Insider Threat Detection via Hierarchical Neural Temporal Point Processes

Shuhan Yuan, Panpan Zheng, Xintao Wu, Qinghua Li
University of Arkansas, Fayetteville, AR, USA
Email: {sy005, pzheng, xintaowu, qinghual}@uark.edu

Abstract—Insiders usually cause significant losses to organizations and are hard to detect. Currently, various approaches have been proposed to achieve insider threat detection based on analyzing the audit data that record information of the employee's activity type and time. However, the existing approaches usually focus on modeling the users' activity types but do not consider the activity time information. In this paper, we propose a hierarchical neural temporal point process model by combining the temporal point processes and recurrent neural networks for insider threat detection. Our model is capable of capturing a general nonlinear dependency over the history of all activities by the two-level structure that effectively models activity times, activity types, session durations, and session intervals information. Experimental results on two datasets demonstrate that our model outperforms the models that only consider information of the activity types or time alone.

Index Terms—insider threat detection, temporal point process, hierarchical recurrent neural network

I. INTRODUCTION

An insider threat is a malicious threat from people within the organization. It may involve intentional fraud, the theft of confidential or commercially valuable information, or the sabotage of computer systems. The subtle and dynamic nature of insider threats makes detection extremely difficult. The 2018 U.S. State of Cybercrime Survey indicates that 25% of the cyberattacks are committed by insiders, and 30% of respondents indicate incidents caused by insider attacks are more costly or damaging than outsider attacks [1].

Various insider threat detection approaches have been proposed [2]–[8]. However, most of the existing approaches only focus on operation type (web visit, send email, etc) information and do not consider the crucial activity time information. In this paper, we study how to develop a detection model that captures both activity time and type information. In literature, the marked temporal point process (MTPP) is a general mathematical framework to model the event time and type information of a sequence. It has been widely used for predicting the earthquakes and aftershocks [9]. The traditional MTPP models make assumptions about how the events occur, which may be violated in reality. Recently, researchers [10], [11] proposed to combine the temporal point process with recurrent neural networks (RNNs). Since the neural network models do not need to make assumptions about the data, the RNN-based MTPP models usually achieve better performance than the traditional MTPP models.

However, one challenge of applying RNN-based temporal point processes in insider threat detection is it cannot model the time information in multiple time scales. For example, user activities are often grouped into sessions that are separated by operations like "LogOn" and "LogOff". The dynamics of activities within sessions are different from the dynamics of sessions. To this end, we propose a hierarchical RNN-based temporal point process model that is able to capture both the intra-session and inter-session time information. Our model contains two layers of long short term memory networks (LSTM) [12], which are variants of the traditional RNN. The lower-level LSTM captures the activity time and types in the intra-session level, while the upper-level LSTM captures the time length information in the inter-session level. In particular, we adopt a sequence to sequence model in the lower-level LSTM, which is trained to predict the next session given the previous session. The upper-level LSTM takes the first and last hidden states from the encoder of the lower-level LSTM as inputs to predict the interval of two sessions and the duration of next session. By training the proposed hierarchical model with the activity sequences generated by normal users, the model can predict the activity time and types in the next session by leveraging the lower-level sequence to sequence model, the time interval between two consecutive sessions and the session duration time from the upper-level LSTM. In general, we expect our model trained by normal users can predict the normal session with high accuracy. If there is a significant difference between the predicted session and the observed session, the observed session may contain malicious activities from insiders.

Our work makes the following contributions: (1) we develop an insider threat detection model that uses both activity type and time information; (2) we propose a hierarchical neural temporal point process model that can effectively capture two time-scale information; (3) the experiments on two datasets demonstrate that combining the activity type and multi-scale time information achieves the best performance for insider threat detection.

II. RELATED WORK

A. Insider Threat Detection

Much of the research work on the characterization of insiders. Based on the intention of the attack, there are three types of insiders, i.e., *traitors* who misuse his privileges to commit malicious activities, *masqueraders* who conduct illegal actions on behalf of legitimate employees of an institute, and *unintentional perpetrators* who unintentionally make mistakes

[13]. Based on the malicious activities conducted by the insiders, the insider threats can also be categorized into three types, *IT sabotage* which indicate to directly uses IT to make harm to an institute, *theft of intellectual property* which indicates to steal information from the institute, *fraud* which indicates unauthorized modification, addition, or deletion of data [14].

It is well accepted that the insiders' behaviors are different from the behaviors of legitimate employees. Hence, analyzing the employees' behaviors via the audit data plays an important role in detecting insiders. In general, there are three types of data sources, host-based, network-based, and context data. The host-based data record activities of employees on their own computers, such as command lines, mouse operations and etc. The network-based data indicate the logs recorded by network equipment such as routers, switches, firewalls and etc. The context data indicate the data from an employee directory or psychological data.

Given different types of data sources, various insider threat detection algorithms have been proposed. For example, some researchers propose to adopt decoy documents or honeypots to lure and identify the insiders [15]. Meanwhile, one common scenario is to consider the insider threat detection as an anomaly detection task and adopt the widely-used anomaly detection approaches, e.g., one-class SVM, to detect the insider threats [6]. Moreover, some approaches treat the employee's actions over a period of time on a computer as a sequence. The sequences that are frequently observed are normal behavior, while the sequences that are seldom observed are abnormal behavior that could be from insiders. Research in [3] adopts Hidden Markov Models (HMMs) to learn the behaviors of normal employees and then predict the probability of a given sequence. An employee activity sequence with low probability predicted by HMMS could indicate an abnormal sequence. Research in [4] evaluates an insider threat detection workflow using supervised and unsupervised learning algorithms, including Self Organizing Maps (SOM), Hidden Markov Models (HMM), and Decision Trees (DT). However, the existing approaches do not model the activity time information. In this work, we aim to capture both the activity time and type information for insider threat detection.

