Few-shot Insider Threat Detection

Shuhan Yuan
Utah State University
Logan, UT, USA
shuhan.yuan@usu.edu

Panpan Zheng
University of Arkansas
Fayetteville, AR, USA
pzheng@uark.edu

Xintao Wu
University of Arkansas
Fayetteville, AR, USA
xintaowu@uark.edu

Hanghang Tong
University of Illinois at Urbana-Champaign
Urbana, IL, USA
htong@illinois.edu

ABSTRACT

Insiders cause significant cyber-security threats to organizations. Due to a very limited number of insiders, most of the current studies adopt unsupervised learning approaches to detect insiders by analyzing the audit data that record information about employees' activities. However, in practice, we do observe a small number of insiders. How to make full use of these few observed insiders to improve a classifier for insider threat detection is a key challenge. In this work, we propose a novel framework combining the idea of self-supervised pre-training and metric-based few-shot learning to detect insiders. Experimental results on insider threat datasets demonstrate that our model outperforms the existing anomaly detection approaches by only using a few insiders.

CCS CONCEPTS

• Security and privacy → Intrusion/anomaly detection and malware mitigation; **• Computing methodologies → Neural networks**.

KEYWORDS

insider threat detection; few-shot learning; cyber-security

ACM Reference Format:
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 and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3340531.3412161

1 INTRODUCTION

A malicious insider indicates an employee who intentionally used his authorized access in a manner that negatively affected the confidentiality, integrity, or availability of the organization's information. Since the insiders' behaviors are different from behaviors of legitimate employees, we can detect insider threats by analyzing employees' behaviors via audit data. In general, user activities are often grouped into sessions that are separated by operations like "LogOn" and "LogOff". However, due to the small number of malicious sessions, supervised learning algorithms cannot be employed. Hence, how to leverage the limited insiders to improve the performance of insider detection is an interesting problem.

To tackle this challenge, in this paper, we propose a framework of combining the idea of self-supervised pre-training [1] and metric-based few-shot learning to detect insiders [4]. Specifically, we first design input representations of user activities in sessions, which capture both activity type and time information, and then adopt the transformer layer proposed in [6] to learn session representations. In order to achieve insider threat detection with only a small number of insiders, our framework is trained by two phases. The first phase is to pre-train the transformer layer by learning an "activity model" on a large number of user sessions. The pre-trained activity model provides strong prior knowledge on how the user sessions are composed. Then, the second phase is to fine-tune the model and learn a similarity function via few-shot learning where the objective is to separate the normal and malicious sessions in the embedding space. After training, new malicious sessions can be detected by having high similarity scores to the observed malicious sessions.

2 FRAMEWORK

We model a user's behavior as a sequence of activities that can be extracted from various types of raw data, such as user logins, emails, and web browsing. Formally, we model the up-to-date activities of a user as a sequence of sessions $U = \{S_1, \cdots S_k, \cdots\}$ where $S_k = \{e_{k_1}, \cdots, e_{k_j}, \cdots, e_{k_T}\}$ indicates the k-th activity session. One session in our scenario is a sequence of activities starting with 'LogOn" and ending with "LogOff". $e_{k_j} = (t_{k_j}, a_{k_j})$ denotes the j-th activity in the user's k-th session and contains activity type a_{k_j} and occurred time t_{k_j}.

Given an extremely unbalanced training set $\mathcal{D} = \{(S_i, y_i)\}_{i=1}^{m}$, where S_i is the i-th session, and $y_i \in \{0, 1\}$ indicates malicious or not of the session, there is a very small number of sessions that are labeled as malicious in $\mathcal{D}$. Note that we consider sessions that contain malicious activities, such as uploading documents to Wikileak, as malicious sessions. The goal of learning in our insider threat detection is to predict whether a new session $S_k = \{e_{k_1}, \cdots, e_{k_j}, \cdots, e_{k_T}\}$ is normal or malicious. To address the challenge that there are usually very few records of known insider

