Institute at Carnegie Mellon University [5].

Previous research works on insider threat detection could be categorized mainly into two streams: conceptual works, which focus on the methodology and theories about insider threat detection, and operational works, which emphasize practical solutions related to certain datasets. [21].

In the operational research fields that relate most to our work, Schonlau, M. et al. [22] adopted a statistical approach to detect insider threats and introduced a command history-based dataset SEA as early as in 2001. Garg, A. et al. [23] then introduced a new dataset based on mouse operation history, putting insider detection problem in a graphical user interface (GUI) context. This work also employed SVM method, which achieves an accuracy rate up to 96% high with few false positives.

As time went by, more comprehensive approaches were introduced to insider threat research thanks to the increasing availability of richer data, such as synthetic dataset CERT [24]. In 2015, Gavai, G. et al. [25] also presented a real-world dataset containing enterprise social and online activity with intentionally injected insider threat events. They modified the isolation forest algorithm and tested it on the dataset based on supervised and unsupervised settings.

Some researchers also experimented graph-based methods in insider threat detection research. For example, a novel organizational graph was created in the paper [26], and then node degree and betweenness centrality are introduced to practice clustering techniques. Outliers in the clustering process are considered malicious insiders. Jiang, J. et al. in paper [27] introduced a GCN model to detect malicious insides. With the calculated similarity of different users inside an organization, they constructed a weighted graph to facilitate better performance. The mentioned two works demonstrate graph neural networks is promising in the research field of insider threat. However, while GNN-based classification models are good at discovering the connections in graph structures, they also may cause the downgrading of node attributes, as found by previous researchers in [19].

To the best of our knowledge, previous work has yet to try to address the node attributes downgrading problem and integrate GNN and CNN layers into one model in insider threat detection research.

2.2. Insider threat feature engineering

To discover malicious insiders, feature engineering often plays an important role due to the fact that human behavior is complex [28,29]. As early as 2005, Liu et al. [30] investigated three feature representation ways on system calls and found that parameter-based features for certain system calls indeed were sensitive to some extent for detecting the insider threat. In the paper [31], the authors used a deep belief network(DBN) to extract hidden features from users' behavior logs and showed its effectiveness in insider threat feature engineering.

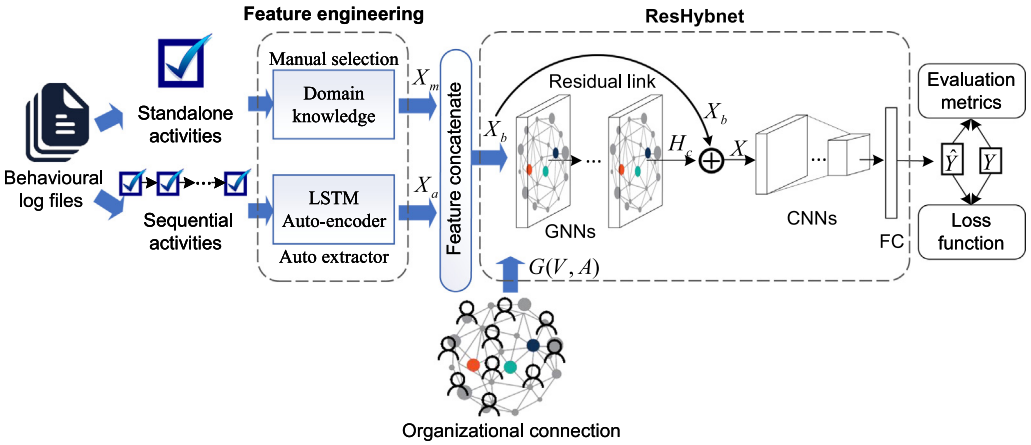

Fig. 1. Daily activity based insider threat detection framework.

Chattopadhyay et al. [32] saw the potential of feature engineering from a time series perspective. From user activity logs, they constructed a time series feature vector based on statistics of single-day features over a period of time. Singh et al. [33] applied isometric feature mapping (ISOMAP) for feature extraction and Emperor Penguin Algorithm for optimal feature selection. Although temporal nature was considered in feature extraction, those two works still focused on the statistical side of feature engineering.

To discover the behavioral pattern of a user in a certain day, Yuan, F. et al. [34] designed a framework to extract fixed-size features from activity sequence of one whole day using the LSTM model and applied a CNN classifier to distinguish insiders. Another later work [35] applied LSTM auto-encoder as a classifier based on the reconstructed loss to do detection rather than feature extraction.

Based on our extensive review, no previous work has extracted latent features from sequential activities using an LSTM auto-encoder model under user-day problem setting.

3. Methodology

In previous works, researchers defined insider detection problems in different ways. For example, paper [26,34] aim to find out insiders by examining users' activities across a time window more than one year long, and considering all the activities as a whole, while Liu et al. [36] define insider detection problem as to find out malicious activities. Those two approaches represent the majority problem settings for insider threat detection.