B. Temporal Point Process

A temporal point process (TPP) is a stochastic process composed of a time series of events that occur in continuous time [16]. The temporal point process is widely used for modeling the sequence data with time information, such as health-care analysis, earthquakes and aftershocks modeling and social network analysis [9], [17], [18]. The traditional methods of temporal point processes usually make parametric assumptions about how the observed events are generated, e.g., by Poisson processes or self-exciting point processes. If the data do not follow the prior knowledge, the parametric point processes may have poor performance. To address this problem, researchers propose to learn a general representation of the dynamic data based on neural networks without assum-

ing parametric forms [10], [11]. Those models are trained by maximizing log likelihood. Recently, there are also emerging works incorporating the objective function from generative adversarial network [19], [20] or reinforcement learning [21] to further improve the model performance. However, the current TPP models only focus on one granularity of time. In our scenario, we propose a hierarchical RNN framework to model the multi-scale time information.

III. PRELIMINARY

A. Marked Temporal Point Process

Marked temporal point process is to model the observed random event patterns along time. A typical temporal point process is represented as an event sequence $S = \{e_1, \cdots, e_j, \cdots, e_T\}$. Each event $e_j = (t_j, a_j)$ is associated with an activity type $a_j \in \mathcal{A} = \{1, \cdots, A\}$ and an occurred time $t_j \in [0, T]$. Let $f^*((t_j, a_j)) = f((t_j, a_j)|\mathcal{H}_{t_{j-1}})$ be the conditional density function of the event a_j happening at time t_j given the history events up to time t_{j-1}, where $\mathcal{H}_{t_{j-1}} = \{(t_{j'}, a_{j'})|t_{j'} <= t_{j-1}, a_{j'} \in \mathcal{A}\}$ as the collected historical events before time t_j. Throughout this paper, we use $*$ notation to denote that the function depends on the history. The joint likelihood of the observed sequence S is:

$$f\big(\{(t_j, a_j)\}_{j=1}^{|S|}\big) = \prod_{j=1}^{|S|} f((t_j, a_j)|\mathcal{H}_{t_{j-1}}) = \prod_{j=1}^{|S|} f^*((t_j, a_j)). \tag{1}$$

There are different forms of $f^*((t_j, a_j))$. However, for mathematical simplicity, it usually assumes the times t_j and mark a_j are conditionally independent given the history $\mathcal{H}_{t_{j-1}}$, i.e., $f^*((t_j, a_j)) = f^*(t_j)f^*(a_j)$, where $f^*(a_j)$ models the distribution of event types; $f^*(t_j)$ is the conditional density of the event occurring at time t_j given the timing sequences of past events [10].

A temporal point process can be characterized by the *conditional intensity function*, which indicates the expected instantaneous rate of future events at time t:

$$\lambda^*(t) = \lambda(t|\mathcal{H}_{t_{j-1}}) = \lim_{dt \to 0} \frac{\mathbb{E}[N([t, t + dt])|\mathcal{H}_{t_{j-1}}]}{dt}, \tag{2}$$

where $N([t, t + dt])$ indicates the number of events occurred in a time interval dt. Given the conditional density function f and the corresponding cumulative distribution F at time t, the intensity function can be also defined as:

$$\lambda^*(t) = \frac{f(t|\mathcal{H}_{t_{j-1}})}{S(t|\mathcal{H}_{t_{j-1}})} = \frac{f(t|\mathcal{H}_{t_{j-1}})}{1 - F(t|\mathcal{H}_{t_{j-1}})}, \tag{3}$$

where $S(t|\mathcal{H}_{t_{j-1}}) = exp(-\int_{t_{j-1}}^{t} \lambda^*(\tau)d\tau)$ is the *survival function* that indicates the probability that no new event has ever happened up to time t since $t_j - 1$. Then, the *conditional density function* can be described as:

$$f^*(t) = f(t|\mathcal{H}_{t_{j-1}}) = \lambda^*(t)exp\Big(-\int_{t_{j-1}}^{t} \lambda^*(\tau)d\tau\Big). \tag{4}$$

Particular functional forms of the conditional intensity function $\lambda^*(t)$ for Poisson process, Hawkes process, self-correcting

process, and autoregressive conditional duration process have been widely studied [16]. For example, a Hawkes process captures the self-excitation phenomenon among events [22]. The conditional intensity function of a Hawkes process is defined as:

$$\lambda^*(t) = \lambda_0 + \sum_{t_j \in \mathcal{H}_t} \gamma(t, t_j), \tag{5}$$

where $\lambda_0 > 0$ is the base intensity that indicates the intensity of events triggered by external signals instead of previous events; $\gamma(t, t_j)$ is the triggering kernel which is usually pre-defined as $\gamma(t, t_j) = \exp(\beta(t - t_j))$. The Hawkes process models the self-excitation phenomenon that the arrival of an event increases the conditional intensity of observing events in the near future. Recently, the Hawkes process is widely used to model the information diffusion on online social networks.

However, these different parameterization models make different assumptions about the latent dynamics that is never known in practice. For example, the self-excitation assumption of the Hawkes process may not be held in many scenarios. The model misspecification can seriously degrade the predictive performance.

