Confront Insider Threat: Precise Anomaly Detection in Behavior Logs Based on LLM Fine-Tuning

Shuang Song[1,2,3], **Yifei Zhang**[1,3,*], **Neng Gao**[1,3,*]
[1]Institute of Information Engineering, Chinese Academy of Sciences
[2]School of Cyber Security, University of Chinese Academy of Sciences
[3]Key Laboratory of Cyberspace Security Defense
{songshuang, zhangyifei, gaoneng}@iie.ac.cn

Abstract

Anomaly-based detection is effective against evolving insider threats but still suffers from low precision. Current data processing can result in information loss, and models often struggle to distinguish between benign anomalies and actual threats. Both issues hinder precise detection. To address these issues, we propose a precise anomaly detection solution for behavior logs based on Large Language Model (LLM) fine-tuning. By representing user behavior in natural language, we reduce information loss. We fine-tune the LLM with a user behavior pattern contrastive task for anomaly detection, using a two-stage strategy: first learning general behavior patterns, then refining with user-specific data to improve differentiation between benign anomalies and threats. We also implement a fine-grained threat tracing mechanism to provide behavior-level audit trails. To the best of our knowledge, our solution is the first to apply LLM fine-tuning in insider threat detection, achieving an F1 score of 0.8941 on the CERT v6.2 dataset, surpassing all baselines.

1 Introduction

Malicious insider threats typically manifest as employees misusing their legal permissions and trust to compromise the confidentiality, integrity, and availability of organizational assets. As technology advances, the concealment and complexity of insider threats have progressively increased, exposing businesses and organizations to unprecedented risks. Innovation in insider threat detection is urgently needed.

Due to the scarcity and latency of labeled insider threat data, classification-based detection has difficulty remaining effective against evolving threat activities. Consequently, anomaly-based detection, which models only benign user behaviors, has become common. However, current anomaly detection methods suffer from low precision, compromising their credibility.

Two main factors reduce anomaly detection precision. First, current behavior data processing often results in information loss. Common methods classify user behaviors into categories to construct behavioral sequences (Yu et al., 2022; Yuan et al., 2020; Rashid et al., 2016) or count vectors (Tuor et al., 2017; Meng et al., 2020; Song et al., 2024b) for anomaly detection. Due to the complexity of user behavior, these methods rely on hundreds (Tuor et al., 2017) to thousands (Yuan et al., 2020) of behavioral categories. Nevertheless, not only is information loss unavoidable, but data sparsity is also introduced. Moreover, behavioral sequences and count vectors are highly aggregated and encoded, resulting in poor readability and coarse-grained threat identification, posing challenges for security audits.

Second, detection models struggle to differentiate between threatening behaviors and benign anomalies. User behavior patterns can change due to environmental factors, many of which are not threatening. Common anomaly detection algorithms (e.g., distance-based algorithms (Meng et al., 2020), one-class classification (Lin et al., 2017; Soh et al., 2019; Rashid et al., 2016), autoencoder reconstruction (Yu et al., 2022; Tuor et al., 2017; Nasir et al., 2021)) lack the capability to distinguish between benign anomalies and actual threats. Moreover, the definition of an insider threat varies based on the specific needs of organizations, introducing additional uncertainties in threat detection.

To address the above issues, we propose a novel solution: precise anomaly detection in behavior logs based on Large Language Model (LLM) fine-tuning. By leveraging the inherent strong pattern identification and generalization capabilities of LLMs, we enhance insider threat detection performance.

*Corresponding authors.

We represent user behavior in natural language using a template to enhance readability and information retention. LLM-based Customizable topic specification is introduced for free text analysis, allowing targeted focus on threat clues that organizations truly care about. We propose a user behavior pattern contrastive task based on contrastive learning to fine-tune the LLM, enabling it to distinguish between different behavioral patterns for effective anomaly detection. Our novel two-stage fine-tuning strategy first adapts the model to generalized behavior patterns, then fine-tunes it to user-specific patterns. The model learns common changes in generalized user behavior patterns, allowing it to assess a lower degree of threat for similar benign changes exhibited by specific users, thereby enhancing its ability to distinguish between benign anomalies and threats. Additionally, we introduce a fine-grained threat tracing mechanism, leveraging a model-agnostic local explanation technique (Ribeiro et al., 2016) to achieve behavior-level threat capture, enhancing detection credibility.

The contributions of this paper are summarized as follows:

1. To the best of our knowledge, this is the first application of LLM fine-tuning in the field of insider threat detection. We develop a framework for insider threat detection based on LLM fine-tuning, encompassing the entire process from natural language behavior representation, through LLM fine-tuning for anomaly detection, to behavior-level threat tracing.
2. We propose a two-stage fine-tuning strategy that enables the LLM to learn both general and specific behavior information.
3. We propose a novel user behavior pattern contrastive task for LLM fine-tuning, which supports both user behavior pattern learning and anomaly-based detection.
4. We introduce a fine-grained threat tracing mechanism based on a model-agnostic local explanation technique for behavior-level threat capture.
5. We validate the performance of our solution using the CERT v6.2 dataset. Experimental results show that our solution outperforms all baselines with an F1 score of 0.8941.

2 Related Work

Current insider threat detection mainly includes classification-based and anomaly-based approaches. Classification-based detection utilizes labeled data for supervised learning, usually maintaining low false alarm rates. However, due to the scarcity of labeled threat behavior data, classification-based detection experiences severe data imbalance issues and struggles to detect unknown threat behaviors beyond the training set (Le et al., 2021; Yuan et al., 2020; Wu and Li, 2021).

Anomaly-based detection models only benign behaviors and detects threats by identifying the differences between test data and benign behavioral patterns. Currently, one-class classification (Meng et al., 2020; Rashid et al., 2016; Lin et al., 2017; Soh et al., 2019) and autoencoder reconstruction (Tuor et al., 2017; Nasir et al., 2021; Yu et al., 2022) are two of the most common anomaly detection techniques. Anomaly detection is favored by researchers due to its lack of requirement for labeled data and sensitivity to evolving threat behaviors. However, existing anomaly detection schemes still struggle with high false alarms.

Natural Language Processing (NLP) was introduced into the field of insider threat detection. Liu et al. (Liu et al., 2019) embedded the behavior categories using Word2vec. Yuan et al. (Yuan et al., 2020) introduced the Transformer and Masked Language Model (MLM) for behavioral sequence representation. However, these schemes did not represent behaviors in natural language, and their detection performance was limited by the state of NLP techniques at the time.