Considering that an employee in an organization usually has a working load pattern following the calendar day, we define our insider threat detection task as finding out malicious user-days. This is also because an insider could behave maliciously on certain days and otherwise on the rest days, and for cybersecurity analysts, monitoring risk level on a daily basis often echoes with the typical operation pattern of an organization.

Based on this problem formation, we lay out the whole solution for insider threat detection in this section, starting with a framework overview, followed by a detailed description of the proposed integrated feature engineering method and the residual hybrid network model. In regard to the implementation of this solution, we present the whole process in Section 4, and apply it on the CERT 4.2 dataset

3.1. Framework overview

The proposed daily activity-based insider threat detection framework is shown in Fig. 1. Overall, the behavioral log files are first pre-processed to generate daily activities, which can be categorized into standalone activities and sequential activities. Then, we apply an integrated feature engineering approach to extract features from them separately. At the same time, a graph is constructed to represent the organizational connection between different user-days. After combining the two feature groups together, we use a special-designed residual hybrid network model to perform insider threat detection with the help of graph information. The detailed processes are described in the following sections.

3.2. Integrated feature engineering

For malicious insiders, we believe the subtle difference in their behavior could be revealed from two aspects. First, the standalone activities in a daily schedule, such as the first logon time, could be an indicator of a threat if it significantly deviates from normal ones. Second, within a single working day, the activities conducted by an employee would happen in time series, which naturally forms a sequence of activities. If an activity sequence shows a different pattern from others, it could also serve as a good indicator for insider threat detection. Therefore, in the framework shown in Fig. 1, we start with feature engineering regarding those two kinds of activities.

3.2.1. Manual engineering

For each user-day, features of standalone activities are selected based on domain knowledge. As a result of this manual selection, the output feature matrix for all user-days from this stream is denoted as X_m ($X_m \in \mathbb{R}^{n\times d_m}$), among which n represents the total of user-day combinations, and d_m stands for the number of manually selected features for each user-day. Regarding a certain user-day, the vector for manual features could be denoted as $x_m^{(i)} \in \mathbb{R}^{d_m}$ ($i \in \{1, 2, \cdots, n\}$).

3.2.2. Automatic engineering

As for sequential activities, we designed an auto-encoder based on a widely used LSTM model to extract useful features automatically.

On the one hand, to use the LSTM auto-encoder, a fixed input sequence length L needs to be set and then the actual sequence will be trimmed or filled to the length L. After numbering and one-hot encoding, each daily activity sequence is encoded as $S^{(i)} = \{s_1, s_2, \cdots, s_L\}$, where $S^{(i)} \in \mathbb{R}^{L\times d_s}$ ($i \in \{1, 2, \cdots, n\}$), and d_s stands for the dimension of an encoded activity (depending on how many unique types of activity are defined). This $S^{(i)}$ sequence serves as the input for our auto-encoder.

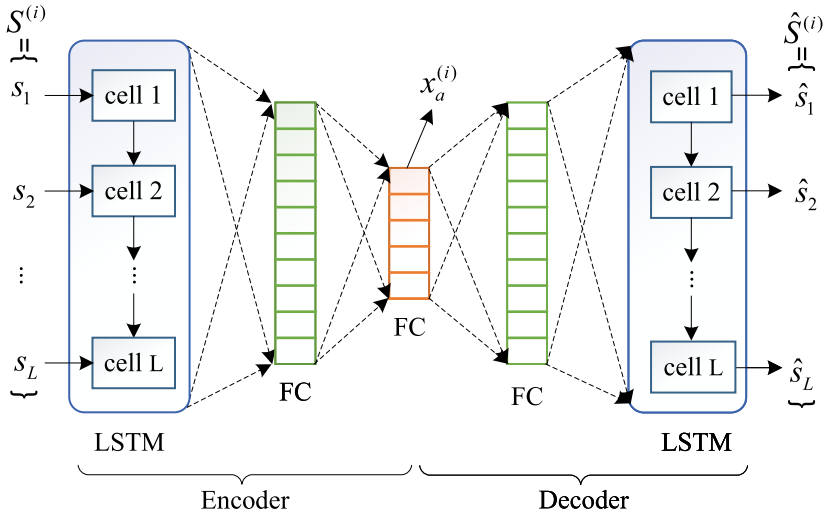

Fig. 2. LSTM auto-encoder for sequential activity feature extraction.

On the other hand, as shown in Fig. 2, the LSTM auto-encoder is a symmetrical network consisting of two LSTM layers with L cells for each and three fully connected (FC) layers. The first half of the LSTM auto-encoder is the encoder part, and the rest is the decoder part.

In the training process, each sequence $S^{(i)} = \{s_1, s_2, \cdots, s_L\}$ will be fed into the LSTM auto-encoder, and the output $\hat{S}_i = \{\hat{s}_1, \hat{s}_2, \cdots, \hat{s}_L\}$ from the decoder side will be compared with the original input sequence. By minimizing the differences between S_i and $\hat{S}^{(i)}$, we could find the best parameters of the LSTM auto-encoder for feature extraction.