B. Sequence-to-Sequence Model

In general, a sequence-to-sequence (seq2seq) model is used to convert sequences from one domain to sequences in another domain. The seq2seq consists of two components, one encoder and one decoder. Both encoder and decoder are long short-term memory (LSTM) models and can model the long-term dependency of sequences. The seq2seq model is able to encode a variable-length input to a fixed-length vector and further decode the vector back to a variable-length output. The length of the output sequence could be different from that of the input sequence. The goal of the seq2seq model is to estimate the conditional probability $P(y_1, \cdots, y_{T'}|x_1, \cdots, x_T)$, where $(x_1, \cdots, x_T)$ is an input sequence and $(y_1, \cdots, y_{T'})$ is the corresponding output sequence. The **encoder** encodes the input sequence to a hidden representation with an LSTM model $\mathbf{h}_j^{en} = LSTM^{en}(\mathbf{x}_j, \mathbf{h}_{j-1}^{en})$ where $\mathbf{x}_j$ is the up-to-date input, $\mathbf{h}_{j-1}^{en}$ is the previous hidden state, and $\mathbf{h}_j^{en}$ is the learned current hidden state. The last hidden state $\mathbf{h}_T^{en}$ captures the information of the whole input sequence. The **decoder** computes the conditional probability $P(y_1, \cdots, y_{T'}|x_1, \cdots, x_T)$ by another LSTM model whose initial hidden state is set as $\mathbf{h}_T^{en}$:

$$P(y_1, \cdots, y_{T'}|x_1, \cdots, x_T) = \prod_{j=1}^{T'} P(y_j|\mathbf{h}_T^{en}, y_1, \cdots, y_{j-1}). \tag{6}$$

In seq2seq model, $P(y_j|\mathbf{h}_T^{en}, y_1, \cdots, y_{j-1}) = g(\mathbf{h}_j^{de})$, where $\mathbf{h}_j^{de} = LSTM^{de}(\mathbf{y}_{j-1}, \mathbf{h}_{j-1}^{de})$ is the j-th hidden vector of the decoder; $g(\cdot)$ is usually a softmax function.

IV. INSIDER THREAT DETECTION

A. 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, Web browsing, and FTP. Formally, we model the up-to-date activities of a user as sequence $U = \{S^1, \cdots S^k, \cdots\}$ where $S^k = \{e_1^k, \cdots, e_j^k, \cdots, e_{T_k}^k\}$ indicates his k-th activity session. For example, each session in our scenario is a sequence of activities starting with "LogOn" and ending with "LogOff". $e_j^k = (t_j^k, a_j^k)$ denotes the j-th activity in the user's k-th session and contains activity type a_j^k and occurred time t_j^k. We define $d_j^k = t_j^k - t_{j-1}^k$ as the inter-activity duration between activities a_j^k and a_{j-1}^k, $d^k = t_{T_k}^k - t_1^k$ as the length time of the k-th session, and $\Delta^k = t_1^k - t_{T_{k-1}}^{k-1}$ as the time interval between the $(k-1)$-th and k-th sessions. Note that $t_{T_{k-1}}^{k-1}$ is the occurred time of the last activity in the $(k-1)$-th session.

The goal of learning in our threat detection is to predict whether a new session $S^k = \{e_1^k, \cdots, e_j^k, \cdots, e_{T_k}^k\}$ is normal or fraudulent. To address the challenge that there are often no or very few records of known insider attacks for training our model, we propose a generative model that models normal user behaviors from a training dataset consisting of only sequences of normal users. The learned model is then used to calculate the fraudulent score of the new session S^k. We quantify the fraudulence of S^k from two perspectives, activity information (including both type and time) within sessions, and session time information (i.e., when a session starts and ends). For example, a user who foresees his potential layoff may have activities of uploading documents to Dropbox and visiting job-searching websites although he may try to hide these abnormal activities in multiple sessions; he may have "LogOn" and "LogOff" times different from his normal sessions as he may become less punctual or may have more sessions during weekends or nights, resulting different session durations and intervals between sessions. Moreover, when a user's account is compromised, activity and session information from the attacker will also be different even if the attacker tries to mimic the normal user's behaviors.

We develop a unified hierarchical model capable of capturing a general nonlinear dependency over the history of all activities. Our detection model does not rely on any predefined signatures and instead use deep learning models to capture user behaviors reflected in raw data. Specifically, our hierarchical model learns the user behaviors in two time scales, intra-session level and inter-session level. For the intra-session level, we adopt the seq2seq model to predict $\hat{S}^k$ based on the previous S^{k-1} and use the marked temporal point process model to capture the dynamic difference of activities. Note that the number of activities of the predicted session $\hat{S}^k$ could be different from that of the previous S^{k-1} as well as the true S^k. For the inter-session level, we aim to model the session interval $\Delta^k = t_1^k - t_{T_{k-1}}^{k-1}$ and the session duration $d^k = t_{T_k}^k - t_1^k$ of the k-th session.