Since the successful practice of LLMs, their potential in outlier detection has gained increasing attention. Egersdoerfer et al. assess ChatGPT's capabilities in log outlier detection (Egersdoerfer et al., 2023); Liu et al. use LLM-based knowledge distillation for time series outlier detection (Liu et al., 2024); Li et al. achieve tabular data outlier detection through supervised LoRA fine-tuning (Li et al., 2024); Dong et al. investigate the application of LLMs in time series outlier detection through direct detection, prompt engineering, and fine-tuning (Dong et al., 2024). These studies all show promising results.

While insider threat detection shares similarities with outlier detection, it involves greater complexity, demanding advanced capabilities of contextual understanding, pattern identification, and general-

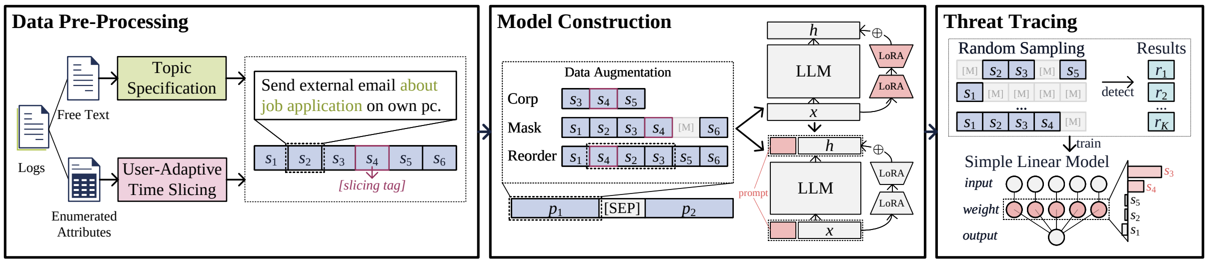

Figure 1: The framework of our solution.

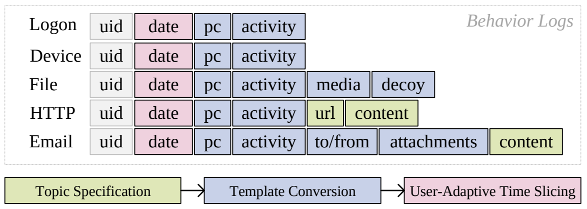
Behavior Logs
Logon uid date pc activity
Device uid date pc activity
File uid date pc activity media decoy
HTTP uid date pc activity url content
Email uid date pc activity to/from attachments content
Topic Specification
Template Conversion
User-Adaptive Time Slicing

Figure 2: Data pre-processing details. Each log field uses the same color as the corresponding processing method.

ization. Recently, Song et al. (Song et al., 2024a) proposed a novel multi-agent collaborative framework for insider threat detection based on LLMs, marking an exciting and recent advancement. In contrast, our work focuses on applying LLM fine-tuning, which remains a promising and complementary direction in this field.

3 Our Solution

As shown in Figure 1, our solution is divided into three stages: data pre-processing, model construction, and threat tracing. In data pre-processing, user behavior logs are transformed into natural language text. In model construction, we fine-tune an LLM for insider threat detection using the user behavior pattern contrastive task. In threat tracing, we analyze the instances detected as positive with a model-agnostic local explanation technique to identify insider threat behaviors.

3.1 Data Pre-Processing

As shown in Figure 2, data pre-processing involves topic specification, template conversion, and user-adaptive time slicing, which we detail separately in the following sections.

3.1.1 Topic Specification

Free text in the logs, which includes email content, web page content, web URLs, etc., typically has a volume far exceeding that of other attribute values, much of which is not valid information for detection. Placing the free text directly alongside other attribute values in the natural language text converted from the behavior logs can introduce significant noise, interfering with the LLM and increasing the complexity of the behavior detection.

Therefore, we divide the entire behavior detection process into two parts for detection task simplification. First, we use a general LLM to analyze whether the free text contains specific topics related to insider threats. If identified, the topic keywords are converted with other log attribute values into natural language text for subsequent detection.

The prompt for topic specification is customizable. Leveraging the powerful generalization capabilities of the LLM, organizations can set the topics to be identified in the prompt according to their specific needs and determine which content corresponds to which topics. This enables the detection model to accurately capture the threat activities of genuine concern for the organization.

3.1.2 Template Conversion

User behavior logs are inherently composed of natural language vocabulary and character strings. Therefore, we only need to simply rearrange the content of the behavior logs using templates to conform to the basic syntax of natural language, thus converting behavior logs into natural language text.

The structure of the conversion template is as follows. Round brackets indicate optional parts, while italic square brackets represent the log contents (and their variations). The choice of prepositions may vary depending on the type of activity.

> *[activity]* (*[decoy]*) (*[media]*) (*[to/from]*) (with *[attachments]*) (about *[topic keywords]*) on *[pc]*.

Since we use the user behavior pattern contrastive task (detailed in 3.2.1) as the fine-tuning task for the LLM, user IDs are not explicitly included in the behavioral sentences to adapt to the fine-tuning task. Additionally, we assign roles (e.g., colleague, supervisor, separated employee) to email addresses and PC IDs to describe the corresponding users' relationships to the behavior subjects, facilitating interpersonal interaction analysis.

3.1.3 User-Adaptive Time Slicing

Using the method described above, we convert a behavior log item into a behavioral sentence, denoted as s. Behavioral sentences are aggregated by user ID and ordered by timestamps. Each day's log forms a behavioral paragraph, denoted as $p = (s_1, s_2, \dots, s_n)$, where n is the number of behavioral sentences in a behavioral paragraph. We apply a user-adaptive time slicing scheme from Song et al.'s work (Song et al., 2024b) to the behavioral paragraphs to strengthen the temporal distribution information of user behavior. Each behavioral paragraph is split into two parts, with a slicing tag (written as 'Go off duty' for context and treated as a behavioral sentence) inserted between them. Identical behavioral sentences within each part are merged to reduce paragraph length and increase information density.

3.2 Model Construction

We propose a user behavior pattern contrastive task based on contrastive learning as a fine-tuning task, and introduce three fine-tuning strategies: ITDLM-0, ITDLM-U, and ITDLM-II, for LLM fine-tuning to achieve insider threat detection.

3.2.1 User Behavior Pattern Contrastive Task

Contrastive learning, being a self-supervised learning algorithm, does not require data labels, which aligns with the needs of anomaly detection models. We construct the user behavior pattern contrastive task based on contrastive learning to fine-tune the LLM for behavioral anomaly detection.

Firstly, we use a behavioral sentence s as the basic unit for augmenting the behavioral paragraphs. Drawing on Xie et. al's work (Xie et al., 2022), we establish a set of data augmentation operations, denoted as $\mathcal{A} = \{a_{crop}(\cdot), a_{mask}(\cdot), a_{reorder}(\cdot)\}$. User behavior contains inherent randomness in both attributes and order. Data augmentation helps the model learn stable behavior patterns. The crop operation enhances focus on key behavioral details, the mask operation helps to mitigate overfitting, and the reorder operation reduces reliance on behavior sequence order.