After training, for each sequence $S^{(i)}$, the extracted feature from the output of the encoder part will be denoted as $x_a^{(i)} \in \mathbb{R}^{d_a}$ $(i \in \{1, 2, \cdots, n\})$, since those features are automatically generated from the trained encoder, where the d_a stands for the dimension of automatic features.

By putting automatic features from all user-days together, we will have the automatic feature matrix of sequential activities, which can be denoted as X_a, where $X_a \in \mathbb{R}^{n \times d_a}$, n is the total number of user-days.

3.2.3. Feature integration

In the final stage of feature engineering, features extracted from the manual and automatic ways are concatenated as the final behavioral feature matrix, as shown in Eq. (1).

$$X_b = concatenate(X_m, X_a), \tag{1}$$

where $X_b \in \mathbb{R}^{n \times (d_m + d_a)}$ stands for the final behavioral feature matrix of all user-days, and each feature vector has a dimension of $d_m + d_a$. For the ith user-day, the behavioral features could be represented as $x_b^{(i)} \in \mathbb{R}^{\{d_m + d_a\}}$, where $i \in \{1, 2, \cdots, n\}$.

3.3. Residual hybrid network

Behavioral features X_b are successfully extracted from users' behavioral log files at the final stage of the integrated feature engineering. For insider threat detection, it will be sent into a special-designed Residual hybrid network model consisting of GNN, CNN, and FC layers, and a residual link, as shown in Fig. 1.

To apply the residual hybrid network model, a graph is first constructed following paper [19], and denoted as $G = \{\mathcal{V}, A\}$, where $\mathcal{V}$ stands for all the vertex of the graph, A stands for the adjacency matrix of the graph, which is derived from the organizational relationship between users. For any user-day, it could be represented as a node in the graph $(v \in \mathcal{V})$.

As shown in Fig. 1, the proposed ResHybnet will take behavioral feature matrix X_b and user-day organizational graph $G(V, A)$ as inputs, and performs the final insider threat detection task.

The process of this detection can be described in Eq. (2), (3), (4).

$$H_c = f_{gnns}(X_b, V, A, \Theta_{gnns}), \tag{2}$$

$$X = H_c + X_b, \tag{3}$$

$$\hat{Y} = f_{fc}(f_{cnns}(X, \Theta_{cnns}), \Theta_{fc}), \tag{4}$$

where f_{gnns}, f_{cnns} and f_{fc} are the mapping functions of the GNN, CNN and FC layers shown in Fig. 1; Θ_{gnns}, Θ_{cnns} and Θ_{fc} are the corresponding model parameters, which can be learned in an end-to-end manner.

As shown in Eq. (2), in the first step of ResHybnet, the GNN component will take the behavioral feature matrix X_b and graph $G(V, A)$ as inputs to embed the topological connections of user-days. At the same time, a residual link from X_b will send the original node attributes to the output side of the GNN component for compensation purposes, as shown in Eq. (3). After adding X_b back to GNN output H_c to generate X, the compensated node feature, X, will be further processed by the CNN component, followed by a fully connected layer to yield the final insider threat detection result from the ResHybnet model, denoted as $\hat{Y}$, as shown in Eq. (4). At the last stage, the effectiveness of the proposed framework is evaluated through the comparison between the predicted results $\hat{Y}$ and the true labels Y.

4. A use case

For a better illustration of the implementation process, we applied it to the CERT4.2 dataset for insider threat research [24], which helps to avoid privacy concerns due to its synthetic nature [37,38]. We constructed a balanced sample dataset using the down-sampling technique. This section presents details on how to construct the sample graph, conduct feature engineering and perform the insider threat detection task.

4.1. Sample organizational graph

For CERT 4.2, there are 70 malicious insiders out of 1000 employees, and the raw dataset is organized by activity categories, which contain logs of activities such as logging on/off a personal

Table 1
File content description for CERT4.2 dataset.

File	Content
LDAP files	Describe organizational information of the enterprise, such as supervisor name, department name .etc
Device	Describe the activities related to USB device, such as connect or disconnect a removable device
Email	Describe all email activities including content in the form of key words.
File	Describe the activities related to file usages on a PC, including file type and content summary.
Http	Describe the web usage information from a PC, including url and key words of content
Logon	Describe the logon/logoff activities on a PC
Psychometric	Describe the personality evaluation result for all employees using big five OCEAN model

computer (PC), opening a file, using a USB drive, etc. Additionally, the organizational relations are also given by LDAP file to elaborate employee's supervisor and department. In Table 1, key information of CERT 4.2 dataset files is briefly described.

We first randomly down-sampled a balanced data set containing 1908 user-day samples, mapping them into 1908 nodes in an organizational graph by several rules:

- First: If user B supervises user A, then every user-day from A will be connected with every user-day from B.
- Second: If user A shares the same supervisor with user B, then every user-day from A will be connected with every user-day from B.
- Third: All the user-days that belong to the same user will be connected with each other.