The whole framework of predicting future events with two time scales is shown in Figure 1. We do not assume any specific parametric form of the conditional intensity function. Instead, we follow [10] to seek to learn a general representation to approximate the unknown dependency structure
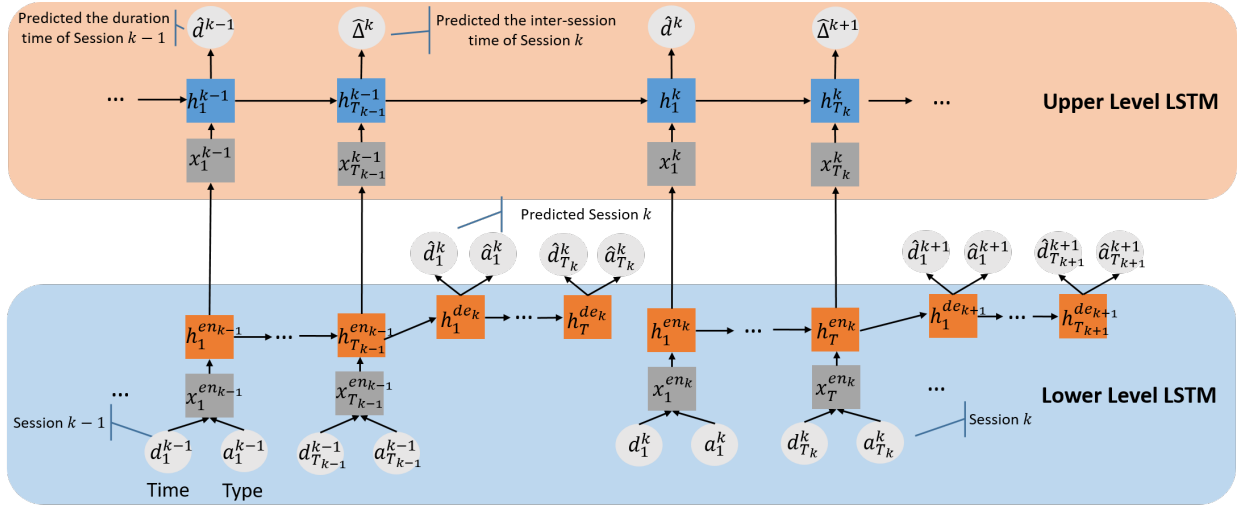
Figure 1: The framework for sequence generation with two time scales. The lower-level LSTM captures event patterns with the time and mark pairs in a session. The upper-level LSTM aims to predict the duration of sessions and inter-sessions.

over the history. We also emphasize that the neural temporal point processes of two levels are connected in our framework. The upper-level LSTM takes the first and last hidden states from the encoder of the lower-level LSTM as inputs to predict the interval of two sessions and the session duration. This connection guarantees the upper-level LSTM incorporates activity type information in its modeling. For insider threat detection, since our model is trained by benign sessions, the predicted session $\hat{S}^k$ would be close to the observed S^k when S^k is normal, and different from S^k when S^k is abnormal. In Session IV-D, we will present details about how to derive fraudulent score by comparing $(\hat{S}^k, \hat{d}^k, \hat{\Delta}^k)$ with (S^k, d^k, Δ^k), where $\hat{\bullet}$ indicates the predicted value.

B. Intra-Session Insider Threat Detection

In this work, we propose to use the seq2seq model to estimate the joint likelihood of k-th session given the $(k-1)$-th session. In particular, the encoder of the seq2seq model is to encode the activity time and type information at $(k-1)$-th session to a hidden representation. The decoder is to model the activity time interval d_{j+1}^k and type a_{j+1}^k information at k-th session given the history.

Encoder: To map the $(k-1)$-th session to a hidden representation, the encoder first maps each activity occurring at time t_j^{k-1} with type a_j^{k-1} to an embedding vector $\mathbf{x}_j^{en_{k-1}}$:

$$\mathbf{x}_j^{en_{k-1}} = \mathbf{w}^t d_j^{k-1} + \mathbf{W}^{em} \mathbf{a}_j^{k-1}, \qquad (7)$$

where d_j^{k-1} is the inter-activity duration between a_j^{k-1} and a_{j-1}^{k-1}; $\mathbf{w}^t$ is a time-mapping parameter; $\mathbf{W}^{em}$ is an activity embedding matrix; $\mathbf{a}_j^{k-1}$ is a one-hot vector of the activity type a_j^{k-1}. Then, by taking the entire sequence of $(k-1)$-th session as inputs to the encoder LSTM, the encoder projects the $(k-1)$-th session to a hidden representation $\mathbf{h}_{T_{k-1}}^{en_{k-1}}$.

Decoder: The decoder is trained to predict the pairs of activity type and time at the k-th session given the information

of $(k-1)$-th session. To predict the activity type information, given the hidden state of the decoder $\mathbf{h}_j^{de_k}$, the probability of the next activity having type value a can be derived by a softmax function:

$$P(a_{j+1}^k = a|\mathbf{h}_j^{de_k}) = \frac{exp(\mathbf{w}_a^s \mathbf{h}_j^{de_k})}{\sum_{a'=1}^{A} exp(\mathbf{w}_{a'}^s \mathbf{h}_j^{de_k})}, \qquad (8)$$

where $\mathbf{w}_a^s$ is the a-th row of the weight matrix $\mathbf{W}^s$ in the softmax function.

To predict the activity time information, we adopt the conditional density function defined in Equation 4. First, inspired by [10], we derive the LSTM-based conditional intensity function $\lambda^*(t)$ as:

$$\lambda^*(t) = exp(\mathbf{v}\mathbf{h}_j^{de_k} + u^t(t - t_j^k) + b), \qquad (9)$$

where the exponential function is deployed to ensure the intensity function is always positive; $\mathbf{v}$ is a weight vector; u^t and b are scalars. Then, we can derive the conditional density function given the history until time t_j^k:

$$\begin{aligned} f^*(t) &= \lambda^*(t)(\int_{t_j^k}^{t} \lambda^*(\tau)d\tau) \\ &= exp\big(\mathbf{v}^t\mathbf{h}_j^{de_k} + u^t(t - t_j^k) + b^t + \frac{1}{u}exp(\mathbf{v}^t\mathbf{h}_j^{de_k} \\ &\quad + b^t) - \frac{1}{u}exp(\mathbf{v}^t\mathbf{h}_j^{de_k} + u^t(t - t_j^k) + b^t)\big). \end{aligned} \qquad (10)$$