We augment each behavioral paragraph with a probability of α to obtain augmented paragraph $\hat{p}$. The augmentation operation $a_i(\cdot)$ is randomly selected from $\mathcal{A}$.

$$\hat{p} = \begin{cases} a_i(p), & \text{with probability } \alpha, \\ p, & \text{with probability } 1-\alpha. \end{cases} \tag{1}$$

We form a pair $(\hat{p}_a, \hat{p}_b)$ with two augmented behavioral paragraphs, $\hat{p}_a$ and $\hat{p}_b$, for contrastive learning. To better adapt to the LLM, we add prompts to the behavioral paragraph pair and generate the input data as follows:

> Instruction: Are behavioral sequences A and B from the same user?
>
> Input: Behavioral sequence A is as follows: *[$\hat{p}_a$]* SEP Behavioral sequence B is as follows: *[$\hat{p}_b$]*
>
> Answer:

During training, we set the augmentation probability to α_{train}, considering a pair of augmented behavioral paragraphs from the same user as a negative instance, and a pair from different users as a positive instance to train the model to distinguish behavioral patterns. During testing, we use the unaugmented behavior paragraph to be tested as the experimental subject to preserve threat features, and a known benign paragraph from the same user, augmented with α_{test}, as the control subject. The model then analyzes the control-experiment pair to generate the detection result $\hat{y}$. A negative result indicates the experimental subject is benign, and a positive result indicates a threat.

By pairing the experimental subject with N known benign behavioral paragraphs for detection, we obtain N results. The frequency r of positive results is calculated as the threat rate of the experimental subject:

$$r = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(\hat{y}_i = y^+) \tag{2}$$

3.2.2 LLM Fine-Tuning Strategies

We propose three fine-tuning strategies based on the user behavior pattern contrastive task, named ITDLM-0, ITDLM-U, and ITDLM-II, as shown in Figure 3. In this section, we denote the user to be tested as u^*, and denote a set of non-tested users as $\mathcal{U} = \{u_1, u_2, \dots, u_M\}$ with a size of M. Let $\hat{p}_i^u$ represent the augmented behavioral paragraph with index i from user u. Let M_0 represent the pre-trained LLM. Each training set contains only benign behaviors.

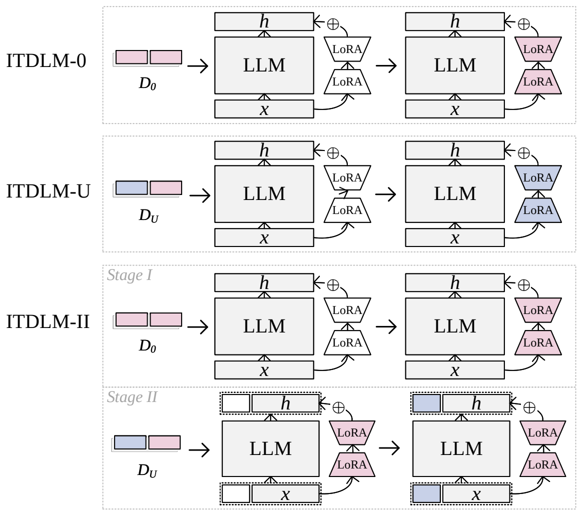

Figure 3: Three fine-tuning strategies: ITDLM-0, ITDLM-U, and ITDLM-II based on the user behavior pattern contrastive task. Blue represents behavioral information of the user to be tested; purple represents behavioral information of the non-tested users.

1) ITDLM-0 We construct the training set using the augmented paragraphs from only non-tested users as follows:

$$\mathcal{D}_0 = \{((\hat{p}_i^{u_k}, \hat{p}_i^{u_l}), y^+), ((\hat{p}_i^{u_k}, \hat{p}_j^{u_k}), y^-) | u_k, u_l \in \mathcal{U}\} \tag{3}$$

Next, M_0 is fine-tuned using the LoRA method as follows. Here, ΔW represents the updatable weight matrix.

$$M_{\text{ITDLM-0}} = \text{Fine-Tune}(M_0, \mathcal{D}_0; \Delta W) \tag{4}$$

The original M_0 is trained into a detection model, $M_{\text{ITDLM-0}}$, capable of distinguishing different behavior patterns. Since the training set does not include behavior data from the user to be tested u^*, $M_{\text{ITDLM-0}}$ detects threatening behaviors for u^* without any prior information on u^*'s behavior patterns.

2) ITDLM-U We construct a training set using the augmented paragraphs from both the user to be tested u^* and the non-tested users as follows:

$$\mathcal{D}_\text{U} = \{((\hat{p}_i^{u^*}, \hat{p}_i^{u_k}), y^+), ((\hat{p}_i^{u^*}, \hat{p}_j^{u^*}), y^-) | u_k \in \mathcal{U}\} \tag{5}$$

Pairs consisting solely of the augmented paragraphs from u^* are considered negative instances, while pairs that include augmented paragraphs from both u^* and non-tested users are considered positive instances. Next, M_0 is fine-tuned using the LoRA method as follows:

$$M_{\text{ITDLM-U}} = \text{Fine-Tune}(M_0, \mathcal{D}_\text{U}; \Delta W) \tag{6}$$

The augmented paragraphs from non-tested users in the training set $\mathcal{D}_\text{U}$ are used solely as behaviors "different from u^*'s pattern". Compared to $M_{\text{ITDLM-0}}$, $M_{\text{ITDLM-U}}$ is built primarily using u^*'s behavioral pattern information rather than general user behavioral pattern information, making it a single-user detection model tailored to u^*.

3) ITDLM-II $M_{\text{ITDLM-0}}$ offers better generalization but lacks specific user behavior information, while $M_{\text{ITDLM-U}}$ incorporates specific information but loses some generalization. To address these shortcomings, we propose a two-stage fine-tuning strategy, ITDLM-II:

$$M_{\text{ITDLM-II}} = \text{Fine-Tune}(M_{\text{ITDLM-0}}, \mathcal{D}_\text{U}; \theta_p) \tag{7}$$

In the first stage, the training set $\mathcal{D}_0$ is used with the LoRA method to fine-tune M_0, resulting in $M_{\text{ITDLM-0}}$. In the second stage, the training set $\mathcal{D}_\text{U}$ is used with the P-tuning v2 method to further fine-tune $M_{\text{ITDLM-0}}$, resulting in $M_{\text{ITDLM-II}}$. Here, θ_p represents the parameters of the prompt vectors. By adding prompt vectors specific to u^* to $M_{\text{ITDLM-0}}$, $M_{\text{ITDLM-II}}$ inherits the general user behavior pattern information from $M_{\text{ITDLM-0}}$ and also possesses behavior pattern information specific to the user to be tested.