After constructing this organizational graph, 70% of the whole sample dataset are selected out for training while the rest 30% for testing. Facilitated by this graph, behavior similarity could potentially be discovered by deep learning models.

4.2. Daily activity feature engineering

4.2.1. Manual feature engineering based on domain knowledge

For manual feature engineering, it is largely relied on domain knowledge to select the potential indicators. In this paper, we studied two types of activity file for this feature group, namely, device.csv file and logon.csv file, and manually selected five features that have the potential to be good indicators.

To be specific, the five manual features for standalone activities are shown in Table 2.

4.2.2. Automatic feature engineering based on LSTM auto-encoder

For automatic feature engineering, we need to encode the daily activities first and join them into a sequence based on the time each activity was performed. We referenced the practice adopted by paper [34] and analyzed all the activities logged in file.csv, logon.csv, device.csv, and email.csv. The rules for encoding those activities are shown in Table 3.

In total, we defined 12 different types of activities and encoded them further into 24 types of activities based on whether it is conducted during work time or off-work time, numbering them from 1 to 24. Our sample dataset's longest daily activity sequence is 74, which serves as our sequence length setting when training the LSTM auto-encoder for feature engineering. For those user-days that have shorter sequences, we add 0 to represent 'none' activity at the end of sequence until it reaches the sequence length setting 74. Before feeding the sequence into the LSTM auto-encoder for training, we further did one-hot encoding for the 24 types of activities plus the 'none' type. Therefore, for the LSTM auto-encoder, the input sequence length is 74, and the input dimension is 25.

In the training part, we take all the sample datasets as input and try to minimize the training loss with an early stopping strategy. After multiple training rounds, we use the encoder output from the middle full connection layer as our automatic features for sequential activities and concatenate it with manual features extracted in Section 4.2.1. For this implementation, we set the encoder output dimension to be 5, so for the final feature matrix, the dimension would be 1908X10, and for each sample node, the feature vector dimension is 10.

4.3. Integrated insider detection with ResHybnet

As illustrated in Section 3.1, the concatenated feature matrix will serve as the node attribute matrix for nodes in the sample graph. With the help of the data loader tool in PyTorch, node attributes data and graph connection data (edge information) will be fed into our ResHybnet model for a supervised insider threat detection task.

During this process, the daily activity pattern would be mined by the GNN and CNN component in our model, where GNN focuses more on finding the behavior similarity between different user-day nodes that have organizational connections, and CNN focuses more on discovering the behavior pattern for each individual node.

5. Experiments

In order to verify the effectiveness of the proposed solution, we conducted our experiment mainly from two aspects. In the first part, we hold the feature engineering part unchanged and investigate the detection performance of our ResHybnet model. In the second part, we apply the best setting of our model and investigate how the different feature groups have contributed.

We performed all experiments with Python language. Specifically, PyTorch package[2] is used for CNN model constructing, scikit-learn[3] package is adopted for binary classifier implementation, and PyTorch Geometric[4] is used for implementing GNN models. Default settings are used for all ready-to-go models and classifiers unless otherwise specified.

For the training process, we employed an early stop strategy, setting the waiting epoch number to be 30, and monitored the test loss value with 0.0001 as delta threshold, and each experiment will be carried out for 10 rounds to report the average. This setting is universally applied to all training processes.

5.1. Performance study of the proposed model

5.1.1. Performance comparison with other classifiers

Since ResHybnet is a model-independent structure that can make the best out of GNN and CNN components, in experimental part, we first chose the most simple and popular GNN and CNN structures to verify the effectiveness of our model, and compared it with commonly used binary classifier such as SVM, LR, RF, GNB. We also compared it with a CNN and GNN classifier using the same setting as the components use in our model. Actually, for our ResHybnet model, when setting it to gnn mode, it would perform exactly like a GNN classifier, while setting it to cnn mode will make it act like a CNN classifier. Specifically, for the CNN

[2] https://pytorch.org/docs/stable/index.html
[3] https://scikit-learn.org/stable/
[4] https://pytorch-geometric.readthedocs.io/en/latest/

Table 2
Manual feature engineering method.

Feature name	Process method	Value type
First logon time Last logoff time First device activity time Last device activity time	Map the time of the activity to the range of [0, 1], where 0 stands for 00:00, 1 stands for 24:00	Float
Number of off-work device activities	Count the number of the activities related to device in off-work time	Integer

Table 3
Daily activity encoding rules.

Activity type	Code for work time	Code for off-work time
Logon a PC	1	13
Logoff a PC	2	14
Connect a USB drive	3	15
Disconnect a USB drive	4	16
Open a .doc file	5	17
Open a .exe file	6	18
Open a .jpg file	7	19
Open a .pdf file	8	20
Open a .text file	9	21
Open a .zip file	10	22
Send an email to internal address	11	23
Send an email to external address	12	24

Table 4
Performance comparison for different detection models.