Hence, given the observed activity time information, we can calculate the conditional density function of the time interval between two consecutive activities $d_{j+1}^k = t_{j+1}^k - t_j^k$ at k-th session:

$$f^*(d_{j+1}^k) = f(d_{j+1}^k|\mathbf{h}_j^{de_k}). \qquad (11)$$

Since the lower-level LSTM is to model the time interval d_{j+1}^k and type a_{j+1}^k information, given a collection of activity sessions from benign employees, we combine the likelihood

functions of the event type (Equation 8) and time (Equation 11) to have the negative joint log-likelihood of the observation sessions:

$$\mathcal{L}_a = -\sum_{k=1}^{M}\sum_{j=1}^{T_k} \Big(\log P(a_{j+1}^k|\mathbf{h}_j^{de_k}) + \log f^*(d_{j+1}^k) \Big), \quad (12)$$

where M is the total number of sessions in the training dataset; T_k is the number of activities in a session. The lower-level LSTM along with the decoder LSTM is trained by minimizing the negative log-likelihood shown in Equation 12.

When the model is deployed for detection, to obtain the predicted activity type $\hat{a}_{j+1}^k$, we simply choose the type with the largest probability $P(a|\mathbf{h}_j^{de_k})$ (calculated by Equation 8):

$$\hat{a}_{j+1}^k = \operatorname*{argmax}_{a\in\mathcal{A}} P(a|\mathbf{h}_j^{de_k}). \quad (13)$$

We further calculate the expected inter-activity duration between $(j+1)$-th and j-th activities $\hat{d}_{j+1}^k = E(t_j^k)$:

$$\hat{d}_{j+1}^k = \int_{t_j^k}^{\infty} t f^*(t) dt. \quad (14)$$

The difference between $\hat{d}_{j+1}^k$ and the observed d_{j+1}^k will be used to calculate the fraudulent score in terms of the timing information of intra-session activities.

C. Inter-Session Insider Threat Detection

The inter-session duration is crucial for insider threat detection. To capture such information, we further incorporate an upper-level LSTM into the framework, which focuses on modeling the inter-session behaviors of employees. Specifically, the upper-level LSTM is trained to predict the inter-session duration between k-th and $(k-1)$-th sessions ($\Delta^k = t_1^k - t_{T_{k-1}}^{k-1}$) and the k-th session duration ($d^k = t_{T_k}^k - t_1^k$).

To predict the inter-session duration Δ^k, the input of the upper-level LSTM is from the last hidden state $\mathbf{h}_{T_{k-1}}^{en_{k-1}}$ of $(k-1)$-th session from the lower-level LSTM as shown in Equation 15, while to predict the k-th session duration d^k, the input of the upper-level LSTM is from the first hidden state $\mathbf{h}_1^{en_k}$ of k-th session as shown in Equation 16.

$$\mathbf{x}_{T_{k-1}}^{k-1} = \mathbf{U}\mathbf{h}_{T_{k-1}}^{en_{k-1}}, \quad (15)$$
$$\mathbf{x}_1^k = \mathbf{U}\mathbf{h}_1^{en_k}, \quad (16)$$

where $\mathbf{U}$ is an input weight matrix for the upper-level LSTM.

Then, we can get the hidden states ($\mathbf{h}_{T_{k-1}}^{k-1}$ and $\mathbf{h}_1^k$) of the upper-level sequence based on an LSTM model. Finally, the conditional density functions of the inter-session duration Δ^k and session duration d^k are:

$$f_s^*(\Delta^k) = f(\Delta^k|\mathbf{h}_{T_{k-1}}^{k-1}), \quad (17)$$
$$f_s^*(d^k) = f(d^k|\mathbf{h}_1^k), \quad (18)$$

where $f_s^*(\Delta^k)$ and $f_s^*(d^k)$ can be calculated based on Equation 10.

To train the upper-level LSTM, the negative log-likelihood of inter-session sequences can be defined as:

$$\mathcal{L}_s = -\sum_{m=1}^{M'}\sum_{k=1}^{K_m} \big(\log f_s^*(\Delta_m^k) + \log f_s^*(d_m^k) \big), \quad (19)$$

where M' is the total number of inter-session level sequences in the training dataset; K indicates the number of sessions in an inter-session level sequence. In our experiments, we use the upper-level LSTM to model the employee sessions in a week. Then, M' indicates the total number of weeks in the training dataset, and K is the number of sessions in a week. The upper-level LSTM is trained by minimizing the negative log-likelihood shown in Equation 19. After training, the upper-level LSTM can capture the patterns of the inter-session duration and session duration.

When the model is deployed for detection, we calculate the predicted inter-session duration between k-th and $(k-1)$-th sessions $\hat{\Delta}^k$ and the k-th session duration $\hat{d}^k$, shown in Equations 20.

$$\hat{\Delta}^k = \int_{t_T^{k-1}}^{\infty} t f_s^*(t), \quad \hat{d}^k = \int_{t_1^k}^{\infty} t f_s^*(t) dt. \quad (20)$$

The difference between $\hat{\Delta}^k$ ($\hat{d}^k$) and the observed Δ^k (d^k) will be used to calculate the fraudulent score in terms of the session timing information.

D. Fraudulent Score

After obtaining the predicted session, we compare the generated times and types in a session with the observed session, respectively. For the activity types, we adopt the Bilingual Evaluation Understudy (BLEU) [23] score to evaluate the difference between the observed session and generated session. The BLEU metric was originally used for evaluating the similarity between a generated text and a reference text, with values closer to 1 representing more similar texts. BLEU is derived by counting matching n-grams in the generated text to n-grams in the reference text and insensitive to the word order. Hence, BLEU is suitable for evaluating the generated sequences and the observed sequences. We define the fraudulent score in terms of intra-session activity type as:

$$score_a = 1 - BLEU(S_a^k, \hat{S}_a^k), \quad (21)$$

where S_a^k indicates the observed activity types in k-th session while $\hat{S}_a^k$ indicates the predicted session. If $score_a$ is high, it means the observed session is a potentially malicious session in terms of session activity types.