3.3 Fine-grained Threat Tracing

The fine-tuned LLM detects threats by identifying anomalous patterns in behavioral paragraphs, allowing us to trace specific sentences that led to a positive judgment using explanation techniques. Following Ribeiro et al.'s method (Ribeiro et al., 2016), we use a simple linear model to map the sample space around positive instances to the corresponding detection result space. The weights of the linear model help identify sentences contributing more to positive results, pinpointing specific threat behaviors.

Fine-grained threat tracing mechanism consists of local sampling and linear fitting.

3.3.1 Local Sampling

For a positive behavioral paragraph $\pi^{u^*} = (s_1, s_2, \ldots, s_n)$ of a user to be tested u^*, we randomly generate a selection vector set $\mathcal{V}$ of size K, $\mathcal{V} = \{v_1, v_2, \ldots, v_K\}$, $v_i \in \{0,1\}^n$. We sample the behavioral paragraph π^{u^*} using s as the basic unit based on $\mathcal{V}$, which results in a sampling set $\Pi_{\mathcal{V}}^{u^*} = \{\pi_{v_1}^{u^*}, \pi_{v_2}^{u^*}, \ldots, \pi_{v_K}^{u^*}\}$, where

$\pi_{v_i}^{u^*} = \pi^{u^*} \otimes v_i$ and $\otimes$ represents the selection operation.

We construct a test set $\mathcal{D}_{\Pi_{\mathcal{V}}^{u^*}} = \{(\hat{p}_j^{u^*}, \pi_{v_i}^{u^*}) | \pi_{v_i}^{u^*} \in \Pi_{\mathcal{V}}^{u^*}\}$, where $\hat{p}_j^{u^*}$ is the augmented known benign behavioral paragraph of user u^*. We use the detection model to analyze the test set $\mathcal{D}_{\Pi_{\mathcal{V}}^{u^*}}$, obtaining the detection results.

3.3.2 Linear Fitting and Threat Tracing

To better capture the impact of selecting behavioral sentences on prediction results and avoid information loss, we process the predictions of fine-tuned LLM as follows:

$$\hat{r}_i = \text{softmax}_{\{\omega_i^-, \omega_i^+\}}(\omega_i^+) = \frac{e^{\omega_i^+}}{e^{\omega_i^-}, e^{\omega_i^+}} \tag{8}$$

Here, ω^+ denotes the predicted probability of the positive label word, and ω^- represents the predicted probability of the negative label word. Since the output of the fine-tuned LLM is only related to the positive and negative label words, the probabilities of other words are not considered. We calculate the softmax of the set $\{\omega^-, \omega^+\}$ and use the resulting value corresponding to ω^+ as a measure of threat for samples.

Construct a training set $\mathcal{D}_L = \{(v_i, d_i, \hat{r}_i) | i = 1, 2, \ldots, K\}$, where d_i is the weight for sampling v_i. Using mean squared error as the loss function, we train a linear model $M_L(\mathbf{w}, b)$ on $\mathcal{D}_L$, where $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$. To prevent overfitting, batch normalization and Dropout are included during training.

Due to the positive detection results of all sampled instances, the samples are also predominantly detected as positive. The sampling dataset $\mathcal{D}_L$ is heavily imbalanced, leading to skewed positive weights in the linear model and interfering with threat tracing. To correct this, we center the weights $\mathbf{w}$ by subtracting the mean to reduce the impact of data imbalance.

$$\hat{w}_i = w_i - \frac{1}{n}\sum_{j=1}^{n} w_j \tag{9}$$

The weights $\hat{\mathbf{w}} = (\hat{w}_0, \hat{w}_1, \ldots, \hat{w}_n)^T$ correspond one-to-one with the behavioral sentences in π^{u^*}. If $\hat{w}_i > 0$, it indicates that s_i represents a threat behavior; otherwise, s_i represents a benign behavior.

4 Experiments

In this section, regarding the behavior log anomaly detection solution based on LLM fine-tuning proposed for insider threat scenarios in this paper, we pose the following three research questions:

RQ1: What is the performance of the detection model based on LLM fine-tuning? How does it compare with the baselines in the field of insider threat detection?

RQ2: How do the hyperparameters involved in the solution affect detection performance?

RQ3: How accurate is the fine-grained threat tracing in behavior-level threat capturing? What do the tracing results reveal about our detection method?

We address the above research questions through experiments and case studies.

# of Users	**# of Insiders**	**# of Insider Insts.**	**# of Threat Insts.**
4,000	5	1,360	33

Table 1: Statistics of CERT v6.2 Dataset.

4.1 Experimental Settings

4.1.1 Dataset

The CMU CERT v6.2 dataset (Lindauer et al., 2014) is a simulated dataset developed by the CERT division of Carnegie Mellon University for insider threat detection. It includes behavioral logs from 4,000 employees over 516 days, covering logon, HTTP, file, email, and device logs, among others. There are 5 insiders included, each corresponding to different insider threat scenarios.

In this paper, we process each user's daily behavioral logs as a single data instance. Instances containing threatening behaviors are labeled as threats. We filter out instances containing fewer than 5 behavioral sentences. The dataset size is shown in Table 1.

4.1.2 Baselines

We select 8 baselines for comparison experiments, including 2 commonly used machine learning models for outlier detection, iForest and OCSVM; 4 representative anomaly-based works in the field of insider threat detection, including Tuor et al.'s work (Tuor et al., 2017), Meng et al.'s work (Meng et al., 2020), Yu et al.'s work (Yu et al., 2022), and Song et al.'s work (Song et al., 2024b); and 2 representative classification-based works which are Yuan et al.'s work (Yuan et al., 2020) and Wu et al.'s work (Wu and Li, 2021).

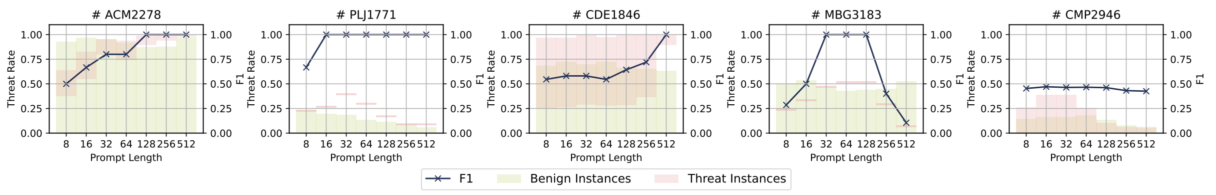

Figure 4: Experimental results for 5 insiders with varying prompt lengths in P-tuning v2. Green bars indicate threat rate ranges for benign instances; pink bars indicate threat rate ranges for threat instances.

