Using Dirichlet Marked Hawkes Processes for Insider Threat Detection

PANPAN ZHENG, Amazon
SHUHAN YUAN, Utah State University
XINTAO WU, University of Arkansas

Malicious insiders cause significant loss to organizations. Due to an extremely small number of malicious activities from insiders, insider threat is hard to detect. In this article, we present a **Dirichlet Marked Hawkes Process (DMHP)** to detect malicious activities from insiders in real-time. DMHP combines the Dirichlet process and marked Hawkes processes to model the sequence of user activities. The Dirichlet process is capable of detecting unbounded user modes (patterns) of infinite user activities, while, for each detected user mode, one set of marked Hawkes processes is adopted to model user activities from time and activity type (e.g., WWW visit or send email) information so that different user modes are modeled by different sets of marked Hawkes processes. To achieve real-time malicious insider activity detection, the likelihood of the most recent activity calculated by DMHP is adopted as a score to measure the maliciousness of the activity. Since the majority of user activities are benign, those activities with low likelihoods are labeled as malicious activities. Experimental results on two datasets show the effectiveness of DMHP.

CCS Concepts: • **Security and privacy → Intrusion detection systems**; • **Computing methodologies → Machine learning approaches;**

Additional Key Words and Phrases: Insider threat detection, Dirichlet process, Hawkes process

ACM Reference format:
Panpan Zheng, Shuhan Yuan, and Xintao Wu. 2021. Using Dirichlet Marked Hawkes Processes for Insider Threat Detection. *Digit. Threat.: Res. Pract.* 3, 1, Article 5 (October 2021), 19 pages.
https://doi.org/10.1145/3457908

1 INTRODUCTION

Insider threat, which is one of the most challenging cybersecurity issues, indicates malicious threat from people within an organization. The insider threat is comprised of both malicious and unintentional threat, while, in this article, we focus on the former, which may involve intentional fraud, the theft of confidential or commercially valuable information, or the sabotage of computer systems. The attacks from malicious insiders are more difficult to detect compared to those from external attackers because insiders are usually authorized users who can legitimately access the IT system inside the organization. The 2018 U.S. State of Cybercrime Survey indicates

This work was conducted when Panpan Zheng was a Ph.D. student at the University of Arkansas.
This work was supported in part by NSF grants 1564250 and 2103829.
Authors' addresses: P. Zheng, Buyer Risk Prevention Machine Learning Team, Amazon, 501 Fairview Ave N, Seattle, WA 98109; email: panpz@amazon.com; S. Yuan, Computer Science, Utah State University, 4205 Old Main Hill, Logan, UT 84322; X. Wu, Computer Science & Computer Engineering Department, University of Arkansas, Fayetteville, AR 72701; email: xintaowu@uark.edu.

2576-5337/2021/10-ART5 $15.00
https://doi.org/10.1145/3457908

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 [4].

Various insider threat detection approaches have been proposed [6, 16, 23, 27–29, 34, 37]. However, the existing works usually focus on detecting malicious sessions, e.g., a sequence of activities in a day, and cannot achieve the dynamic malicious activity detection in a real-time manner. In this article, we aim to develop a detection model that can dynamically detect malicious activities. To this end, besides adopting the activity type (e.g., WWW visit, send email) information, we further consider the crucial activity time information since the time an activity occurred is also a strong indicator about whether the activity is malicious. A potential example of an insider threat attack is an employee who normally does not work after-hours being found connecting a USB flash drive to an office computer system after midnight.

The intuitive idea of detecting malicious insider activities is to model the occurrence likelihoods of activities in terms of activity type and time. Since the majority of activities are benign even for the insiders, the likelihoods of benign activities should be high. Hence, if an activity is occurred with a low likelihood in terms of activity type and time, it could be a malicious activity. In this work, we adopt the **Marked Temporal Point Process (MTPP)**, which is a general mathematical framework, to model the user activities over time. Specifically, we adopt the Hawkes process to model the temporal information of user activities, while the activity types are captured by a mark model. Hawkes process is a self-exciting point process, in which the probability of observing a new activity increases due to previous activities. This phenomenon is commonly observed in our daily activities. For example, when a user starts editing a document, the majority of following activities will be about editing a document.

However, the activity sequence from a user consists of various temporal patterns that are driven by his working patterns. For example, if the user is in a work mode, his activities are dense in a short time window, and the activity types mainly consist of file editing. If the user is in the leisure mode, his activities would be sparse in a time window, and the activity types may be more about Web surfing. We call the different working patterns *user modes*. Hence, only using one marked Hawkes process model is unable to capture the dynamic user activities under various user modes. Meanwhile, due to the potential infinite number of user activities, the number of user modes is also unknown. Hence, it is hard to use a fixed number of marked Hawkes processes to model the complicated user activities. The effectiveness of modeling the sequence of activities and the following detecting malicious insider activities depends on how to capture the unbounded user modes from the observed sequence.

In this article, we develop the **Dirichlet Marked Hawkes Process (DMHP)** to detect malicious insider threats. DMHP is able to cluster user activities into user modes, in which all activities have a similar temporal dynamic pattern in terms of activity type and time. Then, DMHP models the user activities in a specific mode by a marked Hawkes process. For each particular activity in the sequence, we then derive an occurrence likelihood based on DMHP and use it as an evaluation score for predicting whether or not the activity is malicious. Compared with the classical marked temporal point process model, DMHP is able to identify user modes automatically and model these modes by different temporal point processes. Meanwhile, DMHP does not need to preset the number of user modes in the sequence of user activities. Theoretically, the number of user modes generated by DMHP can be infinite, which matches the scenario that user activity stream is also infinite.

Our work makes the following contributions: (1) we develop a dynamic and unsupervised insider threat activity detection model that considers both activity type and time information; (2) to this end, we propose a Dirichlet Marked Hawkes Process that can effectively capture the sequence of user activities with unbounded number of modes; (3) we conduct experiments on two datasets, the CERT Insider Threat Dataset [9] and the UMDWikipedia dataset [15], and evaluation results demonstrate the effectiveness of our model for malicious activity detection. We emphasize that insider attacks are subtle and dynamic and the number of malicious insider activities is only a very small fraction of all activities from insiders (e.g., 0.01% to 0.3% in CERT data). Detection models based on supervised learning usually do not work here due to lack of labeled insider activity records. On the other hand, our DMHP detection model, which is based on unsupervised learning, does not require any labeled attack records, and can score each coming activity in a real-time manner.

This article is organized as follows: Section 2 reviews the related work in the field of insider threat detection. Section 3 gives a brief introduction of the Dirichlet process and temporal point process. Section 4 introduces our DMHP model for the malicious activity detection by deriving a score based on the occurrence likelihood. Section 5 describes the experiments of applying DMHP on two datasets for insider threat detection. Section 6 concludes the article and points out the potential future directions.

2 RELATED WORK

Increasingly sophisticated cyber attacks are one form of insider attacks by people that have both the capability, intent, and domain knowledge about the system (even including deployed attack detection systems). An insider attack often involves multiple phases, and malicious activities are hidden among a huge number of normal activities. The subtle, dynamic, and multi-phase nature of insider attacks makes detection extremely difficult. In literature, based on the intention of attacks, there are three types of insiders, i.e., *traitors* who misuse their privileges to commit malicious activities, *masqueraders* who conduct illegal actions on behalf of legitimate employees of an organization, and *unintentional perpetrators* who unintentionally make mistakes [20]. Based on the malicious activities conducted by insiders, insider threats can also be categorized into three types, *IT sabotage* which indicates to directly use IT to make harm to organizations, *theft of intellectual property* which indicates to steal information from organizations, *fraud* which indicates unauthorized modification, addition, or deletion of data [11].

It is well accepted that insiders' behaviors are different from behaviors of legitimate employees. Hence, analyzing employees' behaviors via audit data plays an important role in detecting insiders [36]. 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 the like. The network-based data indicate the logs recorded by network equipment such as routers, switches, firewalls and so on. The context data indicate the data from an employee directory or psychological data.

Analysts apply various insider threat detection algorithms given different types of data sources. For example, some researchers propose to adopt decoy documents or "honeypots" to lure and identify insiders [31]. 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 [28]. 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 [23] 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 [16] evaluates an insider threat detection workflow using supervised and unsupervised learning algorithms, including self-organizing maps, HMMs, and decision trees. Research in [34] adopts a recurrent neural network to detect malicious activities based on manually designed input features. Research in [18] proposes a heterogeneous graph embedding approach considering each log entry as a node for insider threat detection. After learning the node embeddings, a clustering algorithm is adopted to detect malicious clusters containing malicious activities. Although the proposed graph embedding approach is able to detect malicious activities, it cannot detect them in a real-time manner since the heterogeneous graph is constructed based on activities in several days.