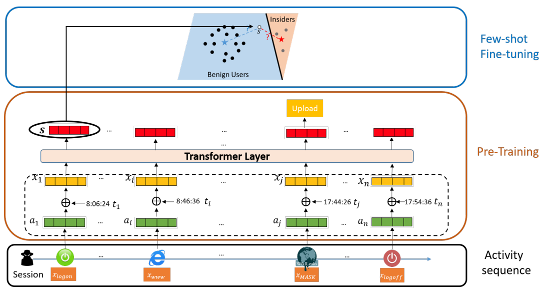


Figure 1: The architecture of our framework

attacks in the training data, we propose a framework that takes advantages of both self-supervised pre-training and metric-based few-shot learning. The pre-training phase uses the user activity sequences to learn session representations by self-supervised learning, while the fine-tuning phase is to train a feed-forward neural network with small parameters to derive the similarity scores among samples. Figure 1 shows the architecture of our framework.

Specifically, we design input representations for activities in a session by combining the activity type and time information. In order to model user sessions, we adopt the transformer layer and use the similar idea as Bidirectional Encoder Representations from Transformers (BERT) to pre-train the model [1]. After pre-training, we fine-tune the model based on a few observed insiders to achieve malicious session detection. The metric-based few-shot learning computes the centers of normal and malicious session representations as prototype representations of normal and malicious sessions and then trains a neural network as a similarity function to compute similarity scores between user sessions and prototype representations. Then, new malicious sessions expect to be detected by having high similarity scores to the observed malicious sessions.

2.1 Self-supervised Pre-training

Input representation. Input representations aim to map user activities in a session into an embedding space so that the transformer layer can encode the user activities into a session representation. Given a user activity in a session, we have both activity type and time information. The existing works only consider the activity type information, like copying documents to a removable disk. However, the timing of a user activity is also an important feature for malicious session detection. For example, an insider may copy classified documents to a removable disk at midnight. Since both activity type and time are important for insider threat detection, input representations should encode both type and time information of user activities to our model. To this end, given an activity type and its corresponding time, we map them to an embedding space. Then, the input representation of an activity is constructed by summing the representations of type and time. The detailed descriptions of the activity type and time representations are given as follows.

Type representations: To obtain type representations, we first consider each activity type as a word in a sentence, each session as a sentence, and all activity sessions as a text corpus. Then, we adopt the word2vec to train type representations $\mathbf{A} \in \mathbb{R}^{a*d}$, where a is the number of activity types.

Time representations: Similar to the position encodings [6], which aim to encode the relative or absolute position information to the model, we propose the time representations to inject the absolute time information of an activity to the input representation. Specifically, we represent the activity time as the offset minute from 12:00 am. As a result, the time representations can be represented as $\mathbf{T} \in \mathbb{R}^{1440*d}$, where 1440 is the total number of minutes in a day. There are two advantages to using absolute time representations. First, we can inject the physical time information to the model. Second, since the transformer layer do not have recurrent structure, the absolute time can also capture the order information of activities in a session. We adopt the same sinusoid function as the position encoding in [6] to generate the time representations.

Finally, the input representation of the j-th activity is defined as:

$$\mathbf{x}_j = \mathbf{a}_{a_j} + \mathbf{t}_{t_j}, \tag{1}$$

where a_j and t_j indicate the indices of representations of type and time given the j-th activity. The input representation to the transformer can be represented as $\mathbf{X}^{n*d}$, where n indicates the length of session. With the well-designed input representation, the transformer layer can encode multiple aspects of user behaviors.

Pre-training. We adopt the transformer layer to model the user activity sessions [6] and use a similar strategy as BERT to pre-train the transformer layer [1]. In our scenario, we consider the log files that record all the user activities as a pre-training corpus and adopt the masked language model to pre-train the transformer layer. Specifically, the model takes all activities in a session with random masks as inputs, where we randomly replace a small ratio of activities in a session with a specific MASK token. The training object is to accurately predict the randomly masked activity types. The purpose of predicting MASK token is to make the transformer layer capture the user behaviors in terms of activity types and time. Since the behaviors of normal and malicious sessions are different, we expect the transformer layer could encode the prior knowledge of user behavior by training to predict the MASK tokens.