Pre-trained Model	Parameter Size	P	Recall	F1
BERT-Large	340M	0.1030	0.8000	0.1672
ChatGLM-6B	6B	0.8667	0.9600	0.8941

Table 2: Comparison results between 2 pre-trained models.

4.1.3 Evaluation Metrics

We use P (Precision), Recall, F1, and Acc (Accuracy) as evaluation metrics. P, Recall, and F1 assess the models' ability to identify positive instances, which is crucial when insider threats are rare. Acc measures overall classification accuracy.

To more strictly compare model precision, we set the threshold for anomaly detection to the minimum non-zero threat rate of the ground truth threat instances.

4.1.4 Implementation Details

In the experiments, we utilize ChatGLM-6B (Du et al., 2022) as the pre-trained LLM. The insiders in the dataset are regarded as users to be tested, while another 200 users are randomly selected as non-tested users. Each user's first 250 days of logs (or 150 days if fewer are available) serve as training instances. The remaining logs of the insiders serve as experimental subjects for testing, with all their training instances as control subjects (where N equals the size of training instances).

The hyperparameters are set as follows: $\alpha_{\text{train}} = 0.8$, $\alpha_{\text{test}} = 0.3$, lora_rank = 8, batch_size = 32, epochs = 15, learning_rate = 1e-4, K=5,000.

To achieve better identification results, we use ChatGLM-3-turbo for free text topic specification on the users being tested. To conserve LLM resources, we randomly insert topic keywords into the data of non-tested users instead.

4.2 Comparison Experiments(RQ1)

ITDLM-0, ITDLM-U, and ITDLM-II perform individual detection for each insider and average the evaluation results. Detection performance comparison is shown in Figure 5. ITDLM-II combines generalization and specificity, achieving optimal detection performance. ITDLM-0, lacking specific user information, has weaker detection performance but still can detect threat instances, validating the approach of distinguishing user behavior patterns in pairs. ITDLM-U, learning only the behavioral patterns of the specific user to be tested, shows improved P over ITDLM-0 but suffers from overfitting, limiting its performance.

Scheme	P	F1
iForest	0.4549	0.5510
OCSVM	0.2471	0.3741
Tuor et al. (Tuor et al., 2017)	0.3827	0.4805
Meng et al. (Meng et al., 2020)	0.4471	0.5406
Yu et al. (Yu et al., 2022)	0.3850	0.4850
Song et al. (Song et al., 2024b)	0.8072	0.8540
Our ITDLM-II	**0.8667**	**0.8941**

Table 3: Comparison results between our solution and 6 anomaly-based baselines.

Scheme	# of Threat Insts.	P	Recall	F1
Yuan et al. (Yuan et al., 2020)	8	0.8824	0.3750	0.5263
	10	0.7333	0.5789	0.6471
	15	**0.9200**	0.6970	0.7931
Wu & Li (Wu and Li, 2021)	15	0.9091	0.6061	0.7273
Our ITDLM-II	0	0.8667	**0.9600**	**0.8941**

Table 4: Comparison results between our solution and 2 classification-based baselines. Column # of Threat Insts. represents the number of threat instances used for training.

We also compare the performance of the models fine-tuned with BERT-Large and ChatGLM-6B in Table 2. ChatGLM-6B's larger parameter scale leads to superior detection performance, making it more effective for insider threat detection.

The comparison results with the baselines in Table 3 and Table 4 show that our solution possesses the best comprehensive detection performance, with the F1 score outperforming all baselines. Our P surpasses all anomaly-based baselines

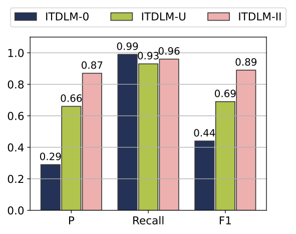

Figure 5: Comparison results between ITDLM-0, ITDLM-U, and ITDLM-II.

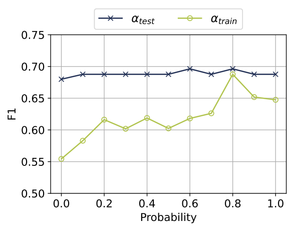

Figure 6: Experimental results with varying α_{train} and α_{test}.

Mean Centering	Acc	P	Recall	F1
Centered	0.8731	0.7945	0.8234	0.7792
Non-Centered	0.7318	0.5160	0.8724	0.5997

Table 5: Performance of threat tracing mechanisms under mean centering and uncentering.

and approaches that of classification-based baselines, while our Recall exceeds all classification-based baselines.

It should be noted that the detection results of anomaly-based baselines fall within the (0,1) range and hardly ever reach 0, making their recall consistently 1 in our evaluation system. Thus, we omit recall scores for anomaly-based baselines.

4.3 Hyperparametric Analysis(RQ2)

We evaluate the detection performance of ITDLM-U under varying α_{train} and α_{test} parameters, as shown in Figure 6. Both curves exhibit an overall upward trend, indicating that behavioral paragraph augmentation improves detection performance. The steeper α_{train} curve suggests that data augmentation during training has a greater impact on performance.

Figure 4 shows the detection performance of ITDLM-II across 5 insiders with varying prompt lengths in P-tuning v2. The F1 score generally increases as prompt length grows (ACM2278, PLJ1771, CDE1846) or initially rises then declines (MBG3183, CMP2946). The figure indicates that F1 peaks when the prompt length matches the user's behavioral complexity, maximizing the differentiation between benign and threat instances. Short prompts lead to underfitting, while excessively long prompts may cause overfitting.

4.4 Verification of Threat Tracing and Case Study(RQ3)

The proposed fine-grained threat tracing mechanism analyzes each true positive instance and averages the evaluation results. Table 5 presents the performance comparison between the mean-centered and non-centered schemes. Mean centering significantly improves P, Acc, and F1, with only a slight decrease in Recall.

Upon statistical analysis, each behavioral paragraph averages 14.35 behavioral sentences. With an Acc of 0.8731, about 12.53 sentences are correctly classified, with fewer than 2 misclassified on average, which is acceptable in practice.

Table 6 displays two threat tracing instances from insider ACM2278. The table shows that threat behaviors traced by our solution basically align with the ground truth. Interestingly, however, our solution fails to detect the user's log-off behaviors after "Go off duty" in both instances, and even misidentifies a log-off before "Go off duty" as a threat. This judgment is reasonable from the perspective of paragraph pattern identification: a log-off sentence is normal at the end of a paragraph but abnormal in the middle. However, from a behavioral logic perspective, the judgment is incorrect: log-off behavior is normal before "Go off duty" and abnormal afterward. This suggests that our solution lacks an understanding of human behavioral logic, an issue also prevalent in existing insider threat detection studies.