Anomaly detection in streaming data has also received attention recently. Research in [12] uses a small set of dynamically maintained orthogonal basis vectors, which represent all the prior non-anomalous data points, to reconstruct an upcoming point, and then marks as anomalous if it cannot be reconstructed well by these basis vectors. However, most existing studies do not provide a practically effective streaming-based model to describe the physical time information and the associated text information. In this work, we model both physical activity time and categorical activity type to achieve insider threat detection at the activity level in a streaming manner. Moreover, to achieve high detection accuracy, we introduce the use of Dirichlet process to capture hidden mode information.

3 PRELIMINARY

3.1 Dirichlet Process

Identifying user modes from an activity sequence can be viewed as a task for clustering sequential data. The widely used models for clustering data from the topic modeling literature are the **Latent Dirichlet Allocation (LDA)** [3], where the number of topics is fixed, and its improved model, **Hierarchical Dirichlet Process (HDP)** [33] with an unbounded number of topics.

The **Dirichlet Process (DP)** is a Bayesian nonparametric model to cluster data without specifying the number of clusters in advance [1, 2, 8, 32]. The Dirichlet process is parameterized by a concentration parameter $\eta > 0$ and a base distribution G_0 over a space Θ. It indicates that a random distribution G drawn from DP is a distribution over Θ, denoted as $G \sim DP(\eta, G_0)$. For example, if G is a distribution on reals, G_0 must be a distribution on reals too. The expectation of the distribution G is the base distribution G_0. The concentration parameter controls the variance of G, where the higher values of η lead to tight distributions around G_0. DP is widely used for clustering with the unknown number of clusters.

Chinese Restaurant Process (CRP) is one kind of representations for Dirichlet process. CRP assumes a restaurant with an infinite number of tables, each of which can seat an infinite number of customers. Within the context of clustering, each table indicates a cluster while each customer is a data point. The simulation process of CRP is as follows:

(1) The first customer always sits at the first table.
(2) Customer n $(n > 1)$ is distributed to:
  (a) a new table with probability $\frac{\eta}{\eta+n-1}$.
  (b) an existing table z with probability $\frac{n_z}{\eta+n-1}$ where n_z is the number of customers at table z.

Let $\beta_1, \ldots, \beta_n$ be a sequence sampled from CRP. The conditional distribution of β_n can be written as:

$$\beta_n | \beta_{1:n-1} \sim \frac{1}{\eta + n - 1} \left(\eta G_0 + \sum_z n_z \delta_{\beta_z} \right), \tag{1}$$

where δ_{β_z} is a point mass centred at β_z. According to Equation (1), we can see that the probability to be distributed to a new table is associated with concentration parameter η, while the likelihood of an upcoming sample β_n to be allocated to an existing table z is proportional to the table size n_z. In other words, the table with a bigger size has more opportunity to serve the upcoming customer. Therefore, we can see DP has a clustering property ("The rich gets richer") for the online streaming data.

3.2 Temporal Point Process

A **Temporal Point Process (TPP)** is a stochastic process composed of a time series of events that occur in continuous time [24]. The temporal point process is widely used for modeling the temporal dynamics pattern behind a time-series of events that occur in continuous time, such as health-care analysis, earthquakes and aftershocks modeling, and social network analysis [7, 25, 38]. *Conditional intensity function*, which describes the instantaneous rate for the occurrence of an upcoming event, is one important metric to characterize temporal point process. Given an event time sequence $\mathcal{T} = \{t_1, \ldots, t_n\}$, the conditional intensity function at time t $(t > t_n)$ can be expressed as

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

where $N([t, t + dt])$ indicates the number of events occurred in a time interval dt, $\mathcal{H}_{t_n} = \{t_{n'} | t_n >= t_{n'}\}$ is the collection of historical events until time t_n, and symbol $*$ indicates that the intensity may rely on the history $\mathcal{H}_{t_n}$.

Besides $\lambda^*(t)$, *survival* is another significant metric to measure the probability that an upcoming event does not occur until time t, and is formulated as

$$S^*(t) = S(t|\mathcal{H}_{t_n}) = exp\left(-\int_{t_n}^{t} \lambda^*(\tau)d\tau\right). \tag{3}$$

The *conditional density function* at time t can be described as

$$f^*(t) = \lambda^*(t) \cdot S^*(t) = \lambda^*(t) \cdot exp\Big(-\int_{t_n}^{t} \lambda^*(\tau)d\tau\Big). \tag{4}$$

With an observation window $[0, T]$, the likelihood of the observed event time sequence $\mathcal{T} = \{t_1, \ldots, t_n\}$, s.t. $T > t_n$, is formulated as

$$\mathcal{L}_{\mathcal{T}} = \prod_{t_i \in \mathcal{T}} f^*(t_i) = \prod_{t_i \in \mathcal{T}} \lambda^*(t_i) \cdot exp\Big(-\int_{0}^{T} \lambda^*(\tau)d\tau\Big). \tag{5}$$

The **Hawkes process** is one category of temporal point processes and it has a distinctive feature: *self-excitation*, the occurrence likelihood of an upcoming event increases due to the previous events which have just occurred [10, 26]. In the Hawkes process, the conditional intensity function is defined as:

$$\lambda^*(t) = \lambda_0 + \sum_{t_i \in \mathcal{T}} \gamma(t, t_i), \tag{6}$$

where $\lambda_0 > 0$ is the base intensity which is independent of the historical events, $\gamma(t, t_i)$ is the triggering kernel that is usually a monotonically decreasing function to guarantee recent events have more influence on the occurrence of the upcoming event. The Hawkes process models the self-excitation phenomenon that a new event arrival increases the upcoming event's conditional intensity which then decreases back towards λ_0 gradually. The Hawkes process is widely used to model the cluster patterns, e.g., the information diffusion on online social networks or the earthquake occurrences.

Recently, several studies combine the Hawkes process and Dirichlet mixture model for clustering event streams [5, 21, 30, 35]. For example, research in [5] proposes a Dirichlet Hawkes process to cluster document streams, while research in [21] proposes a Hierarchical Dirichlet Hawkes process to identify the learning patterns from detailed traces of learning activity. In this article, our task is to detect whether the most recently observed activity from a user's activity sequence is malicious or not, rather than clustering streaming activities into groups. Our detection algorithm is based on Dirichlet Marked Hawkes process that adopts the categorical distribution to model activity type and uses mode to effectively capture user behavior pattern, thus achieving high detection accuracy.

4 FRAMEWORK

Considering an activity sequence $\mathcal{E} = \{e_1, \ldots, e_n, \ldots\}$, $e_n = (t_n, y_n)$ indicates the activity with type y_n occurring at time t_n, where $y_n \in \mathcal{Y} = \{1, \ldots, D\}$ and D is the number of activity types. Our insider threat detection task is to detect malicious activities that have different behavior patterns from normal ones. For this task, we develop a DMHP framework that is an infinite mixture model based on the Chinese restaurant process (Section 3.1). Our mixture model assumes that the activity stream $\mathcal{E}$ consists of unbounded hidden clusters $Z = \{z_k\}_{k=1}^{K}$ where $z_k \in \mathbb{Z}$ and K, like the number of occupied *tables* in CRP, represents the number of hidden clusters. Note that K changes dynamically along the activity stream. In our insider threat detection scenario, we refer to these hidden clusters as *user modes* and use the term *mode* throughout this article. Table 1 shows the summary of notations. Our DMHP adopts the Dirichlet process to determine the prior distribution of user modes. Each mode has its temporal dynamics in terms of activity type and time. We adopt one set of marked Hawkes processes to describe each mode. For each activity in the streaming sequence, we calculate a combined likelihood score based on the current K modes we have at hand and use the score to predict whether the activity is malicious.

Table 1. Summary of Notations

Symbol	Definition
(t_n, y_n)	The nth activity occurring at time t_n with activity type y_n.
z_n	The mode assignment for activity (t_n, y_n).
D	The category number of activity types.
K	The number of modes we have till t_n.
L	The number of triggering kernels for one specific mode.
λ_0	The base intensity of the Hawkes process.
λ^*, f^*, S^*	The conditional intensity, density, and survival functions, respectively.
$\lambda^*_{z_n}$	The conditional intensity of th4 Hawkes process for the mode z_n.
$(\alpha'_1, \ldots, \alpha'_L)$	The concentration parameter of Dirichlet prior for the Hawkes process.
$(\theta'_1, \ldots, \theta'_D)$	The concentration parameter of Dirichlet prior for categorical distribution.
$\boldsymbol{\alpha}_{z_n}$	The weight vector for L different time scale kernels.
$\boldsymbol{\theta}_{z_n}$	The parameter vector for categorical distribution with mode z_n.
$s(t_n, y_n)$	The combined score provided by the Dirichlet mode-specific model for (t_n, y_n).
ϵ	The insider activity detection threshold.
ϕ^c_n	The weight of particle c at timestamp n.