For the activity time, as shown in Equation 22, we define the fraudulent score in terms of intra-session activity time by computing the mean absolute error (MAE) of the predicted time of each activity with the observed occurring time:

$$score_t = \frac{1}{|S|}\sum_{j=1}^{|S|} |d_j^k - \hat{d}_j^k|. \quad (22)$$

Since the upper-level LSTM takes each session's first and last hidden states as inputs to predict the time lengths of

sessions and inter-sessions, we can further derive the time scores by comparing the predicted time lengths with the observed ones. We define the fraudulent score in terms of inter-session duration as:

$$score_\Delta = |\hat{\Delta}^k - \Delta^k|. \quad (23)$$

Similarly, we define the fraudulent score in terms of session duration as:

$$score_d = |\hat{d}^k - d^k|. \quad (24)$$

Note that although $\hat{d}^k$ can be derived based on all the predicted activity time from lower-level LSTM, the error usually is high due to the accumulated error over the whole sequence. Hence, we use the upper-level LSTM to get the session time length. Finally, by combining Equations 21, 22, 23 and 24, we define the total fraudulent score (FS) of a session as:

$$FS = \alpha_1 score_a + \alpha_2 score_t + \alpha_3 score_d + \alpha_4 score_\Delta, \quad (25)$$

where $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are hyper-parameters, which can be set based on the performance of insider threat detection via using each score alone.

V. EXPERIMENTS

A. Experiment Setup

Dataset. We adopt the CERT Insider Threat Dataset [24], which is the only comprehensive dataset publicly available for evaluating the insider threat detection. This dataset consists of five log files that record the computer-based activities for all employees, including **logon.csv** that records the logon and logoff operations of all employees, **email.csv** that records all the email operations (send or receive), **http.csv** that records all the web browsing (visit, download, or upload) operations, **file.csv** that records activities (open, write, copy or delete) involving a removable media device, and **decive.csv** that records the usage of a thumb drive (connect or disconnect). Table II shows the major activity types recorded in each file. The CERT dataset also has the ground truth that indicates the malicious activities committed by insiders. We use the latest version (r6.2) of CERT dataset that contains 3995 benign employees and 5 insiders.

We join all the log files, separate them by each employee, and then sort the activities of each employee based on the recorded timestamps. We randomly select 2000 benign employees as the training dataset and another 500 employees as the testing dataset. The test dataset includes all sessions from five insiders. The statistics of the training and testing datasets is shown in Table I. Based on the activities recorded in the log files, we extract 19 activity types shown in Table II. The activity types are designed to indicate the malicious activities.
Baselines. We compare our model with two one-class classifiers: 1) One-class SVM (**OCSVM**) [25] adopts support vector machine to learn a decision hypersphere around the positive data, and considers samples located outside this hypersphere as anomalies; 2) Isolation Forest (**iForest**) [26] detects the anomalies with short average path lengths on a set of trees. For both baselines, we consider each activity type as an input

Table I: Statistics of Training and Testing Datasets

	Training Dataset	Testing Dataset
# of Employees	2000	500
# of Sessions	1039805	142600
# of Insiders	0	5
# of Malicious Sessions	0	68

Table II: Operation Types Recorded in Log Files

Files	Operation Types
logon.csv	Weekday Logon (employee logs on a computer on a weekday at work hours)
	Afterhour Weekday Logon (employee logs on a computer on a weekday after work hours)
	Weekend Logon (employees logs on at weekends)
	Logoff (employee logs off a computer)
email.csv	Send Internal Email (employee sends an internal email)
	Send External Email (employee sends an external email)
	View Internal Email (employee views an internal email)
	View external Email (employee views an external email)
http.csv	WWW Visit (employee visits a website)
	WWW Download (employee downloads files from a website)
	WWW Upload (employee uploads files to a website)
device.csv	Weekday Device Connect (employee connects a device on a weekday at work hours)
	Afterhour Weekday Device Connect (employee connects a device on a weekday after hours)
	Weekend Device Connect (employee connects a device at weekends)
	Disconnect Device (employee disconnects a device)
file.csv	Open doc/jpg/txt/zip File (employee opens a doc/jpg/txt/zip file)
	Copy doc/jpg/txt/zip File (employee copies a doc/jpg/txt/zip file)
	Write doc/jpg/txt/zip File (employee writes a doc/jpg/txt/zip file)
	Delete doc/jpg/txt/zip File (employee deletes a doc/jpg/txt/zip file)

feature and the feature value is the number of activities of the corresponding type in a session. In this paper, we do not compare with other RNN based insider threat detection methods (e.g., [7]) as these methods were designed to detect the insiders or predict the days that contain insider threat activities.
Hyperparameters. We map the extracted activity types to the type embeddings. The dimension of the type embeddings is 50. The dimension of the LSTM models is 100. We adopt Adam [27] as the stochastic optimization method to update the parameters of the framework. When training the upper-level LSTM by Equation 19, we fix the parameters in the lower-level LSTM and only update the parameters in the upper-level LSTM.