#	Review Behavior Paragraph
1	Log on with own pc... View external email on own pc for 3 times. Log off with own pc. *Go off duty. Log on with own pc. Connect the device on own pc. Open file from usb on own pc for 4 times. Upload information on harmful pages on own pc for 4 times. Disconnect the device on own pc. Log off with own pc.*
2	Log on with own pc... Send external email on own pc. Log off with own pc. *Go off duty. Log on with own pc. Connect the device on own pc. Upload information on harmful pages on own pc for 3 times. Open file from usb on own pc for 3 times. Disconnect the device on own pc. Log off with own pc.*

Table 6: Case study of insider ACM2278. Sentences identified as threats are highlighted in red, with ground truth threat sentences italicized. Column # represents the serial number of the instances.

5 Conclusion

In this paper, we introduce a precise behavioral anomaly detection solution tailored for insider threats based on LLM fine-tuning. By representing user behaviors in natural language, we reduce information loss and improve behavioral data readability. We fine-tune the LLM with a user behavior pattern contrastive task to detect anomalous behavior patterns. We propose three fine-tuning strategies: ITDLM-0, ITDLM-U, and ITDLM-II. The two-stage fine-tuning strategy, ITDLM-II, achieves the highest F1 score of 0.8941 among all baselines. Additionally, we implement a fine-grained threat tracing mechanism to capture threats at the behavioral level, achieving an Acc of 0.8731.

6 Limitation

The CERT v6.2 dataset uses outdated text generation techniques, leading to poor-quality free text that prevents LLMs' sentiment analysis from functioning and reduces the accuracy of topic specification, thereby limiting our approach's performance. ITDLM-0, which is better suited to the characteristics of large models that achieve high generalization through massive data, has the potential to develop a highly accurate universal insider threat detection model. But this potential is limited by our experimental environment and dataset quality. Additionally, our solution lacks comprehension of actual human behavioral logic, as noted in the case study.

References

Manqing Dong, Hao Huang, and Longbing Cao. 2024. Can LLMs Serve As Time Series Anomaly Detectors? *Preprint*, arXiv:2408.03475.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. GLM: General Language Model Pretraining with Autoregressive Blank Infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335, Dublin, Ireland. Association for Computational Linguistics.

Chris Egersdoerfer, Di Zhang, and Dong Dai. 2023. Early Exploration of Using ChatGPT for Log-based Anomaly Detection on Parallel File Systems Logs. In *Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing*, pages 315–316. ACM.

Duc C. Le, Nur Zincir-Heywood, and Malcolm Heywood. 2021. Training regime influences to semi-supervised learning for insider threat detection. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 13–18, San Francisco, CA, USA. IEEE.

Aodong Li, Yunhan Zhao, Chen Qiu, Marius Kloft, Padhraic Smyth, Maja Rudolph, and Stephan Mandt. 2024. Anomaly Detection of Tabular Data Using LLMs.

Lingli Lin, Shangping Zhong, Cunmin Jia, and Kaizhi Chen. 2017. Insider Threat Detection Based on Deep Belief Network Feature Representation. In *2017 International Conference on Green Informatics (ICGI)*, pages 54–59.

Brian Lindauer, Joshua Glasser, Mitch Rosen, Kurt C Wallnau, and L ExactData. 2014. Generating test data for insider threat detectors. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, 5(2):80–94.

Chen Liu, Shibo He, Qihang Zhou, Shizhong Li, and Wenchao Meng. 2024. Large Language Model Guided Knowledge Distillation for Time Series Anomaly Detection. *Preprint*, arXiv:2401.15123.

Fucheng Liu, Yu Wen, Dongxue Zhang, Xihe Jiang, Xinyu Xing, and Dan Meng. 2019. 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*, pages 1777–1794, London United Kingdom. ACM.

Weizhi Meng, Wenjuan Li, Yu Wang, and Man Ho Au. 2020. Detecting insider attacks in medical cyber-physical networks based on behavioral profiling. *Future Generation Computer Systems*, 108:1258–1266.

Rida Nasir, Mehreen Afzal, Rabia Latif, and Waseem Iqbal. 2021. Behavioral Based Insider Threat Detection Using Deep Learning. *IEEE Access*, 9:143266–143274.

Tabish Rashid, Ioannis Agrafiotis, and Jason R.C. Nurse. 2016. A New Take on Detecting Insider Threats: Exploring the Use of Hidden Markov Models. In *Proceedings of the 8th ACM CCS International Workshop on Managing Insider Security Threats*, pages 47–56, Vienna Austria. ACM.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM.

Charlie Soh, Sicheng Yu, Annamalai Narayanan, Santhiya Duraisamy, and Lihui Chen. 2019. Employee profiling via aspect-based sentiment and network for insider threats detection. *Expert Systems with Applications*, 135:351–361.

Chengyu Song, Linru Ma, Jianming Zheng, Jinzhi Liao, Hongyu Kuang, and Lin Yang. 2024a. Audit-LLM: Multi-Agent Collaboration for Log-based Insider Threat Detection. *Preprint*, arXiv:2408.08902.

Shuang Song, Neng Gao, Yifei Zhang, and Cunqing Ma. 2024b. BRITD: behavior rhythm insider threat detection with time awareness and user adaptation. *Cybersecurity*, 7(1):2.

Aaron Tuor, Samuel Kaplan, Brian Hutchinson, Nicole Nichols, and Sean Robinson. 2017. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*.

Chunhui Wu and Wenjuan Li. 2021. Enhancing intrusion detection with feature selection and neural network. *International Journal of Intelligent Systems*, 36(7):3087–3105.

Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive Learning for Sequential Recommendation. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 1259–1273. ISSN: 2375-026X.

Jongmin Yu, Hyeontaek Oh, Minkyung Kim, and Sehun Jung. 2022. Unusual Insider Behavior Detection Framework on Enterprise Resource Planning Systems Using Adversarial Recurrent Autoencoder. *IEEE Transactions on Industrial Informatics*, 18(3):1541–1551. Number: 3.

Shuhan Yuan, Panpan Zheng, Xintao Wu, and Hanghang Tong. 2020. Few-shot Insider Threat Detection. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2289–2292, Virtual Event Ireland. ACM.

Yifei Zhang, Neng Gao, and Cunqing Ma. 2023. Learning to Select Prototypical Parts for Interpretable Sequential Data Modeling. 37(5):6612–6620.

A Glossary

As shown in Table 7, the terms and corresponding explanations related to the field of insider threat detection in this paper are presented.