4.1 Marked Hawkes Process

In this section, we develop a marked Hawkes process model with Dirichlet priors to capture both *time* and *type* information.

Time. We adopt Hawkes process to model the *time* information. Given an observed activity stream $(t_1, y_1), \ldots, (t_{n-1}, y_{n-1})$, for an upcoming activity (t_n, y_n) with an unknown mode assignment z_n, based on Equation (4), the occurrence likelihood associated with *time* is measured by:

$$p(t_n|z_n, t_{1:n-1}) = \lambda^*_{z_n}(t_n) \cdot exp\Big(-\int_{t_{n-1}}^{t_n} \lambda^*_{z_n}(\tau)d\tau\Big), \tag{7}$$

and its intensity value is as follows:

$$\lambda^*_{z_n}(t_n) = \lambda_0 + \sum_{t_i<t_n} \gamma_{z_n}(t_n, t_i)\mathbb{1}[z_i = z_n], \tag{8}$$

where z_i refers to the mode assigned to the ith activity, and $\gamma_{z_n}(t_n, t_i)$ is the triggering function of mode z_n with the form

$$\gamma_{z_n}(t_n, t_i) = \sum_{l=1}^{L} \alpha^l_{z_n} \cdot \kappa(\tau_l, t_n - t_i), \tag{9}$$

$$\boldsymbol{\alpha}_{z_n} \sim Dir(\alpha'_1, \alpha'_2, \ldots, \alpha'_L), \tag{10}$$

$$\kappa(\tau_l, t_n - t_i) = \exp\left(-((t_n - t_i) - \tau_l)^2/2\sigma^2_l\right)/\sqrt{2\pi\sigma^2_l}, \tag{11}$$

where $Dir(\alpha'_1, \alpha'_2, \ldots, \alpha'_L) = \frac{\Gamma(\sum_l \alpha'_l)}{\prod_l \Gamma(\alpha'_l)} \prod_{l=1}^{L} \alpha_{z_n,l}^{\alpha'_l-1}$ $\kappa(\tau_l, t_n - t_i)$ is the Gaussian **radial basis function (RBF)** kernel function, $\boldsymbol{\alpha}_{z_n} \in \mathbb{R}^L$ is the weight vector for L different time scale kernels, $\boldsymbol{\alpha}' = (\alpha'_1, \alpha'_2, \ldots, \alpha'_L) \in \mathbb{R}^L$ are the corresponding hyperparameters of Dirichlet prior, and τ_l is the typical reference time points.

Type. We make use of the categorical distribution with a Dirichlet prior to model the upcoming activity type y_n underlying mode z_n

$$p(y_n|z_n, y_{1:n-1}) \sim Cat(\boldsymbol{\theta}_{z_n}), \tag{12}$$

$$\boldsymbol{\theta}_{z_n} \sim Dir(\theta'_1, \theta'_2, \dots, \theta'_D), \tag{13}$$

where $\boldsymbol{\theta}_{z_n} \in \mathbb{R}^D$ is the parameter vector for categorical distribution with mode z_n and $\boldsymbol{\theta'} = (\theta'_1, \theta'_2, \dots, \theta'_D) \in \mathbb{R}^D$ is the corresponding hyperparameters of Dirichlet prior. Then, based on the historical activity types, by the Dirichlet-Categorical conjugate relation, the occurrence likelihood associated with *type* can be concretely represented as

$$p(y_n = d | z_n, y_{1:n-1}) = \frac{C^{y_n=d, z_n} + \theta'_{y_n=d}}{C^{z_n} + \sum_{i=1}^{D} \theta'_i}, \tag{14}$$

where $C^{y_n=d,z_n}$ is the number of activities with type d in mode z_n excluding the current activity, $\theta'_{y_n=d}$ is the dth element of hyperparameter vector $\boldsymbol{\theta'}$ that is related to activity type y_n, and C^{z_n} is the total number of activities in mode z_n excluding the current activity.

Time & Type. Given the history of past activities, the conditional density function that the upcoming activity will happen at time t_n with type y_n can have different forms. In practice, we typically choose a simple factorized formulation to reduce the excessive complications caused by jointly and explicitly modeling the time and type information. In our article, by Equations (7) and (14), the joint occurrence likelihood of the upcoming activity (t_n, y_n) under user mode z_n provided by marked Hawkes process is formulated as

$$p(t_n, y_n | z_n, t_{1:n-1}, y_{1:n-1}) = p(y_n | z_n, y_{1:n-1}) \cdot p(t_n | z_n, t_{1:n-1}). \tag{15}$$

4.2 Dirichlet Marked Hawkes Process

Our proposed DMHP model is a mixture model consisting of a set of marked Hawkes processes. As a non-parametric Bayesian model, DMHP can be taken as a generative model. Its equivalent generative process for one user-specific activity stream can be described as follows:

(1) Initialize parameters: Base intensity λ_0, hyperparameters for Dirichlet priors, $(\alpha'_1, \alpha'_2, \dots, \alpha'_L)$ and $(\theta'_1, \theta'_2, \dots, \theta'_D)$
(2) Draw α_1 from Dir$(\alpha'_1, \alpha'_2, \dots, \alpha'_L)$ and θ_1 from Dir$(\theta'_1, \theta'_2, \dots, \theta'_D)$
(3) Draw t_1 from Poisson(λ_0) and $y_1 \sim$ Cat(θ_1)
(4) Assign $K \leftarrow 1$ and $z_1 \leftarrow K$
(5) For $n > 1$:
  (a) Draw time $t_n > t_{n-1}$ for the new activity from Poisson$(\lambda_0 + \sum_{i=1}^{n-1} \gamma_{z_i}(t_n, t_i))$
  (b) Draw the mode z_n for the new activity

$$p(z_n = k | t_{1:n}) = \begin{cases} \frac{\lambda^*_{z_n=k}(t_n)}{\lambda_0 + \sum_{i=1}^{n-1} \gamma_{z_i}(t_n, t_i)}, & k = 1, \dots, K \\ \frac{\lambda_0}{\lambda_0 + \sum_{i=1}^{n-1} \gamma_{z_i}(t_n, t_i)}, & k = K + 1 \end{cases} \tag{16}$$

  If $z_n = K + 1$, draw $\boldsymbol{\alpha}_{z_n}$ from Dir$(\alpha'_1, \alpha'_2, \dots, \alpha'_L)$, draw $\boldsymbol{\theta}_{z_n}$ from Dir$(\theta'_1, \theta'_2, \dots, \theta'_D)$, $K \leftarrow K + 1$.
  (c) Draw type for the new activity

$$y_n | z_n, y_{1:n-1} \sim \sum_d \frac{C^{y_n=d, z_n} + \theta'_{y_n=d}}{C^{z_n} + \sum_{i=1}^{D} \theta'_i} \cdot \delta_d, \tag{17}$$

  where δ_d is a point mass centred at activity type d.

In the above process, K is a mode counter to record the total number of modes we have until now, k refers to some certain mode, and z_n represents the mode assignment of the nth activity. In step 5(a), we first draw time t_n for the new activity by a Poisson process parameterized by an intensity value that is derived from the historical activities and current user modes. The generation of the activity timing in the process can be viewed as the superposition of a Poisson process λ_0 and multiple Hawkes processes. In step 5(b), we draw the mode z_n for the new activity with the probability mass function (shown in Equation (16)) that is constructed by Equation (8). If it is a new mode, we draw its weight vector $\boldsymbol{\alpha}_{z_n}$ for the L scale kernels, $\boldsymbol{\theta}_{z_n}$ to parameterize the categorical distribution and then increase K by one. In step 5(c), we draw type for the new activity with the mass function constructed by Equation (14).