Model name	Acc (%)	Pre (%)	Rec (%)	F1 (%)
ResHybnet(GCN+CNN)	92.62	91.52	92.87	92.19
SVM	90.56	87.68	92.91	90.22
CNN	90.44	87.13	93.40	90.15
GNB	89.69	86.67	92.16	89.33
RF	89.69	87.46	91.04	89.21
LR	89.34	87.36	90.30	88.81
GCN	70.77	62.14	96.34	75.53

component, we used a simple two-layer convolutional networks, and for the GNN component we choose one of the most popular, namely GCNs, and apply a simple two-layer GCN model for our hybrid model. The comparison results on the constructed sample dataset are presented in Table 4.

From Table 4, we could clearly see that the proposed detection model achieved the best result overall, leading performance in accuracy and precision, with the number of 92.62%, 91.52%. Although Recall score does not top the list, the more balanced f1 score is notably higher than all the other classification models. What worth noticing is, when the components adopted in our model working separately as a classification model, none of them could achieve a desirable result. In particular, the GCN model comes in the last place in all the metrics except for Recall rate. One explanation is that in the GNN model, each node will collect node attribute information from its neighbors for updating in each training round, which inevitable causes the attribute of its own to be downgraded. For this dataset, although there is some information hidden in organizational graph, individual node attribute still plays a more important role in classification.

Therefore, from Table 4, we could fairly say, for this insider threat sample dataset, the proposed ResHybnet model could make the best use of the discovering power from both components, and has a satisfying performance.

5.1.2. Discussion on the power of residual link

As shown in the previous part, our model outperforms not only traditional classifiers but also the deep learning models that have been adopted as a component of the model. However, this improved performance does not come by simply stacking components together. We did an ablation experiment by taking out the residual link in our model to show the power of it. We set the ResHybnet model to work in hybrid mode, and tried GCNs and GATs as GNN component separately. Each time, we switch the residual link on/off to see the effect. The result is shown in Table 5, and Fig. 3.

From Table 5, we could see no matter which GNN component we choose, just stacking it with CNN component will not improve the performance. Without the residual link, the hybrid model's performance would even downgrade to the worst situation. This phenomenon is also well visualized by the huge gap of performance under the setting of with or without residual link in Fig. 3.

As mentioned in the contribution part, we think the power of this residual link mainly comes from the fact it keeps the full information of the node attribute, and adds it back to the output of GNN component. In this way, our model could make use of the information from graph and individual node attribute to achieve better performance.

5.2. Ablation study of feature engineering

In our solution, the other contribution is the integrated feature engineering combining manual features from standalone activities and automatic features from sequential activities. In this section, we will show how these two groups of features contribute to the final detection task, holding the best hybrid model setting unchanged. Which is to say, we choose GCN for GNN component and working in hybrid mode. Firstly, we will show the comparison result for different feature combinations, and secondly, we will investigate how the length of sequence affects the final result.

5.2.1. Representative power of different feature groups

To better capture the nature of employee's behavioral pattern in all kinds of activities, we designed two groups of features for engineering as shown in Section 4.2. For standalone activities, it is relatively easy to select out some feature based on domain knowledge and experience. However, we believe there also exist some patterns behind how those activities are performed, which could only be discovered by algorithm. With the proposed LSTM autoencoder, we extracted out feature automatically from activity

Table 5
Performance comparison for with/without the residual link.

Model setting	Residual link status	Acc (%)	Pre (%)	Rec (%)	F1 (%)
ResHybnet (GCN+CNN)	With residual link	92.62	91.52	92.87	92.19
	Without residual link	70.38	63.12	89.51	73.73
ResHybnet (GAT+CNN)	With residual link	91.01	89.79	91.19	90.47
	Without residual link	68.69	62.66	88.02	72.16

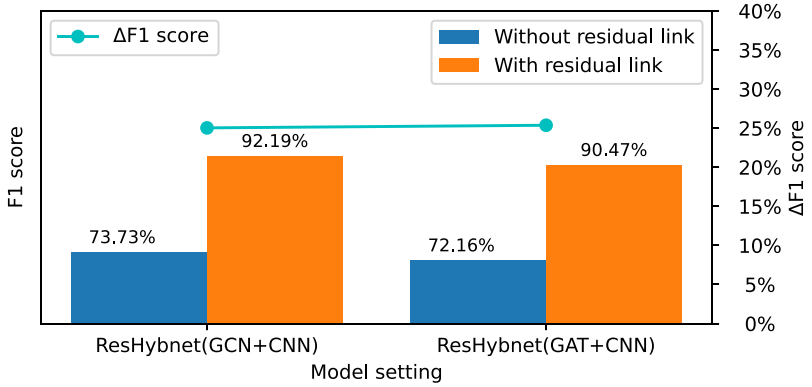

Fig. 3. F1 score comparison for with/without the residual link.

Table 6
Performance comparison for different feature groups.