B. Experiment Results

We aim to detect all the 68 malicious sessions from the totally 142,600 sessions in the testing set. Figure 2 shows the receiver operating characteristic (ROC) curves of our model for insider threat detection by leveraging various fraudulent scores. By using each fraudulent score separately, we can
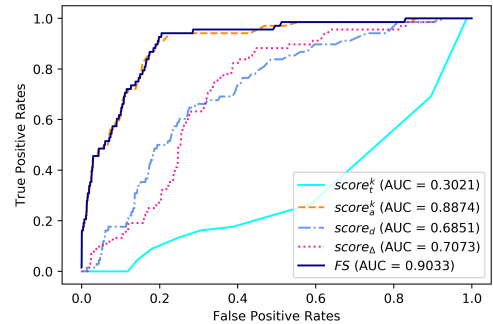

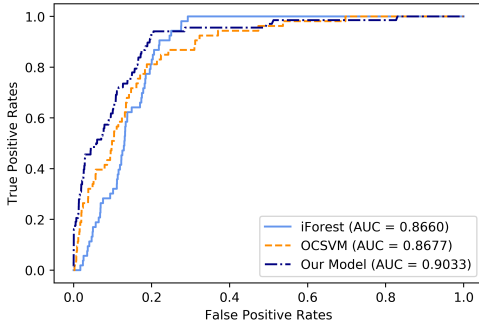Figure 2: ROC curve of malicious session detection using various fraudulent scores

Figure 3: ROC curve of malicious session detection using various approaches

Figure 4: ROC curve of vandalism session detection using various fraudulent scores

notice that the $score_a$ derived from intra-session activity types achieves the highest area under cure (AUC) score, which indicates the activity types of malicious sessions are different from the normal sessions. Meanwhile, the session duration time and inter-session duration time also make positive contributions to the malicious session detection. The $score_d$ derived from the session duration time and $score_\Delta$ derived from the inter-session duration time achieve good performance with AUC=0.6851 and 0.7073, respectively, which indicates the duration of malicious sessions and inter-sessions are usually different from those of normal sessions. We also notice that the $score_t$ based on the inter-activity activity time information does not help much on insider threat detection. The AUC derived from $score_t$ is 0.3021. After examining the data, we find that there is no much difference in terms of inter-activity time information between malicious sessions and normal sessions. Since adopting $score_t$ does not achieve reasonable performance in the CERT Insider Threat dataset, we set $\alpha_2 = 0$ when deriving the total insider threat detection FS. As a result, our detection model using the total insider threat detection FS, which combines all the intra- and inter-session information, achieves the best performance with the AUC=0.9033 when the hyper-parameters in Equation 25 are $\alpha_1 = 1, \alpha_2 = 0, \alpha_3 = 1, \alpha_4 = 1$.

Figure 3 further shows the ROC curves of our model and two baselines. We can observe that our model achieves better performance than baselines in terms of AUC score. Especially, we can notice that when our model only adopts the activity type information ($score_a$) for malicious session detection, our model is slightly better than baselines in terms of AUC. With further combining the activity time and type information, our model significantly outperforms the baselines with the AUC=0.9033.

C. Vandal Detection

Dataset. Due to the limitation of the CERT dataset where the inter-activity duration times are randomly generated, the inter-activity time in the intra-session level does not make contributions to the insider threat detection. To further show the advantage of incorporating activity time information, we apply our model for detecting vandals on Wikipedia. Vandals
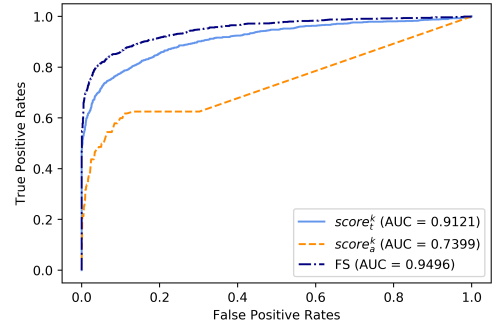
can be considered as insiders in the community of Wikipedia contributors. The study has shown that the behaviors of vandals and benign users are different in terms of edit time, e.g., vandals make faster edits than benign users [28]. Hence, we expect that using the inter-activity time information can boost the performance of vandal detection.

We conduct our evaluation on the UMDWikipedia dataset [28]. This dataset contains information of around 770K edits from Jan 2013 to July 2014 (19 months) with 17105 vandals and 17105 benign users. Each user edits a sequence of Wikipedia pages. We adopt half of the benign users for training and the other half of the benign users and all the vandals for testing. Since user activities on Wikipedia do not have explicit indicators, such as LogON or LogOff, to split the user activity sequence into sessions, we consider user activities in a day as a user session. As a result, the session duration is always 24hrs, and the inter-session duration is 0. Therefore, in this experiment, we focus on vandalism session detection with only using information from the intra-session level and adopt the lower-level LSTM shown in Figure 1 accordingly. Note that we filter out all the sessions with number of activities less than 5. The seq2seq model takes a feature vector as an input and predicts the next edit time and type. In this experiment, we consider the activity type as whether the current edit will be reverted or not. The feature vector of the user's t-th edit is composed by: (1) whether or not the user edited on a meta-page; (2) whether or not the user consecutively edited the pages less than 1 minute, 3 minutes, or 5 minutes; (3) whether or not the user's current edit page had been edited before.