4.3 Insider Malicious Activity Detection

In insider threat detection, our goal is to evaluate whether or not the most recently observed activity (t_n, y_n) is anomalous on the fly. The above Dirichlet Marked Hawkes process gives a basic idea on how a sequence of samples $\{(t_n, y_n)\}$ is generated. In particular, we show in Equation (16) of step 5(b) how to calculate the probability of mode z_n given the time sequence, i.e., $p(z_n = k|t_{1:n})$. In step 5(c), we further show in Equation (17) how to generate y_n given mode z_n and the previous type sequence $y_{1:n-1}$. This generative process gives us an idea on how to distribute the most recent observed activity (t_n, y_n) to user modes in a probabilistic way. Formally, we have:

$$p(z_n = k|t_{1:n}, y_{1:n}) \sim p(y_n|z_n = k, y_{1:n-1}) \cdot p(z_n = k|t_{1:n}). \tag{18}$$

We then derive a combined likelihood score for the most recent activity (t_n, y_n) given the previously observed activity stream $(t_1, y_1), \ldots, (t_{n-1}, y_{n-1})$. The score is based on current K modes and is formulated as

$$s(t_n, y_n) = \sum_{k=1}^{K} \underbrace{p(t_n, y_n|z_n = k, t_{1:n-1}, y_{1:n-1})}_{\text{intra-mode likelihood}} \cdot \underbrace{p(z_n = k|t_{1:n}, y_{1:n})}_{\text{mixture weight}}. \tag{19}$$

Note that $p(z_n = k|t_{1:n}, y_{1:n})$ measures the mixture weight of (t_n, y_n) from mode k given the whole activity sequence until (t_n, y_n) and $p(t_n, y_n|z_n = k, t_{1:n-1}, y_{1:n-1})$ is the probability of activity (t_n, y_n) occurring inside the given mode k. In our work, we assume *time* and *type* are conditionally independent given the user mode z_n and apply Equation (15) to reduce the excessive complications.

Algorithm 1 shows the framework of our insider activity detection. We assume the first activity (t_1, y_1) is normal. For each activity (t_n, y_n) s.t. $n > 1$, lines 3–7 illustrate whether a new mode is needed for the upcoming activity. Lines 8–12 are the loop to iterate all of modes we have at hand to calculate the corresponding likelihood scores under specific modes, the mixture weights, and the final combined likelihood score. In particular, line 9 is for the computation of likelihood given a specific mode k; line 10 is for calculating the mixture weights over all the modes. Then, a combined likelihood score of the activity is derived in line 12. Given a predefined threshold ϵ, lines 13–17 are to evaluate whether activity (t_n, y_n) is malicious or not: if the combined likelihood score is below ϵ, it is labeled as an insider activity; otherwise, it is normal. In general, the complexity of Algorithm 1 for online detection is O(K) [17].

Our DMHP detection framework consists of two components, *time* and *type*. For an ablation evaluation, we have listed three variants of our proposed model based on different input information:

(1) *Time model.* Only uses activity *time* information as input. Given Algorithm 1, we need do the following adaptions:
   (a) In line 9, to fix the likelihood score, we replace Equation (15) with Equation (7).
   (b) In line 10, for the computation of mixture weights, we rewrite Equation (18) as $p(z_n = k|t_{1:n}, y_{1:n}) \sim p(z_n = k|t_{1:n})$.
(2) *Type model.* Only uses *type* information as input. To formulate this model, we need do the following adaptions based on Algorithm 1:
   (a) In line 9, to derive the likelihood score, we replace Equation (15) with Equation (14).
   (b) In line 10, to calculate the mixture weights, we rewrite Equation (18) as $p(z_n = k|t_{1:n}, y_{1:n}) \sim p(z_n = k|y_{1:n})$.
(3) *Time&type model.* Takes both *time* and *type* information as input, which is depicted in Algorithm 1.

ALGORITHM 1: Insider Threat Detection via Dirichlet Marked Hawkes Process

Input: Activity stream $(t_1, y_1), \ldots, (t_n, y_n), \ldots$, concentration factors λ_0, $(\alpha'_1, \alpha'_2, \ldots, \alpha'_L)$, $(\theta'_1, \theta'_2, \ldots, \theta'_D)$, and insider likelihood threshold ϵ.
Output: Real-time insider activity detection report $\mathcal{Y}$: 1 represents the insider activity and 0 indicates the normal one.

```
1:  $\mathcal{Y} \leftarrow list[0]; K \leftarrow 1$
2:  for each activity $(t_n, y_n), n = 2, \ldots$ do
3:      Draw $r \sim$ Uniform(0,1)
4:      if $r < \frac{\lambda_0}{\lambda_0 + \sum_{i=1}^{n-1} \gamma_{z_i}(t_n, t_i)}$ then
5:          Generate a new mode by Equation (16)
6:          $K \leftarrow K + 1$
7:      end if
8:      for $k \in \{1, \ldots, K\}$ do
9:          Compute the likelihood of $(t_n, y_n)$ in mode $k$ by Equation (15)
10:         Compute the weight of mode $k$ for $(t_n, y_n)$ by Equation (18)
11:     end for
12:     Compute the combined likelihood score $s(t_n, y_n)$ by Equation (19)
13:     if $s(t_n, y_n) < \epsilon$ then
14:         $\mathcal{Y}$.insert(1)
15:     else
16:         $\mathcal{Y}$.insert(0)
17:     end if
18: end for
19: return $\mathcal{Y}$
```

4.4 Inference

Given an observed activity stream $(t_1, y_1), \ldots, (t_{n-1}, y_{n-1})$, our goal is to judge whether the most recent activity (t_n, y_n) is abnormal via a combined likelihood score. In this *on-the-fly* detection, one challenge is how to efficiently derive the latent variables, *user mode* z_n, from the posterior distribution $p(z_n = k | t_{1:n}, y_{1:n})$ (Equation (18)). To address this issue, we apply a **Sequential Monte Carlo (SMC)** method [22] to infer the latent variables in our DMHP framework. Generally speaking, the inference algorithm consists of two parts: *sampling user modes* and *updating triggering kernel*.

Sampling User Mode. An SMC is used to get an approximation for the posterior $p(z_n | t_{1:n}, y_{1:n})$ in which a number of particles are maintained, and each particle refers to an instance of latent variable (i.e., a hidden mode) and has a weight to tell how well this instance can interpret the data. The weight of particle c at timestamp n goes as follows:

$$\phi_n^c = \frac{p(z_{1:n} | y_{1:n}, t_{1:n})}{q(z_{1:n} | y_{1:n}, t_{1:n})}, \tag{20}$$

where $c \in \{1, \ldots, C\}$ and C is the total number of particles; $p(\cdot)$ is the true posterior distribution, and $q(\cdot)$ is the corresponding proposal distribution.

To minimize the variance of the resulting particle weight, we take the posterior distribution $p(z_n | z_{1:n-1}, y_{1:n}, t_{1:n})$ as the proposal distribution $q(\cdot)$. Then, given the weight of particle c at time-stamp $n-1$, i.e. ϕ_{n-1}^c, the unnormalized weight ϕ_n^c can be recursively updated by

$$\phi_n^c = \phi_{n-1}^c \cdot p(y_n | z_n^c, y_{1:n-1}). \tag{21}$$

Updating Triggering Kernel. Updating Triggering Kernel. For a stream of activities $(t_1, y_1), \ldots, (t_n, y_n)$, with current user activity (t_n, y_n), time parameter $\boldsymbol{\alpha}_{z_n}$ should be updated before the next activity is coming. Following the work in [5], a set of samples $\boldsymbol{\alpha}$ are drawn from a Dirichlet distribution, $\{\boldsymbol{\alpha}_{z_n}^i\}_{i=1}^N \sim Dir(\alpha'_1, \alpha'_2, \ldots, \alpha'_L)$, where

ALGORITHM 2: Inference Algorithm of the DMHP

```
1:  Initialize $\phi_1^c$ to $\frac{1}{C}$ for all $c \in \{1 \dots C\}$
2:  for each activity timing $t_n, n = 1, 2, \dots$ do
3:      for $c \in \{1, \dots, C\}$ do
4:          if one activity type $y_n$ occurs at the time $t_n$ then
5:              sample mode $z_n$ with Equation (18)
6:              update triggering kernel parameter $\boldsymbol{\alpha}$ by the sampling method stated above.
7:              update particle weights with Equation (21)
8:          end if
9:      end for
10:     Normalize particle weights
11:     if $||\phi_n||_2^{-2} < threshold$ then
12:         resample particles;
13:     end if
14: end for
```

N is the sampling size, and these $\boldsymbol{\alpha}$s are used to obtain an estimation of the time parameter via maximizing the likelihood of activity streaming in terms of *time* information (Equation (5)).