After pre-training, similar to BERT, we consider the final hidden state of the first activity in the session, i.e., the hidden state of "LogOn", as the session representation $\mathbf{s}$:

$$\mathbf{s} = \text{transformer\_layer}(\mathbf{X})[0], \tag{2}$$

where $\mathbf{X} \in \mathbb{R}^{n*d}$ is the input representation of a session.

2.2 Few-shot Fine-tuning

After self-supervised pre-training, the session representations derived from the transformer layer capture the information of user behaviors in terms of activity types and time. We further fine-tune the session representations and design a similarity function to detect insider threats via a small number of malicious sessions. Concretely, the few-shot learning phase consists of two goals. The first goal is to fine-tune the transformer layer to make the representations of normal and malicious sessions locate separately in the embedding space. Then, we can derive prototype representations of normal and malicious sessions by adopting the mean operation so that each prototype representation is surrounded by sessions in that class. The second goal is to derive a similarity function to evaluate the similarity between a session and prototype representations. After fine-tuning, a new malicious session is expected to have a higher similarity score to the malicious prototype representation.

In few-shot learning, each training iteration is formulated as a training episode. In each episode, we randomly sample k normal sessions and k malicious sessions as the support set $\mathcal{S} = \{(S_i, y_i)\}_{i=1}^{2*k}$, and further sample q samples from both normal and malicious sessions as the query set $Q = \{(S_j, y_j)\}_{j=1}^{q}$ from the training set. We then compute the representation of each session in the support and query sets by Equation 2. We denote the representations of S_i from the support set and S_j from the query set as $\mathbf{s}_i$ and $\mathbf{s}_j$.

Based on the support set, we derive prototype representations for normal and malicious sessions by a mean operation over representations of normal and malicious sessions separately:

$$\mathbf{c}_y = \frac{1}{k} \sum_{(S_i, y_i) \in \mathcal{S}_y} \mathbf{s}_i, \tag{3}$$

where $\mathcal{S}_y$ indicates the set of samples with label y and $y \in \{0, 1\}$.

After obtaining two prototype representations from the support set, we further derive the similarity scores of samples in the query set with the representation of each prototype. To this end, we first combine the representation of a query sample with a prototype representation by a concatenate operation $\mathbf{r}_{jy} = Concat(\mathbf{s}_j, \mathbf{c}_y)$. Then, we adopt a fully connected neural network as a similarity function to map the representation $\mathbf{r}_{jy}$ to a similarity score l_{jy}:

$$l_{jy} = g(\mathbf{r}_{jy}; \theta_g), \tag{4}$$

where l_{jy} measures the similarity between query j and prototype y ranging from 0 to 1; $g(\cdot)$ is a fully connected neural network parameterized by θ_g.

We adopt the mean square error as the loss function to fine-tune the transformer layer as well as the neural network $g(\cdot)$ [5]:

$$\mathcal{L} = \sum_{y \in \{0,1\}} \sum_{j=1}^{q} (l_{jy} - \mathbb{1}(y_j == y))^2. \tag{5}$$

The general idea is to consider the task as a regression problem. The training procedure is to make the similarity score close to 1 if the query sample and prototype belong to the same class, otherwise, the similarity score should be close to 0. In the fine-tuning phase, we update the parameters θ_g in $g(\cdot)$.

Detection. When deploy the model for malicious session detection, we adopt all the malicious sessions in the training set to compute the prototype representation of malicious sessions and randomly sample the same number of normal sessions to compute the prototype representation of normal sessions. Given an upcoming session as a new query set, we compute its similarity scores with the two prototype representations. The upcoming session will be detected as malicious if its similarity to the malicious prototype is higher than the one to the normal prototype.

3 EXPERIMENTS

3.1 Experimental Setup

Dataset. We evaluate our proposed approach on two datasets.

CERT Insider Threat Dataset (CERT) [2] is the only comprehensive dataset for evaluating the insider threat detection, which consists of records of computer-based activities over 516 days. Table 1 shows the statistics of the dataset. We extract activity types as combinations of activity types and their context, such as "upload to Wikileak.org", which leads to 1435 fine-grained activity types. We split the dataset chronologically into training and testing sets. We use sessions occurred in the first 396 days as the training set and randomly select 10,000 sessions from remaining 120 days as the testing set. There are 15 malicious sessions in the training set and 33 malicious sessions in the testing set.