B Technical Principle and Details of Referenced Works

We reference several techniques from previous works to enhance our insider threat detection. Song et al.'s user-adaptive time slicing (Song et al., 2024b) is adopted to incorporate behavioral time distribution information into our behavioral paragraphs, while Xie et al.'s data augmentation algorithms (Xie et al., 2022) are adopted to augment behavioral paragraphs for pattern contrast. Ribeiro et al.'s local model-agnostic explanation technique (Ribeiro et al., 2016) supports our fine-grained threat tracing. This section details these techniques and their underlying principles.

B.1 User-Adaptive Time Slicing

Song et al. (Song et al., 2024b) propose that when time slicing aligns with a user's behavioral temporal distribution, behavior within each slice remains stable, while significant variation occurs across slices.

Following their work, we divide the 24-hour day into T time slices of granularity G, where $TG = 24h$. Each slice spans from $b + Gt$ to $b + G(t + 1)$, with b as the bias and t as the slice index. $G = 1, 2, 3, 4, 6, 8, 12, 24h$, and $b = 0, 1, \ldots, (G - 1)h$. For each (G, b) set, we calculate the categorical count vector $x_t^{(G,b)}$ of user behaviors within each slice, covering 129 behavior categories. The vectors form a matrix $\mathbf{x}^{(G,b)} = (x_1^{(G,b)}, x_2^{(G,b)}, \ldots, x_t^{(G,b)}, \ldots, x_T^{(G,b)})$ chronologically. We calculate the covariance $c^{(G,b)} = \text{cov}(\mathbf{x}^{(G,b)})$ to measure the alignment between time slicing and the user's behavioral temporal distribution. The average covariance $\bar{c}^{(G,b)}$ is calculated for a specific user, and the (G, b) with the highest $\bar{c}^{(G,b)}$ is considered optimal.

Based on Song et al.'s findings and our experience, the optimal G is typically 12 hours, dividing the day into working and leaving periods. Therefore, in our scheme, we set G to 12 hours and determine the optimal bias b for each user by calculating covariance.

B.2 Data Augmentation

Drawing on Xie et al.'s work (Xie et al., 2022), we employ cropping, masking, and reordering operations to augment behavioral paragraphs for user behavioral pattern contrast.

The operation $a_{crop}(\cdot)$ randomly takes a continuous sub-paragraph with $n_c = \lfloor \eta \times n \rfloor$ sentences from paragraph p:

$$a_{crop}(p) = (s_c, s_{c+1}, \ldots, s_{c+n_c-1}) \quad (10)$$

The operation $a_{mask}(\cdot)$ randomly masks $n_m = \lfloor \gamma \times n \rfloor$ behavioral sentences in paragraph p. Let $\mathcal{I}_p = (i_1, i_2, \ldots, i_j, \ldots, i_{n_m})$, where i_j represents the index of a masked behavioral sentence.

$$a_{mask}(p) = (\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_i, \ldots, \hat{s}_n) \quad (11)$$

$$\hat{s}_i = \begin{cases} s_i, & i \notin \mathcal{I}_p, \\ [MASK], & i \in \mathcal{I}_p. \end{cases} \quad (12)$$

The operation $a_{reorder}(\cdot)$ randomly selects a continuous sub-paragraph of $n_r = \lfloor \beta \times n \rfloor$ behavioral sentences from paragraph p, denoted as $(s_r, s_{r+1}, \ldots, s_{r+n_r-1})$, and randomly shuffles them into $(\hat{s}_r, \hat{s}_{r+1}, \ldots, \hat{s}_{r+n_r-1})$. That is:

$$a_{reorder}(p) = (s_1, \ldots, \hat{s}_r, \ldots, \hat{s}_{r+n_r-1}, \ldots, s_n) \quad (13)$$

Terms	Explanations
Insider Threat	Employees misuse their legal permissions and trust to compromise the confidentiality, integrity, and availability of organizational assets.
User vs. Insider	In this paper, "user" refers to all organizational employees who interact with the organization's system, while "insider" specifically denotes those employees whose actions pose insider threats.
User Behavior Logs / Behavior Logs	In this paper, user behavior logs refer to structured datasets that record users' operations and activities within an organization's system or network environment.
User Behavior Patterns	The regular behavioral characteristics or time distribution of users in specific environments. In this paper, it refers to the typical interactions of users with organizational systems or network environments.
Anomaly-Based Detection / Anomaly Detection	The process of modeling benign behavior patterns and evaluating the deviation of test data from those patterns to identify anomalies.
Benign Anomalies vs. Threats	In this paper, benign anomalies refer to deviations from benign behavior patterns that do not fundamentally harm organizational assets, while threats are behaviors that may cause substantial harm to organizational assets.
Threat Tracing	The process of pinpointing specific behaviors or operations that lead to anomalies within a detected sequence or set of behaviors.
Time Slicing	The segmentation of continuous user behavior data into distinct time-based segments to capture the temporal distribution and variation of behaviors.

Table 7: The terms and corresponding explanations related to the field of insider threat detection.

B.3 Local Interpretable Model-agnostic Explanations

Ribeiro et al. (Ribeiro et al., 2016) suggest training interpretable local surrogate models, such as linear models or decision trees, to explain individual samples in complex models. We perturb a behavior paragraph π by randomly selecting an arbitrary number of behavior sentences, generating nearby samples π_{v_i}, where v_i is the selection vector with $v_i \in \mathcal{V}$. The complex detection model produces detection results of perturbed samples. Perturbed samples are weighted based on their proximity to π, with closer samples given higher weight. The sample weights are calculated using cosine distance and the Gaussian kernel function. We train a linear surrogate model on perturbed samples to approximate the complex model locally, and thus achieve the interpretation of the behavior paragraph π.

It is worth noting that many explanation algorithms can achieve fine-grained threat tracing within our framework. In fact, insider threats often involve multiple related behaviors rather than existing in isolation. For instance, information leaks may consist of after-hours logins, file copying, and uploading to WikiLeaks. Therefore, explanation algorithms based on prototypical concepts (Zhang et al., 2023) have the potential to perform better in fine-grained threat tracing.

Prompts for Insider Threat Detection:

Instruction: Are behavioral sequences A and B from the same user?
Input: Behavioral sequence A is as follows: *[$\hat{p}_a$]* SEP Behavioral sequence B is as follows: *[$\hat{p}_b$]*
Answer:

Prompts for URL Topic Specification:

Determine which category the following domain belongs to: Harmful Information or Others. Note: Content that may lead to the leakage of information from government, enterprises, or organizations is also considered Harmful Information; domain names of piracy websites fall under Others; random strings are not necessarily classified as Harmful Information. If you are unsure, classify it as Others. Your response must follow the format: [answer]. Please provide the answer directly.
The domain to be tested is: *[domain]*
Your answer:

Prompts for Topic Specification of Web Content and Email Content:

Determine which category the following content belongs to: Job Application or Others. Your response must follow the format: [answer]. Please provide the answer directly.
The content to be tested is: *[content]*
Your answer:

Figure 7: Prompts used in our solution.