Algorithm 2 shows the sequential Monte Carlo framework for mode sampling and triggering kernel updating. Line 1 initializes the particles and, for one specific activity, lines 3–9 iterate all particles to sample the mode and update the time parameter: line 5 is for the mode sampling, line 6 aims to trigger kernel updating, and line 7 is to update particle weights. In addition, lines 11–13 show the rules for particle resampling after the particle weight normalization (line 10). The Algorithm 2 preserves complexity of O(C) [22].

5 EXPERIMENTS

5.1 Experiment Setup

Dataset. We adopt the CERT Insider Threat Dataset [9], which is the only publicly available dataset, to evaluate insider threat detection. In Section 5.4, we apply our model for detecting vandalism on Wikipedia and report evaluation results on the UMD Wikipedia dataset [15]. The CERT dataset consists of five log files that record the computer-based activities for all employees: **logon.csv** that records 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 operations (visit, download, or upload), **device.csv** that records the usage of a thumb drive (connect or disconnect) and **file.csv** that records activities (open, write, copy or delete) on a removable media device. To describe employees' action behavior, instead of the five original coarse-grained categories for activities, we apply 18 fine-grained categories to describe the employees' operations. For example, based on the occurrence time, *logon* can be detailed as *Weekday_Logon_Normal*, *Weekday_Logon_After*, *Weekend_Logon* and *Logoff*. We show the 18 fine-grained categories in Table 2.

The CERT dataset contains 3,995 benign employees and 5 insiders. For each user, it records activities from January 2010 to June 2011. On average, the number of activities for each employee is around 40,000. We preprocess the original dataset by joining all the log files, grouping them by each employee, and then sorting the activities of each employee based on the recorded time- stamps. We include all five insiders and randomly select five benign employees in our experiment. We can see that from Table 3 that malicious activities take up a very small percentage of all activities: ACM2278 only has 0.0701% (22/31,370) malicious activities; CDE1846 only has 0.3549% (134/37,754) malicious activities, CMP2946 only has 0.3920% (243/61,989) malicious activities, MBG3183 only has 0.0094% (4/42,438) malicious activities, and PLJ1771 only has 0.0858% (18/20,964) malicious activities. It is challenging to detect effective insider threat detection algorithm due to the extremely low percentage of malicious activities.

Table 2. Operation Types Extracted from Log Files

Files	Operation Types
logon.csv	Weekday_Logon_Normal (on weekdays, employees log on in working hours.)
	Weekday_Logon_After (on weekdays, employees log on out of working hours.)
	Weekend_Logon (employees log on at weekends)
	Logoff (employees log off)
email.csv	Email_In (an email operation between employees in the same company)
	Email_Out (an email operation between one company employees and people out of this company)
http.csv	WWW_Visit (employees visit a website)
	WWW_Download (employees download files from a website)
	WWW_Upload (employees upload files to a website)
device.csv	Connect_normal (a device connection operation occurs on a weekday at working hours)
	Connect_after (a device connection operation occurs on a weekday after working hours)
	Connect_weekend (a device connection operation occurs on weekends)
	Disconnect (a device disconnection operation occurs)
file.csv	File_exe (employees do some operations on an executive file)
	File_jpg (employees do some operations on a picture with extended name .jpg)
	File_zip (employees do some operations on a compressed file with extended name .zip)
	File_txt (employees do some operations on a plain text file)
	File_doc (employees do some operations on a text file with extended name .doc)

Table 3. Statistics of Malicious Users

	ACM2278	CDE1846	CMP2946	MBG3183	PLJ1771
# of Malicious Activities	22	134	243	4	18
# of Total Activities	31370	37754	61989	42438	20964

In our experiment, we treat each employee's activities as one sequence, each of which is sorted by the activity occurrence time, and we have ten streams which are from five insiders above and five normal users randomly selected. For each activity sequence, we feed it into our proposed DMHP algorithm and then obtain the combined likelihood score of each activity from one specific employee dynamically. Given an empirical threshold ϵ, each activity in the stream can be judged by the real-time combined likelihood score: it is labeled as an insider activity if the corresponding combined likelihood score is less than τ; otherwise, it is labeled as a normal activity.

Hyperparameters. For the *time* model, we set the base intensity of Hawkes process $\lambda_0 = 0.1$, the number of RBF Kernels $L = 7$, the relevant reference time points as 3, 7, 11, 24, 48, 96, 192 hours to capture both short-term and long-term dependence of user activities, and the corresponding bandwidths are 1, 5, 7, 12, 24, 24, 24. $Dir(\alpha'_1, \alpha'_2, \ldots, \alpha'_L)$ is a symmetric Dirichlet distribution with the concentration parameter value 0.1. For the *type* mode, $Dir(\theta'_1, \theta'_2, \ldots, \theta'_D)$ is one symmetric Dirichlet distribution with the concentration parameter value 0.01. We do not specify the exact value of likelihood threshold ϵ; instead, we rank each activity based on its likelihood value and report the number of real malicious activities in the certain percentage of ranked activities. For the reproducibility, the source code is available online.[1]

Baselines. Our DMHP *time&type* uses both type and time information in the modeling. In our evaluation, we compare it with five baselines, DMHP *time*, DMHP *type*, Isolation Forest [19], Local Outlier Factor [14], and ADeMS [12]. DMHP *time* only uses time information as input, and DMHP *type* only uses activity type information. Hence, the comparison with these two baselines can be considered as an ablation study. Isolation Forest and Local Outlier Factor are two typical unsupervised anomaly detection methods. Isolation Forest detects anomalies

[1]https://github.com/PanpanZheng/DMHP.

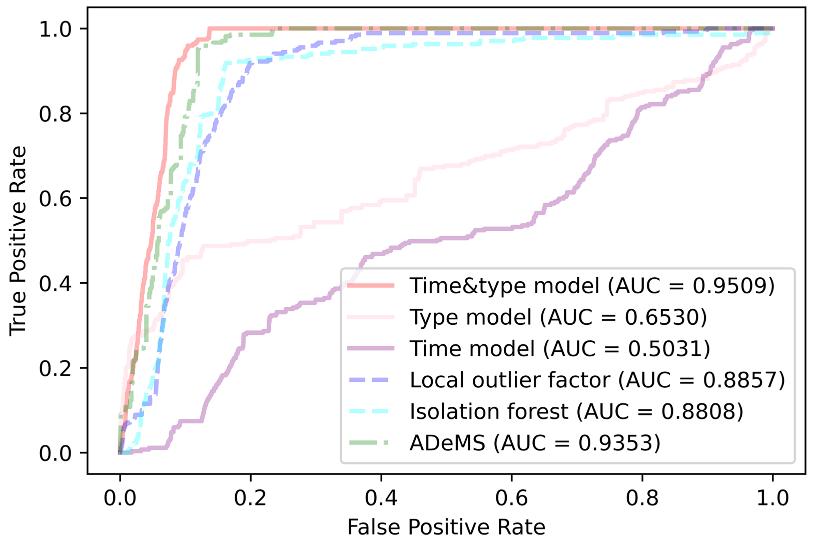

Fig. 1. ROC curve for malicious activity detection.

based on the concept of isolation without employing distance or density metrics whereas Local Outlier Factor identifies anomalies by measuring the local deviation of a given data point with respect to its neighbors. ADeMS is a state-of-the-art streaming-based anomaly detection algorithm. ADeMS maintains a set of few orthogonal vectors to capture the prior from the historical non-anomalous data points and use a reconstruction error test to evaluate the anomaly of upcoming data. On the implementation, we use scikit-learn for Isolation Forest and Local Outlier Factor, and the GitHub implementation[2] for ADeMS. To build the inputs of Isolation Forest and Local Outlier Factor, we encode each activity by a vector with 43 dimensions, in which we adopt the one-hot vector to represent the 24 hours in a day as time information and 18 activity types as type information, while one dimension to indicate the time gap since the previous activity. For the baseline ADeMS, the input vector does not include the time gap information because including the time gap information would damage the performance in the CERT dataset, based on our experiments.

Training & Testing. The activities in the first 1.5 months are used to train the proposed model and baselines. We think the dynamic models would stabilize after training with activities of the first 1.5 months. All the rest activities (16.5 months) are employed for evaluation. In all models, we also treat all *WWW_visit* activities as normal as *WWW_visit* activities trivially contribute to the insider threat detection.