Table 1: Statistics of two datasets

Dataset	# of Employees	# of Insiders	# of Sessions	# of Malicious Sessions
CERT	4000	5	1,581,358	48
Wikipedia	4073	822	10,113	4627

Training Details. In experiments, the model consists of 2 transformer layers. The multi-head attention sub-layer consists of 4 heads, and the dimension of each attention in multi-head attention sub-layer is 64, so the dimension of feed-forward sub-layer is 256. In the few-shot training phase, in each episode, we randomly select 15 normal sessions and 15 malicious sessions to compose the support set. We augment the malicious sessions by randomly shuffling activities between "LogOn" and "LogOff".

UMDWikipedia Dataset (Wikipedia) [3] contains around 770K edits with 17105 vandals and 17105 benign users (Table 1). We consider user activities in a day as a session. If a session contains activities that are reverted by administrators, the session is malicious. We filter out sessions with activity numbers less than 15.

Training Details. To represent the type information, we adopt 7 binary features: whether or not the user edited on a meta-page; if the edited page is a meta-page, whether or not this meta-page is empty; whether or not the user consecutively edited the pages in less than 1 minute, 3 minutes, or 15 minutes; whether or not the user's current edit page had been edited before; whether or not the current edit will be reverted by the platform. In the few-shot training stage, for each episode, we randomly pick 50 normal sessions and 50 malicious sessions to construct the support set.

Baselines. We compare our model with three baselines, Recurrent Neural Network (**RNN**), One-class SVM (**OCSVM**) and Isolation Forest (**iForest**). We replace the transformer layers with RNN and adopt the same objective function defined in Equation 5 to train the RNN. For OCSVM and iForest, we use activity types to compose the input feature vector, and the value each feature is the number of the corresponding activity in a session.

3.2 Experimental Results

Few-shot Malicious Session Detection. Table 2 shows the precision, recall, F1, and false positive rate (FPR). Compared to baselines, our proposed model achieves the best performance in terms of F1 score and FPR on both datasets. Since OCSVM and iForest only adopt normal sessions for training, these two approaches cannot achieve good performance for malicious session detection. Although OCSVM achieves high recall value, the precision is extremely low. Since the RNN model is also trained in the same setting of few-shot learning, it achieves better performance than OCSVM and iForest. However, the F1 score of RNN is still lower than that of our model. It indicates that using transformer layers with carefully designed input representations to model user sessions can improve the performance of few-shot malicious session detection.

Table 2: Comparison of our framework with baselines

Dataset	Models	Precision	Recall	F1	FPR
CERT	OCSVM	0.0026	0.8182	0.0051	0.1818
	iForest	0.0056	0.3939	0.0110	0.6060
	RNN	0.3038	0.7273	0.4286	0.0055
	Our model	0.9200	0.6970	0.7931	0.0001
Wikipedia	OCSVM	0.5576	0.9870	0.7126	0.7830
	iForest	0.4920	0.1230	0.1968	0.1270
	RNN	0.9548	0.8257	0.8856	0.0393
	Our model	0.9920	0.8626	0.9228	0.0069

Table 3: Performance of our framework trained by various numbers of malicious sessions

Dataset	# of Malicious Sessions	Precision	Recall	F1	FPR
CERT	5	0.0047	0.8140	0.0096	0.7194
	8	0.8824	0.3750	0.5263	0.0001
	10	0.7333	0.5789	0.6471	0.0007
	15	0.9200	0.6970	0.7931	0.0001
Wikipedia	5	0.4994	0.9529	0.6554	0.9497
	15	0.6939	0.8440	0.7616	0.3709
	30	0.9807	0.8637	0.9185	0.0171
	50	0.9920	0.8626	0.9228	0.0069

Various numbers of malicious sessions in the training set. We further evaluate the performance of our model trained by various numbers of malicious sessions in the training set. Table 3 shows the experimental results. Overall, on both CERT and Wikipedia, F1 scores are decreasing as the number of malicious sessions reduces. Meanwhile, we can observe that, even with few malicious sessions, the proposed few-shot insider threat detection model can still achieve a reasonable F1 score and a low false positive rate.