Feature combination	Acc (%)	Pre (%)	Rec (%)	F1 (%)
Manual + Automatic Feature	92.62	91.52	92.87	92.19
Manual Feature	92.01	89.97	93.36	91.63
Automatic Feature	86.03	83.06	88.25	85.56

sequence. Table 6 shows how different feature groups contribute in our solution.

From Table 6, it shows that in some situations, manual feature selection based on domain knowledge could already help to obtain satisfying result, while the hidden pattern in sequential behavior may not be so significant. However, adding those automatic features to feature pool indeed improves final classification performance for this case. Specifically, regardless of recall rate drop, adding automatic feature to manual feature group will improve f1 score from 91.63% to 92.19%, and the rest metrics also enjoyed improvement.

5.2.2. Discussion on sequence length in automatic feature engineering

In the previous section, we showed that for this sample dataset, adding sequential activity features to the feature representation indeed improves the performance of insider threat discovering. At the beginning, we intuitively set the maximal sequence length as the input sequence length for the LSTM auto-encoder, and fill the empty positions at the end with 'none' activities. However, it is possible there is a balance that needs to be reached between detection performance and input sequence length due to computation concerns. Moreover, it is possible that employee's activities have a circling pattern, and taking the maximal sequence length as the input for LSTM auto-encoder is not a must choice.

Therefore, in this experimental part, we tried to find some insights by looking into how the sequence length choice in feature engineering part will affect the final detection performance. Here, we hold model setting and manual features unchanged, compare the performance of classification for different sequence length choices during the feature engineering process.

Fig. 4 shows the statistical characteristics for daily activity sequence. Based on this information, we set the different input

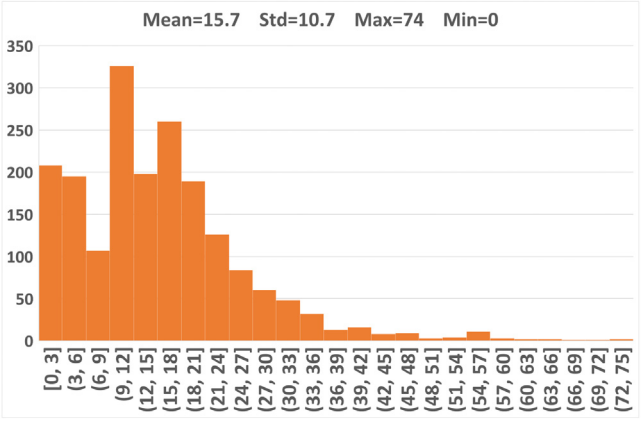

Fig. 4. Statistics and histogram for activity sequence lengths.

length for auto-encoder to train and extract feature, and for all sequence lengths, feature output dimension is set to the same number of 5.

In the comparison process, we investigated different sequence length based on how it deviates from the mean. Specifically, starting from the average length, we set the sequence length roughly every additional standard deviation both above and below average, until out of boundary. That is to say, for a specified user-day sample, if the actual sequence length exceeds our setting, the rest would be cut off, and if the actual sequence length is smaller than the setting, empty positions at the end will be filled with 'none' activity type. We present the comparison result in Table 7.

From Table 7 and Fig. 5, we could observe there exists a general trend of increased overall performance along with increased sequence length. However, there are two things worth noticing. First, starting from 1 standard deviation above mean all the way to maximal sequence length, F1 score actually fluctuates around a point, which means for this dataset and LSTM auto-encoder, after a certain point of sequence length, features extracted are becoming limited. Second, for sequence length equals to and smaller than the mean, the final performance actually suffered drawing back compared with the performance using only manually selected features.

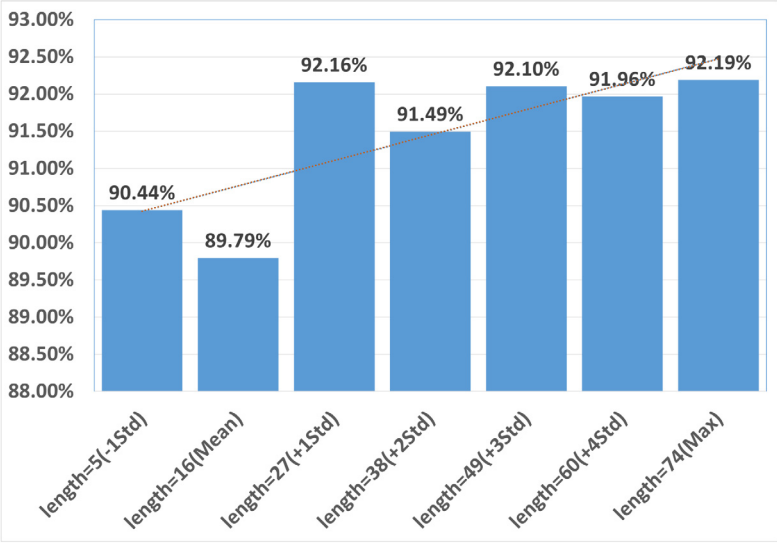

Fig. 5. F1 score trend for different activity sequence lengths.