5.2 Malicious Insider Activity Detection

Figure 1 shows the **receiver operating characteristic (ROC)** curves and the corresponding area under the ROC curve (AUC) values of our DMHP models (*time&type*, *type*, *time*), Local Outlier Factor, and Isolation Forest. We can see that our DMHP *time&type* achieves the best accuracy with the AUC value of 0.9509, which is significantly higher than *time* (AUC 0.5031) and *type* (AUC 0.6530). The reason behind this can summarized as follows: First, our *time* model is a self-exciting Hawkes process that the occurrence likelihood of a new event increases due to just occurred events (Equation (6)); therefore, if a series of malicious activities occur consecutively in a very short period of time, our *time* model tends to provide high likelihoods for these malicious activities. Second, in general,

[2]https://github.com/takuti/stream-anomaly-detect.

Table 4. Category-Specific AUC Values

	Logon	Email	Http	Device	File
Local outlier factor	0.8789	0.8568	0.8852	0.8945	0.9208
Isolation forest	0.8524	0.9127	0.7605	0.9012	0.8637
ADeMS	0.9437	0.9204	0.9533	0.9334	0.9457
Time model	0.4982	0.5970	0.4405	0.4392	0.4738
Type model	0.6931	0.4931	0.7616	0.6884	0.7907
Time&type model	**0.9592**	**0.9314**	**0.9844**	**0.9535**	**0.9585**

Table 5. The Number of Detected Malicious Activities in the Top 15% of Total Activities with the Lowest Likelihood Scores

	ACM2278	CDE1846	CMP2946	MBG3183	PLJ1771
# of Total Activities	31,370	37,754	61,989	42,438	20,964
# of Malicious Activities	22	134	94	4	15
Isolation forest	8	133	77	0	11
Local outlier factor	16	114	54	3	13
ADeMS	22	130	72	4	11
Time model	0	26	13	0	0
Type model	22	76	23	4	9
Time&type model	**22**	**134**	**91**	**4**	**15**

our *type* model is a frequency-based model that it tends to assign a high (low) likelihood to the activity type with a high (low) frequency. Notably, in the CERT dataset, some malicious activities are of high-frequency, e.g., *email-in*, *email-out* or *website visit*, which makes the *type* model difficult to achieve high accuracy. However, as shown in Figure 1, our *time&type* model, which combines the *time* and *type* models, assigns a smaller likelihood to the malicious activity. This demonstrates the advantage of combining both activity type and time information in sequence modeling for insider threat detection. In addition, the *time&type* model also significantly outperforms Local Outlier Factor (AUC 0.8857), Isolation Forest (AUC 0.8808), and ADeMS (AUC 0.9353).

For a comprehensive evaluation, we report in Table 4 the corresponding AUC values from all compared models for each of the five *log* categories, i.e., *logon*, *email*, *http*, *device*, and *file*. We can observe that *time&type* obtains the highest AUC values in all five categories, *logon* 0.9592, *email* 0.9314, *http* 0.9844, *device* 0.9535, and *file* 0.9585. For baselines, ADeMS performs well on all activity types, while Isolation Forest achieves good AUC values in *email* and *device* but poor AUC values in *logon*, *http* and *file*. We also observe that DMHP with time information alone does not achieve good performance across all five categories.

We also show in Table 5 a detailed comparison of model performance for each insider. We calculate the number of detected malicious activities in the top 15% activities with the lowest likelihood scores from each model. As shown in Rows 2 and 3, the malicious activities take up a very small percentage of activities of insiders. Moreover, those malicious activities are often hidden among normal activities and occur in a long period. Our *time&type* achieves highest recalls for all insiders except *CMP2946*. Concretely, it captures 22 real malicious activities for *ACM2278* with recall 22/22 = 100%, 128 real malicious activities for *CDE1846* with recall 134/134 = 100%, 91 real malicious activities for *CMP2946* with recall 91/94 = 96.80%, 4 real malicious activities for *MBG3183* with recall 4/4 = 100%, 15 real malicious activities for *PLJ1771* with recall 15/15 = 100%. Generally, ADeMs also performs well in the malicious activity detection. Isolation Forest achieves its highest recall for *CDE1846*. However, it

Table 6. Malicious Activities of *ACM2278* (Activities 1-12 and 13-22 form Two Attack Instances)

ID	Malicious Activities		ID	Malicious Activities	
1	08-18 21:47:42	Weekday_Logon_After	12	08-19 06:10:59	Logoff
2	08-18 22:59:20	Connect_After	13	08-24 01:02:58	Weekday_Logon_After
3	08-19 01:34:19	WebsiteUpload	14	08-24 03:24:16	Connect_After
4	08-19 01:34:19	File_jpg	15	08-24 03:48:51	WebsiteUpload
5	08-19 01:37:20	WebsiteUpload	16	08-24 03:48:51	File_jpg
6	08-19 01:37:20	File_jpg	17	08-24 03:43:48	WebsiteUpload
7	08-19 01:38:10	WebsiteUpload	18	08-24 03:43:48	File_doc
8	08-19 01:38:10	File_txt	19	08-24 03:34:21	WebsiteUpload
9	08-19 01:46:04	WebsiteUpload	20	08-24 03:34:21	File_txt
10	08-19 01:46:04	File_zip	21	08-24 04:15:32	Disconnect
11	08-19 05:23:05	Disconnect	22	08-24 04:20:39	Logoff

Table 7. Detected Malicious Activities of *ACM2278* in Top 3% of Activities with the Lowest Likelihood Scores

	Detected Activity #	Detected Activity IDs
Isolation Forest	2	11, 12
Local Outlier Factor	7	2, 11, 12, 14, 17, 21, 22
ADeMS	22	1–22
Time model	0	-
Type model	19	1–6, 8, 10–17, 19–22
Time&type model	22	1–22

obtains fairly low recall values for other insiders, e.g., *ACM2278* (36.36%) and *MBG3183* (0%). This is because malicious activities in *ACM2278* and *MBG3183* are heavily associated with *http*, and Isolation Forest seems to be not good at capturing the malicious *http* behavior pattern, which is demonstrated by the comparably lower AUC in Table 4.

5.3 Insider Threat Detection: Case Study

In accordance with Table 5, the high recall values provided by DMHP *time&type* model in the top 15% activities lead us to a hypothesis that DMHP *time&type* has a tendency to assign low (high) combined likelihood scores to the malicious (normal) activities. To further testify our hypothesis and provide a clear picture for malicious activity detection, we focus on one insider, *ACM2278*, as a case study.

With the scenario design of CERT dataset, we learn that *ACM2278* refers to *an insider 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 and then, leaves the organization shortly thereafter.* There are 22 malicious activities carried by *ACM2278*, which form two concrete instances of insider behavior pattern: one instance is from activities 1 to 12 and another is from activities 13 to 22. For reproducibility purpose, we list them in Table 6. In both attack instances, the insider logs on the company's computer after work, connects the removable drives to computer, uploads data, and finally ends at the next day's morning before the working hour.

Table 7 shows the performance of insider threat detection provided by all models for *ACM2278*. We report the recall value calculated in the top 3% percent of activities with the lowest combined likelihood scores from each model. In accordance with Table 7, we can see that DMHP *time&type* model and ADeMS both perform well

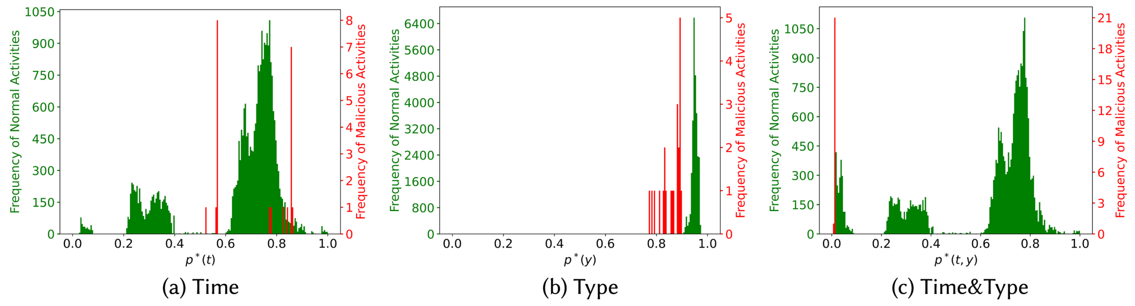

Fig. 2. Histogram of likelihood scores of activities from *ACM2278* in terms of time, type, and both time and type.

and they detect all 22 malicious insider activities in the top 3% percent of alerted activities; *type* model performs better than the other two baselines with recall 19/22 = 86.36% while *time* and Isolation Forest perform poor.

Empirical Distribution of Occurrence Likelihoods. In Figure 2, the green histograms indicate occurrence likelihoods for normal activities of *ACM2278*, while the red histograms indicate occurrence likelihoods for malicious activities.