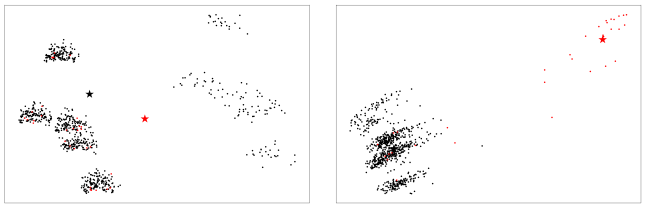
Ablation studies. In order to better understand our framework, we conduct ablation experiments on the CERT dataset. First, the input representations consist of two components, i.e., the representations of activity type and time. We train the model by removing one components at each time, rather than using both components. Meanwhile, since our framework is trained by two phases, we also study the performance without pre-training or without few-shot fine-tuning. In the scenario without few-shot fine-tuning, we adopt the L2-distance as the similarity score to label sessions.

Table 4: Performance after removing various components

	Precision	Recall	F1	FPR
w.o. type representation	0.1013	0.2424	0.1429	0.0070
w.o. time representation	0.8519	0.6970	0.7667	0.0003
w.o. pre-training	0.8519	0.6970	0.7667	0.0003
w.o. few-shot fine-tuning	0.0085	0.5758	0.0168	0.2197

Table 4 shows the experimental results. As we expect, without using the representations of activity types in a session, the model cannot achieve malicious session detection. Meanwhile, by removing the time representations, the F1 score also slightly reduces. For the two training phases, we can also notice that the performance of the model reduces without pre-training, and the model cannot achieve reasonable performance without few-shot fine-tuning.

Visualization. We adopt PCA to project the session embeddings to a two-dimensional space and visualize the normal and malicious sessions. We adopt all malicious sessions and the same number of normal sessions in the training set to compute the prototype embeddings. Then, we select all malicious sessions and randomly choose 800 normal sessions from the testing set. Figure 2 shows the visualization of malicious and normal sessions as well as two prototype representations before and after the few-shot learning phase. We can observe that before the few-shot learning phase, malicious and normal sessions are mixed together, and prototypes of malicious and normal sessions close to each other. After the few-shot learning phase, malicious and normal sessions are separated into two parts, and prototypes of two types of sessions have a long distance in the two-dimensional space.



(a) Before few-shot learning **(b) After few-shot learning**

Figure 2: The visualization of session embeddings. The red and black dots indicate malicious and normal sessions, respectively, while the red and black stars indicate the prototypes of malicious and normal sessions, respectively.

4 CONCLUSION

We have developed a novel framework that consists of two training phases, self-supervised training phase and few-shot learning phase, for insider threat detection. In the pre-training phase, a "masked activity model" is adopted to pre-train the transformer layer so that the model can learn the prior knowledge of user sessions. After that, the few-shot learning phase adopts a few insiders to derive a similarity function that can evaluate the similarity between new user sessions and observed normal or malicious sessions. A session with a high similarity score to the observed malicious sessions can be labeled as malicious sessions. Experimental results demonstrated the effectiveness of our framework.

ACKNOWLEDGMENTS

This work was supported in part by NSF (1564250, 1946391) and the Department of Energy (DE-OE0000779).

REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT 2019*.
[2] J. Glasser and B. Lindauer. 2013. Bridging the Gap: A Pragmatic Approach to Generating Insider Threat Data. In *2013 IEEE Security and Privacy Workshops*.
[3] Srijan Kumar, Francesca Spezzano, and VS Subrahmanian. 2015. Vews: A wikipedia vandal early warning system. In *KDD*.
[4] Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical Networks for Few-shot Learning. In *NIPS*.
[5] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. 2017. Learning to Compare: Relation Network for Few-Shot Learning. In *CVPR*.
[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *NIPS*.