Table 7
Performance comparison for different activity sequence length settings.

Input sequence length	Acc (%)	Pre (%)	Rec (%)	F1 (%)
Length = 5 (−1 Std)	90.91	89.08	91.87	90.44
Length = 16 (Mean)	90.28	88.37	91.27	89.79
Length = 27 (+1 Std)	92.59	91.38	92.95	92.16
Length = 38 (+2 Std)	91.92	90.33	92.69	91.49
Length = 49 (+3 Std)	92.50	90.97	93.28	92.10
Length = 60 (+4 Std)	92.40	91.12	92.84	91.96
Length = 74 (Max)	92.62	91.52	92.87	92.19

Thus, we think how to choose the proper length is largely determined by how long the maximal length is. If the activity length is relatively small like the situation in our sample dataset, then using the maximal sequence length would certainly give a better result. However, if the maximal sequence is too long for processing, and the distribution of length is extremely imbalanced, then analyzing statistic of sequence length could be a good starting point for reaching a balanced choice. For example, in this specific situation, setting the length to be 1 standard deviation above mean could have already given a satisfying result.

6. Conclusion

In this paper, we proposed a residual hybrid network (ResHybnet) and an integrated featuring engineering solution to handle the challenges in insider threat detection research. Experimental results on the CERT 4.2 dataset show the proposed ResHybnet model could outperform other state-of-art classifiers in insider threat detection task, and the residual link design in our detection model has proved to be effective. At the same time, the proposed LSTM auto-encoder approach for automatic feature engineering indeed indicates there is some hidden information in daily activity sequence, and LSTM auto-encoder is an effective feature extractor.

However, there are still some limitations in our work that could be improved in the future. Firstly, we only utilized part of the activities provided by CERT 4.2 dataset in our use case study, and it could be more promising as well as challenging if more activity data are introduced in the feature engineering part. Secondly, although our approach shows potential to a certain extent, in practice, insider threat detection is an extremely imbalanced classification problem, and we need to face this obstacle in real-world application. Thirdly, in real world application, an online learning strategy would be more suitable for insider threat scenario, and this could be our prior working direction in the future. As for insider threat detection itself, we think there still exist two major challenges, one of which is that currently not so many rich datasets are available due to privacy issues, and the other is how to better protect privacy for an average employee when monitoring insiders within an organization. We expect follow-up researchers to propose some solutions regarding those two aspects.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partially supported by the Research Program of Chongqing University of Arts and Sciences, China (Grant No. P2020RG08).

References

[1] Rasool RU, Ahmed K, Anwar Z, Wang H, Ashraf U, Rafique W. CyberPulse++: A machine learning-based security framework for detecting link flooding attacks in software defined networks. Int J Intell Syst 2021;36(8):3852–79.
[2] Agrawal N, Kumar R. Security perspective analysis of industrial cyber physical systems (I-CPS): A decade-wide survey. ISA Trans 2022.
[3] Wang Y, Shen Y, Wang H, Cao J, Jiang X. Mtmr: Ensuring mapreduce computation integrity with merkle tree-based verifications. IEEE Trans. Big Data 2016;4(3):418–31.
[4] Sun N, Zhang J, Rimba P, Gao S, Zhang LY, Xiang Y. Data-driven cybersecurity incident prediction: A survey. IEEE Commun Surv Tutor 2018;21(2):1744–72.
[5] Center CNIT. Common sense guide to mitigating insider threats. Seventh. Carnegie Mellon University; 2022.
[6] Wang H, Sun L. Trust-involved access control in collaborative open social networks. In: 2010 Fourth International conference on network and system security. IEEE; 2010, p. 239–46.
[7] Wang H, Cao J, Zhang Y. A flexible payment scheme and its role-based access control. IEEE Trans Knowl Data Eng 2005;17(3):425–36.