Figure 2(a) shows that the likelihood scores of malicious activities derived from the *time* model mainly lie in two intervals [0.5, 0.6] and [0.7, 0.9]. Similarly, we can observe that the empirical distribution of normal activities derived from the *time* model roughly has two peaks with 0.3 and 0.7 as centers. In this case, likelihood scores of malicious and normal activities are interleaved together and it is hard to find a good threshold to separate them clearly, which results in a low AUC for *time* model.

On the contrary, Figure 2(b) shows that likelihood scores of insider-threat malicious activities for *type* ranges between 0.7 and 0.9, while the ones of normal activities mainly spread from 0.9 to 1.0. Hence, we can notice that *type* model can generally separate malicious and normal activities, which results in a good recall for *ACM2278* shown in Table 5. From Figure 2(c), we can clearly observe that the combined likelihood scores of malicious activities from the *time&type* model concentrates on the extreme left part of the empirical distribution of normal activities. As a result, *time&type* model produces a high AUC value as shown in Figure 1.

5.4 Wikipedia Vandal Detection

In this subsection, to further show the advantage of our DMHP model, we conduct a different task, i.e., detecting vandalism on Wikipedia by using the UMDWikipedia dataset [15]. In Wikipedia, each article page is edited by users including certified editors and volunteers. Each edit contains various attributes such as page ID, title, time, categories, page type, revert status, and content. Because Wikipedia is based on crowdsourcing mechanism and applies the freedom-to-edit model (i.e., any user can edit any article), it leads to a rapid growth of vandalism (deliberate attempts to damage or compromise integrity). Those vandals who commit acts of vandalism can be considered as insiders in the community of Wikipedia contributors. It is worth noting that user edits in Wikipedia are in principle similar to user activities in insider threat detection that are recorded in log files. Hence, we can leverage log files from Wikipedia to detect the malicious activities conducted by the vandals.

In this experiment, we focus on identifying malicious activities on Wikipedia pages. Because DMHP needs a number of historical activities to capture the patterns, we selected three Wikipedia pages, “List of metro systems”, “Star Trek Into Darkness”, and “St. Louis Cardinals”, which have a large number of edits on the UMDWikipedia dataset. For each page, we collect all edits from its contributors (including vandals). Each page has more than 300 edits and at least three malicious edits. Specifically, if an edit is reverted by bots or certified editors, we consider it malicious. We define two types of activities based on whether the edit is on a content page or a meta page. The reason we treat the edits of each page (rather than of a contributor) as a sequence is to have the ground

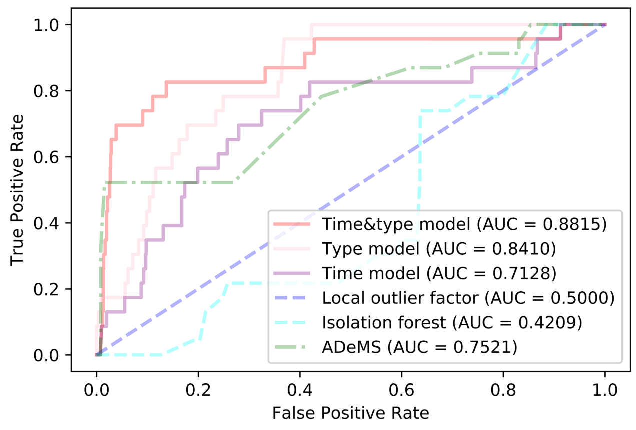

Fig. 3. ROC curve for malicious edit detection.

Table 8. The Performance of Malicious Activity and Hidden User Detection with Various λ_0 Values on Page "List of Metro Systems"

λ_0	Predicted User #	AUC	AMI	V-measure	Homogeneity	NMI
0.00001	10	0.8440	0.4446	0.6287	1.0000	0.6771
0.001	16	0.8464	0.3146	0.5001	1.0000	0.5774
0.01	16	0.8464	0.3146	0.5001	1.0000	0.5774
0.1	20	0.8815	0.2352	0.4075	1.0000	0.5058
0.2	22	0.8031	0.2163	0.3840	1.0000	0.4875
0.3	32	0.6719	0.1464	0.2887	1.0000	0.4107

truth of user modes. In this setting, the number of user modes is the same as the number of contributors and the edits from the same contributor can be considered as activities under a user mode in DMHP. To build the inputs of baselines, we represent each activity by a vector with 26 dimensions, in which 24 dimensions are used to represent 24 hours in a day as time information and 2 dimension are used to represent activity types.

Experimental Results. Figure 3 shows the ROC curve of malicious edit detection. We can observe that the *time&type* model significantly outperforms baselines in terms of AUC on malicious edit detection. Similar to results on the CERT dataset, with combining activity types and time information, *time&type* achieves higher AUC value than the model using activity type or time alone. We also observe that, in contrast with the CERT dataset, baselines (especially Isolation Forest) achieve poor performances on Wikipedia dataset. This is because Isolation Forest is based on attribute-wise analysis so that it tends to fail when the distribution for anomalous points becomes less discriminative, e.g., if the anomalous and non-anomalous points share the similar attribute range or distribution [13]. Compared with the CERT dataset, activity-type information in the Wikipedia dataset is subtle with only the choices of *meta page* or not, which is not sufficient to discriminate whether or not an activity is malicious. We further select the page "List of metro systems" for our case study. The edit sequence consists of 396 edits from 6 contributors, among which there are 20 malicious edits. Table 8 shows the performance of malicious edit detection with various λ_0 values. Our *time&type* model achieves good performance on detecting malicious edits on the page with AUC = 0.8815 when $\lambda_0 = 0.1$. A further investigation shows that our *time&type* model can detect 12 out of 20 malicious edits in 10% of total edits with the lowest combined likelihood scores.

Sensitivity of Hyperparameters. We further study whether the detection performance of our DMHP is sensitive to hyperparameters. From Equation (16), we can see that the base intensity λ_0 mainly determines whether a new *mode* is created or not for an upcoming activity. In other words, λ_0 affects the number of *modes* which may further affect the detection performance. To evaluate the performance of DMHP with various λ_0 values, we use the Wikipedia page "List of metro systems" as an illustrative example. We can observe from columns 1 and 2 in Table 8 that the number of *modes* increases with the increment of λ_0. This is because, as shown in Equation (16), the bigger λ_0 is, the higher probability of creating a new *cluster* will be. Concretely, the empirical quantity relation between the number of *modes* and λ_0 follows the rule: as λ_0 approaches 0, the number of *modes* goes to 1, while it goes to the total number of activities in the stream as λ_0 approaches $+\infty$ [33].

To investigate the relationship between the number of *modes* and the detection accuracy, we report the changes of AUC values with variation of λ_0 in column 3 of Table 8. We can see when λ_0 ranges from 0.00001 to 0.1, the AUC values do not change much although the number of predicted hidden users (*modes*) increases accordingly. This is because, as λ_0 is 0.00001, the number of predicted users is 10, which is larger than the ground-truth number of real contributors. In other words, the model tries to distinguish different editing styles in a fine-grained manner even if these edits are actually from the same user. As a result, the AUC values slightly increase from 0.8440 to 0.8815. However, if the λ_0 keeps increasing, we can observe a significant reduction of AUC values. This is because the number of the predicted user is too large compared with ground-truth, which is out of a reasonable range. In this case, the model has already been overfitted, so it is very important to choose an appropriate value for λ_0 which is sensitive.

To study why the number of predicted hidden users (modes) does not affect much the detection performance when λ_0 ranges from 0.00001 to 0.1, we further adopt four clustering metrics, *Adjusted Mutual Information (AMI), V-measure, Homogeneity,* and *Normalized Mutual Information (NMI)*, to evaluate the performance of our *time&type* model on the cluster result. Note that we have the ground truth about which user (*mode*) each edit is from for UMDWikipedia data. From Table 8, we can observe that with the increasing of the predicted number of *modes*, the values of all clustering metrics except *Homogeneity* decrease. However, the Homogeneity scores are always 1 with different λ_0 values, which indicates that each mode contains edits from one single user. This explains why the detection performance in terms of AUC is insensitive to the parameter value of the base intensity λ_0 in the range of $(0.00001, 0.1)$.

6 CONCLUSIONS AND FUTURE WORK

In this article, we have developed the DMHP framework for insider threat detection. DMHP adopts the Dirichlet process as a prior to identify the underlying user modes of a user activity sequence. For each user mode, the marked Hawkes process is adopted to model the user activities in terms of type and temporal information. DMHP is able to detect malicious activities in real-time by modeling the occurring likelihood of user activities such that a low likelihood of an activity could indicate a suspicious malicious activity. Our experiments on two datasets show that DMHP can assign low likelihoods to malicious activities and achieve good performance in terms of AUC.