Experiment Results. From Figure 4, we can observe that only using the inter-activity time information can achieve surprisingly good performance on vandalism session detection with AUC=0.9121, which indicates the inter-activity time information is crucial for vandalism session detection. Meanwhile, adopting the activity type information can also achieve the vandalism session detect with AUC=0.7399. Hence, using inter-activity time information achieves better performance than using the activity type information in terms of AUC. It also means vandals have significantly different patterns in activity time compared with benign users. Finally, with combining the activity type and time information, our model
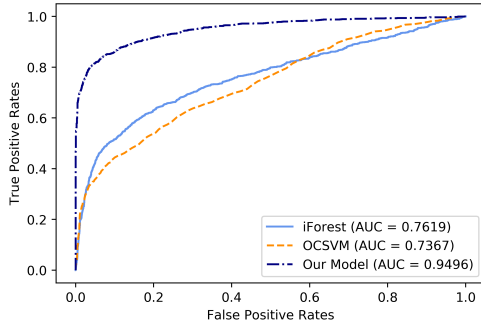
Figure 5: ROC curve of vandalism session detection using various approaches

can achieve even better performance with AUC=0.9496.

We further compare our model with two baselines, i.e., One-class SVM and Isolation Forest. For the baselines, we consider the same features as the seq2seq model and further combine activity types. The value of each feature is the mean value of the corresponding feature in a day. Figure 5 indicates that our model significantly outperforms baselines in terms of AUC on the vandalism session detection task. Similar to the results on the CERT dataset, when our model only adopts the activity type information ($score_a$ shown in Figure 4), the model achieves similar performance as baselines. With considering the activity time information, the performance of our model is improved by a large margin.

VI. Conclusion

In this paper, we have proposed a two-level neural temporal point process model for insider threat detection. In the lower-level, we combined the seq2seq model with marked temporal point processes to dynamically capture the intra-session information in terms of activity times and types. The upper-level LSTM takes the first and last hidden states from the encoder of the lower-level LSTM as inputs to predict the interval of two sessions and the session duration based on activity history. Experimental results on an insider threat detection dataset and a Wikipedia vandal detection dataset demonstrated the effectiveness of our model.

Acknowledgment

References

[1] CSO, U. S. S. C. D. of SRI-CMU, and ForcePoint, "2018 u.s. state of cybercrime," Tech. Rep., 2018.
[2] H. Eldardiry, E. Bart, J. Liu, J. Hanley, B. Price, and O. Brdiczka, "Multi-domain information fusion for insider threat detection," in *2013 IEEE Security and Privacy Workshops*. IEEE, 2013, pp. 45–51.
[3] T. Rashid, I. Agrafiotis, and J. R. Nurse, "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*, 2016.
[4] D. C. Le and A. N. Zincir-Heywood, "Evaluating insider threat detection workflow using supervised and unsupervised learning," in *2018 IEEE Security and Privacy Workshops*, 2018.
[5] M. B. Salem, S. Hershkop, and S. J. Stolfo, "A survey of insider attack detection research," in *Insider Attack and Cyber Security: Beyond the Hacker*, 2008.
[6] A. Sanzgiri and D. Dasgupta, "Classification of insider threat detection techniques," in *Proceedings of the 11th Annual Cyber and Information Security Research Conference*, 2016.
[7] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," in *AI for Cyber Security Workshop*, 2017.
[8] T. E. Senator, H. G. Goldberg, A. Memory, and et al., "Detecting insider threats in a real corporate database of computer usage activity," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013.
[9] A. Reinhart, "A review of self-exciting spatio-temporal point processes and their applications," *arXiv:1708.02647 [stat]*, 2017.
[10] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song, "Recurrent marked temporal point processes: Embedding event history to vector," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.
[11] H. Mei and J. Eisner, "The neural hawkes process: A neurally self-modulating multivariate point process," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.
[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[13] L. Liu, O. D. Vel, Q. Han, J. Zhang, and Y. Xiang, "Detecting and preventing cyber insider threats: A survey," *IEEE Communications Surveys Tutorials*, vol. 20, no. 2, pp. 1397–1417, 2018.
[14] I. Homoliak, F. Toffalini, J. Guarnizo, Y. Elovici, and M. Ochoa, "Insight into insiders and it: A survey of insider threat taxonomies, analysis, modeling, and countermeasures," *ACM Comput. Surv.*, vol. 52, no. 2, pp. 30:1–30:40, Apr. 2019.
[15] L. Spitzner, "Honeypots: catching the insider threat," in *Proceedings of the 19th Annual Computer Security Applications Conference*, 2003.
[16] J. G. Rasmussen, "Lecture notes: Temporal point processes and the conditional intensity function," *arXiv:1806.00221 [stat]*, 2018.
[17] Q. Zhao, M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec, "Seismic: A self-exciting point process model for predicting tweet popularity," in *KDD*, 2015.
[18] M. Farajtabar, "Point process modeling and optimization of social networks," Ph.D. dissertation, 2018.
[19] S. Xiao, M. Farajtabar, X. Ye, J. Yan, L. Song, and H. Zha, "Wasserstein learning of deep generative point process models," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.
[20] H. Zha, J. Yan, X. Liu, L. Shi, and C. Li, "Improving maximum likelihood estimation of temporal point process via discriminative and adversarial learning," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 2018.
[21] S. Li, S. Xiao, S. Zhu, N. Du, Y. Xie, and L. Song, "Learning temporal point processes via reinforcement learning," in *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, 2018.
[22] A. G. Hawkes, "Spectra of some self-exciting and mutually exciting point processes," *Biometrika*, vol. 58, no. 1, pp. 83–90, 1971.
[23] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 2002.
[24] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *IEEE Security and Privacy Workshops*, 2013.
[25] D. M. J. Tax and R. P. W. Duin, "Support vector data description," *Machine Learning*, vol. 54, no. 1, pp. 45–66, 2004.
[26] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, 2008.
[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, 2015.
[28] S. Kumar, F. Spezzano, and V. Subrahmanian, "Vews: A wikipedia vandal early warning system," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015.