C Prompts Used in Our Solution

As shown in Figure 7, the prompts used in our solution are presented, including prompts for insider threat detection, prompts for URL topic specification, and prompts for topic specification of web content and email content.

D Baselines and Evaluation Metrics

D.1 Introduction to Baselines

We briefly introduce the 8 baselines as follows.

1) Anomaly-based Baselines Anomaly-based baselines include iForest, OCSVM, Tuor et al.'s work (Tuor et al., 2017), Meng et al.'s work (Meng et al., 2020), Yu et al.'s work (Yu et al., 2022), and Song et al.'s work (Song et al., 2024b), totaling six methods.

- **iForest and OCSVM** are common outlier detection models. We construct input vectors using traditional behavior classification and counting methods. Behavior counts are obtained within a one-day window covering 129 categories. We utilize GridSearchCV from sklearn to determine the hyperparameters for both models.

- **Tuor et al.** extract 408 behavior categories from logs, obtaining user-day count vectors. They utilize Deep Neural Network (DNN) to reconstruct the count vectors and fit a multivariate Gaussian distribution to compute decomposable anomaly scores.

- **Meng et al.** use Euclidean distance and Euclidean norm to measure differences between two behavioral profiles. The larger difference between the test profile and benign profile indicates higher anomaly levels.

- **Yu et al.** use an RNN-based autoencoder to reconstruct behavioral sequences for anomaly detection. They utilize adversarial learning to align hidden layer features with a normal distribution, thereby reducing feature space uncertainty.

- **Song et al.**, based on traditional behavior counting, construct feature sequences considering covariances for user adaptation. Stacked bidirectional LSTMs and feedforward neural networks are used for detection.

2) Classification-based Baselines Classification-based baselines include Yuan et al.'s work (Yuan et al., 2020) and Wu et al.'s work (Wu and Li, 2021), totaling two methods.

- **Yuan et al.** encode behavior category sequences using a Transformer, pre-training with the MLM and fine-tuning for binary classification of behavior sequence representations.

- **Wu et al.** first employ the Fast Correlation-Based Filter (FCBF) for feature selection from activity features in system logs. They then combine neural networks and random forests to detect threats.

D.2 Introduction to Evaluation Metrics

We select Precision (P), Recall, F1, and Accuracy (Acc) as our evaluation metrics. Let TP (True Positives) represent the number of threat instances predicted as positive; FP (False Positives) represent the number of benign instances predicted as positive; TN (True Negatives) represent the number of benign instances predicted as negative; FN (False Negatives) represent the number of threat instances predicted as negative.

P is the proportion of actual threat instances among the predicted positive instances:

$$P = \frac{TP}{TP + FP} \tag{14}$$

Recall is the proportion of correctly predicted positive instances among all actual threat instances, which is:

$$Recall = \frac{TP}{TP + FN} \tag{15}$$

The F1 score is the harmonic mean of P and Recall, providing a balanced metric that considers both precision and recall:

$$F1 = 2 \times \frac{P \times Recall}{P + Recall} \tag{16}$$

Acc measures the overall proportion of correct predictions and is more reflective of model performance when the dataset is balanced.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{17}$$

E ITDLM-II vs. ITDLM-U: Experimental and Case Study Analysis of Threat Differentiation Ability

As shown in Figure 8, ITDLM-U suffers from overfitting, as evidenced by its tendency to produce extreme threat rates of 1 or 0, which prevents it from distinguishing between benign anomalies and threats, resulting in a high false alarm rate. In

#	Label	$r_{\text{ITDLM-U}}$	$r_{\text{ITDLM-II}}$	Review Behavior Paragraph
1	threat	1.0000	1.0000	...Go off duty. Log on with own pc. Connect the device on own pc. Open file from usb on own pc for 4 times. Upload information on harmful pages on own pc for 4 times. Disconnect the device on own pc. Log off with own pc.
2	threat	1.0000	0.9396	...Go off duty. Log on with own pc. Connect the device on own pc. Upload information on harmful pages on own pc for 3 times. Open file from usb on own pc for 3 times. Disconnect the device on own pc. Log off with own pc.
3	benign	1.0000	0.8792	...Go off duty. Log on with own pc. Connect the device on own pc for 2 times. Delete file from usb on own pc for 3 times. Copy file from usb on own pc for 2 times. Copy file to usb on own pc for 3 times. Open file from usb on own pc. Disconnect the device on own pc for 2 times. Write file to usb on own pc. Log off with own pc.
4	benign	1.0000	0.6040	...Go off duty. Log on with own pc. Connect the device on own pc for 3 times. Disconnect the device on own pc for 3 times. Delete file from usb on own pc for 3 times. Copy file from usb on own pc. Open file from usb on own pc for 2 times. Open decoy file in disk on own pc. Log off with own pc.
5	benign	1.0000	0.3087	...Go off duty. Log on with own pc. Connect the device on own pc for 4 times. Open file from usb on own pc. Copy file to usb on own pc for 3 times. Disconnect the device on own pc for 4 times. Copy file from usb on own pc for 2 times. Log off with own pc.

Table 8: Case study of insider ACM2278 under ITDLIM-U and ITDLIM-II. Column # represents the serial number of the instances.

contrast, the predictions from ITDLM-II are much smoother.

Table 8 shows the detection results of ITDLM-U and ITDLM-II for five test instances of user ACM2278, including two threats and three benign instances. According to the CERT descriptions, the insider threat scenario for ACM2278 is:

User who did not previously use removable drives or work after hours begins logging in after hours, using a removable drive, and uploading data to wikileaks.org.

Each instance in the table involves '*work after hours*' (e.g., "Log on" after "Go off duty"). However, the benign instances do not include '*uploading data to wikileaks.org.*'

From ITDLM-U's perspective, all five instances deviate from ACM2278's past behavioral patterns and are marked as anomalies with a threat rate of 1. In contrast, ITDLM-II identifies the two instances involving *'uploading data to wikileaks.org'* as more anomalous, assigning them higher threat rates. The case study demonstrates that ITDLM-II can better distinguish between benign anomalies and threats.

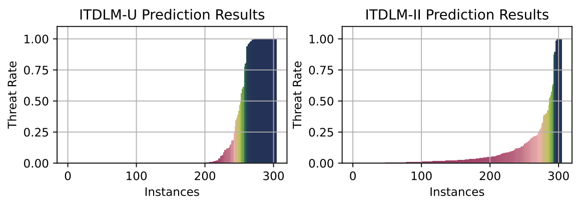

Figure 8: Threat rate distribution of instances to be tested under ITDLM-II and ITDLM-U.