Our DMHP framework can seamlessly integrate more information about user activities. For example, we can incorporate topic modeling in DMHP to model the content information such as emails or visited Web pages. We plan to extend our framework to capture those content and contextual information in our future work. We also emphasize that our introduced *user mode* in DMHP can help accurately model a user's activity sequence because in practice one user often has multiple roles and different working patterns. In our future work, we will conduct more studies to learn how various parameter settings in DMHP (e.g., base intensity, concentration parameters, reference time) affect the number of *modes* and the detection accuracy. We will further evaluate the statistical significance of our model over baselines under different parameter settings. Moreover, in our DMHP model, we introduce the use of a user-specified threshold ϵ to judge whether the most recent activity is malicious or not. In fact, it is difficult to specify the appropriate threshold value as ϵ is different for different users and should also dynamically change along the time. We plan to study how to adjust ϵ values dynamically based on the capacity of examining the alerted fraudulent activities each day. Finally, we are interested in how to improve efficiency of DMHP models (e.g., by using different approximations for posteriors).

REFERENCES

[1] Charles E. Antoniak. 1974. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics* (1974), 1152–1174.

[2] David M. Blei and Michael I. Jordan. 2006. Variational inference for Dirichlet process mixtures. *Bayesian Analysis* 1, 1 (2006), 121–143.

[3] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3, Jan (2003), 993–1022.

[4] CSO, CMU, and ForcePoint. 2018. *2018 U.S. State of Cybercrime*. Technical Report.

[5] Nan Du, Mehrdad Farajtabar, Amr Ahmed, Alexander J. Smola, and Le Song. 2015. Dirichlet-Hawkes processes with applications to clustering continuous-time document streams. In *21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 219–228.

[6] Hoda Eldardiry, Evgeniy Bart, Juan Liu, John Hanley, Bob Price, and Oliver Brdiczka. 2013. Multi-domain information fusion for insider threat detection. In *2013 IEEE Security and Privacy Workshops*. IEEE, 45–51.

[7] Mehrdad Farajtabar. 2018. *Point Process Modeling and Optimization of Social Networks*. Ph.D. Dissertation.

[8] Samuel J. Gershman and David M. Blei. 2012. A tutorial on Bayesian nonparametric models. *Journal of Mathematical Psychology* 56, 1 (2012), 1–12.

[9] J. Glasser and B. Lindauer. 2013. Bridging the gap: A pragmatic approach to generating insider threat data. In *IEEE Security and Privacy Workshops*.

[10] Alan G. Hawkes. 1971. Spectra of some self-exciting and mutually exciting point processes. *Biometrika* 58, 1 (1971), 83–90.

[11] Ivan Homoliak, Flavio Toffalini, Juan Guarnizo, Yuval Elovici, and Martín Ochoa. 2019. Insight into insiders and IT: A survey of insider threat taxonomies, analysis, modeling, and countermeasures. *ACM Comput. Surv.* 52, 2 (2019).

[12] Hao Huang and Shiva Kasiviswanathan. 2016. Streaming anomaly detection using randomized matrix sketching. In *VLDB Endowment* 9.

[13] H. Huang, H. Qin, S. Yoo, and D. Yu. 2012. A new anomaly detection algorithm based on quantum mechanics. In *2012 IEEE 12th International Conference on Data Mining*. 900–905.

[14] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. 2009. LoOP: Local outlier probabilities. In *18th ACM Conference on Information and Knowledge Management*. ACM, 1649–1652.

[15] Srijan Kumar, Francesca Spezzano, and V. S. Subrahmanian. 2015. Vews: A Wikipedia vandal early warning system. In *21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[16] D. C. Le and A. N. Zincir-Heywood. 2018. Evaluating insider threat detection workflow using supervised and unsupervised learning. In *2018 IEEE Security and Privacy Workshops*.

[17] C. L. Liu, Tsai T. H., and Lee C. H. 2014. Online Chinese restaurant process. In *20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 591–600.

[18] 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 *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS'19)*. Association for Computing Machinery, New York, 1777–1794. https://doi.org/10.1145/3319535.3363224

[19] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 8th IEEE International Conference on Data Mining*. IEEE, 413–422.

[20] L. Liu, O. De Vel, Q. Han, J. Zhang, and Y. Xiang. 2018. Detecting and preventing cyber insider threats: A survey. *IEEE Communications Surveys Tutorials* 20, 2 (2018), 1397–1417.

[21] Charalampos Mavroforakis, Isabel Valera, and Manuel Gomez-Rodriguez. 2017. Modeling the dynamics of learning activity on the web. In *26th International Conference on World Wide Web*.

[22] Del Moral, Arnaud Doucet, and Ajay Jasra. 2006. Sequential monte carlo samplers. *J. Royal Stat. Soc. B.* 411–436.

[23] 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 *8th ACM CCS International Workshop on Managing Insider Security Threats*.

[24] Jakob Gulddahl Rasmussen. 2018. Lecture notes: Temporal point processes and the conditional intensity function. *arXiv:1806.00221 [stat]* (2018).

[25] Alex Reinhart. 2017. A review of self-exciting spatio-temporal point processes and their applications. *arXiv:1708.02647 [stat]* (2017).

[26] Marian-Andrei Rizoiu, Young Lee, Swapnil Mishra, and Lexing Xie. 2017. A tutorial on Hawkes processes for events in social media. *arXiv preprint arXiv:1708.06401* (2017).

[27] Malek Ben Salem, Shlomo Hershkop, and Salvatore J. Stolfo. 2008. A survey of insider attack detection research. In *Insider Attack and Cyber Security: Beyond the Hacker*.

[28] Ameya Sanzgiri and Dipankar Dasgupta. 2016. Classification of insider threat detection techniques. In *11th Annual Cyber and Information Security Research Conference*.

[29] Ted E. Senator, Henry G. Goldberg, Alex Memory, William T. Young, Brad Rees, Robert Pierce, Daniel Huang, Matthew Reardon, David A. Bader, Edmond Chow, Irfan Essa, Joshua Jones, Vinay Bettadapura, Duen Horng Chau, Oded Green, Oguz Kaya, Anita Zakrzewska, Erica Briscoe, Rudolph IV L. Mappus, Robert McColl, Lora Weiss, Thomas G. Dietterich, Alan Fern, Weng–Keen Wong, Shubhomoy Das, Andrew Emmott, Jed Irvine, Jay-Yoon Lee, Danai Koutra, Christos Faloutsos, Daniel Corkill, Lisa Friedland, Amanda Gentzel, and David Jensen. 2013. 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 (KDD'13)*. Association for Computing Machinery, New York, NY, USA, 1393–1401.

[30] Yeon Seonwoo, Alice Oh, and Sungjoon Park. 2018. Hierarchical dirichlet gaussian marked Hawkes process for narrative reconstruction in continuous time domain. In *2018 Conference on Empirical Methods in Natural Language Processing*. 3316–3325.

[31] L. Spitzner. 2003. Honeypots: Catching the insider threat. In *19th Annual Computer Security Applications Conference*.

[32] Yee Whye Teh. 2010. Dirichlet Process. In *Encyclopedia of Machine Learning*. Springer, 280–287.

[33] Yee W. Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2005. Sharing clusters among related groups: Hierarchical Dirichlet processes. In *Advances in Neural Information Processing Systems*. 1385–1392.

[34] 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 *AI for Cyber Security Workshop*.

[35] Hongteng Xu and Hongyuan Zha. 2017. A Dirichlet mixture model of Hawkes processes for event sequence clustering. In *Advances in Neural Information Processing Systems*. 1354–1363.

[36] Shuhan Yuan and Xintao Wu. 2021. Deep learning for insider threat detection: Review, challenges and opportunities. *Computers & Security* 104 (2021), 1–14. https://doi.org/10.1016/j.cose.2021.102221

[37] Shuhan Yuan, Panpan Zheng, Xintao Wu, and Hanghang Tong. 2020. Few-shot insider threat detection. In *29th ACM International Conference on Information and Knowledge Management (CIKM'20)*. Association for Computing Machinery, New York, 2289–2292. https://doi.org/10.1145/3340531.3412161

[38] Zhengyi Zhou, David S. Matteson, Dawn B. Woodard, Shane G. Henderson, and Athanasios C. Micheas. 2015. A spatio-temporal point process model for ambulance demand. *J. Amer. Statist. Assoc.* 110, 509 (2015), 6–15.

Received November 2020; revised Februray 2021; accepted March 2021