[8] You M, Yin J, Wang H, Cao J, Wang K, Miao Y, Bertino E. A knowledge graph empowered online learning framework for access control decision-making. World Wide Web 2022;1–22.
[9] Sun X, Li M, Wang H, Plank A. An efficient hash-based algorithm for minimal k-anonymity. In: Conferences in research and practice in information technology, vol. 74. CRPIT, 2008, p. 101–7.
[10] Yin Y, Yu W, Bu X, Yu Q. Security data-driven iterative learning control for unknown nonlinear systems with hybrid attacks and fading measurements. ISA Trans 2022.
[11] Wang H, Cao J, Zhang Y, Wang H, Cao J, Zhang Y. Building access control policy model for privacy preserving and testing policy conflicting problems. Access Control Manag Cloud Environ 2020;225–47.
[12] Lin G, Wen S, Han Q-L, Zhang J, Xiang Y. Software vulnerability detection using deep neural networks: a survey. Proc IEEE 2020;108(10):1825–48.
[13] Sun X, Wang H, Li J, Pei J. Publishing anonymous survey rating data. Data Min Knowl Discov 2011;23(3):379–406.
[14] Chen X, Li C, Wang D, Wen S, Zhang J, Nepal S, Xiang Y, Ren K. Android HIV: A study of repackaging malware for evading machine-learning detection. IEEE Trans Inf Forensics Secur 2019;15:987–1001.
[15] Yin J, You M, Cao J, Wang H, Tang M, Ge Y-F. Data-driven hierarchical neural network modeling for high-pressure feedwater heater group. In: Australasian database conference. Springer; 2020, p. 225–33.
[16] Tang C, Yin J. A localization algorithm of weighted maximum likelihood estimation for wireless sensor network. J Inform Comput Sci 2011;8(16):4293–300.
[17] Hu X, Ma W, Chen C, Wen S, Zhang J, Xiang Y, Fei G. Event detection in online social network: Methodologies, state-of-art, and evolution. Comp Sci Rev 2022;46:100500.
[18] Yin J, Tang M, Cao J, You M, Wang H, Alazab M. Knowledge-driven cybersecurity intelligence: software vulnerability co-exploitation behaviour discovery. IEEE Trans Ind Inf 2022.
[19] Hong W, Yin J, You M, Wang H, Cao J, Li J, Liu M. Graph intelligence enhanced bi-channel insider threat detection. In: Network and system security: 16th international conference, NSS 2022, Denarau Island, Fiji, December 9–12, 2022, Proceedings. Springer; 2022, p. 86–102.
[20] Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM, Solomon JM. Dynamic graph cnn for learning on point clouds. Acm Trans Graphics (Tog) 2019;38(5):1–12.
[21] Homoliak I, Toffalini F, Guarnizo J, Elovici Y, Ochoa M. Insight into insiders and it: A survey of insider threat taxonomies, analysis, modeling, and countermeasures. ACM Comput Surv 2019;52(2):1–40.
[22] Schonlau M, DuMouchel W, Ju W-H, Karr AF, Theus M, Vardi Y. Computer intrusion: Detecting masquerades. Statist Sci 2001;58–74.
[23] Garg A, Rahalkar R, Upadhyaya S, Kwiat K. Profiling users in GUI based systems for masquerade detection. In: Proceedings of the 2006 IEEE Workshop on information assurance, vol. 2006. 2006, p. 48–54.
[24] Glasser J, Lindauer B. Bridging the gap: A pragmatic approach to generating insider threat data. In: 2013 IEEE Security and privacy workshops. IEEE; 2013, p. 98–104.
[25] 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. 2015, p. 13–20.
[26] Gamachchi A, Boztas S. Insider threat detection through attributed graph clustering. In: 2017 IEEE Trustcom/BigDataSE/ICESS. IEEE; 2017, p. 112–9.
[27] Jiang J, Chen J, Gu T, Choo K-KR, Liu C, Yu M, Huang W, Mohapatra P. Anomaly detection with graph convolutional networks for insider threat and fraud detection. In: MILCOM 2019-2019 IEEE Military communications conference. MILCOM, IEEE; 2019, p. 109–14.
[28] Zhang S, Liu Z, Chen Y, Jin Y, Bai G. Selective kernel convolution deep residual network based on channel-spatial attention mechanism and feature fusion for mechanical fault diagnosis. ISA Trans 2022.
[29] Wang H, Wang Y, Taleb T, Jiang X. Special issue on security and privacy in network computing. World Wide Web 2020;23:951–7.
[30] Liu A, Martin C, Hetherington T, Matzner S. A comparison of system call feature representations for insider threat detection. In: Proceedings from the Sixth Annual IEEE SMC Information assurance workshop. IEEE; 2005, p. 340–7.
[31] Lin L, Zhong S, Jia C, Chen K. Insider threat detection based on deep belief network feature representation. In: 2017 International conference on green informatics. ICGI, IEEE; 2017, p. 54–9.
[32] Chattopadhyay P, Wang L, Tan Y-P. Scenario-based insider threat detection from cyber activities. IEEE Trans Comput Soc Syst 2018;5(3):660–75.
[33] Singh M, Mehtre B, Sangeetha S. Insider threat detection based on user behaviour analysis. In: International conference on machine learning, image processing, network security and data sciences. Springer; 2020, p. 559–74.
[34] Yuan F, Cao Y, Shang Y, Liu Y, Tan J, Fang B. Insider threat detection with deep neural network. In: International conference on computational science. Springer; 2018, p. 43–54.
[35] Paul S, Mishra S. LAC: LSTM autoencoder with community for insider threat detection. In: 2020 the 4th International conference on big data research (ICBDR'20). 2020, p. 71–7.
[36] 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. 2019, p. 1777–94.
[37] Sun X, Wang H, Li J. Satisfying privacy requirements: One step before anonymization. In: Pacific-Asia conference on knowledge discovery and data mining. Springer Berlin Heidelberg; 2010, p. 181–8.
[38] Sun X, Wang H, Li J, Zhang Y. Satisfying privacy requirements before data anonymization. Comput J 2012;55(4):422–37.