Hindawi
Security and Communication Networks
Volume 2021, Article ID 9937229, 11 pages
https://doi.org/10.1155/2021/9937229

Research Article

A Deep Learning-Based Trust Assessment Method for Cloud Users

Wei Ma,[1,2,3] **Qinglei Zhou**,[1] **Mingsheng Hu**,[2] **and Xing Wang**[4]

[1]*School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China*
[2]*School of Information Science and Technology, Zhengzhou Normal University, Zhengzhou 450044, China*
[3]*School of Information Engineering, North China University of Water Resources and Electric Power, Zhengzhou 450046, China*
[4]*College of Electrical Engineering, Yuquan Campus, Zhejiang University, Hangzhou 310027, China*

Correspondence should be addressed to Wei Ma; wei.ma1222@gmail.com and Qinglei Zhou; ieqlzhou@zzu.edu.cn

Received 15 May 2021; Revised 15 June 2021; Accepted 24 June 2021; Published 2 July 2021

Academic Editor: Mamoun Alazab

Attacks launched from the inside of the cloud are threats not only to the cloud users but also to the cloud infrastructures. Although with trusted computing the cloud service providers can guarantee the trust and security of the cloud environment for the users, the trustworthiness of users is not properly assessed. Inspired by the concept of variable trust, the main contribution of this paper is that we propose a trust assessment method for cloud users based on deep learning. Firstly, we extract users' activities from system logs and employ stacked LSTM (long short-term memory) neural network to model normal activity patterns to build trust profiles for different users. Secondly, the trust profile is capable of predicting future behavioural actions of the specific user, and by calculating the similarity between predicted actions and actual actions the trustworthiness of the user will be assessed with a baseline to detect the trust state of the cloud user dynamically. And in the end, we design and conduct experiments on a public dataset. The results of experiments indicate that when the user is in abnormal state, there are notable differences between predicted actions and user's actual actions, which proves the efficiency of the proposed method.

1. Introduction

Security issue is one of the most important problems in cloud computing and building trust is a practical method to resolve it. Typically, trust is a mutual relationship in cloud environment. On one hand, cloud service providers (CSPs) usually need to provide trust evidence to cloud users to ensure that the security of computing environment is guaranteed. For example, TCG's trusted computing-based trust chain [1], trustworthiness attestation [2], and transparency of cloud platform [3] are all included in trust evidence. On the other hand, CSPs also need evidence to trust the cloud users while the users trust the CSP. However, the trustworthiness of cloud users is always neglected. Few CSPs would ask for trust evidence from users and which leads to inside attack performed by malicious inside attackers. Malicious cloud users may exploit cloud resources to launch attack against some other cloud users or other Internet services. In traditional way, malware such as viruses, trojans, and worms are used to perform attack activities by malicious insiders. And, Botnet and information theft are used more often nowadays. For example, malicious user may be able to steal data from a target virtual machine with side channel [4], and DDoS attack may be launched by a malicious tenant against some other tenant [5]. In 2009, Zeus Botnet was found in Amazon EC2 and, in 2011, SONY PSN was attacked by hackers using Amazon EC2 resources in the same way. A lot of work has discussed malicious insiders in different aspects [6–8].

Hence, it is important to distinguish and identify "untrusted" inside users for cloud environment. Typically, there are two types of malicious insiders [9]: the traitors and the masqueraders. The traitor means a legitimate user who has privileges to visit some specific resources but counters to security policy, while the masquerader means a user who does not have corresponding privileges to some specific resources but manages to access them. In cloud environment, traitors can be considered as benign users which are controlled by bot-ware to perform malicious activities and masqueraders can be considered as users who

act like normal users but intend to impersonate some other users. What those two kinds of malicious insiders have in common is acting departure from normal patterns. Hence, the key for CSPs to trust users is to profile normal patterns of users. It means that the CSPs should know what a normal user would do and when the behaviour pattern changes CSPs should be aware. The technique anomaly detection is used to address such problems. Benefiting from the technique of anomaly detection, multiple kinds of behaviours are used to profiling user for detecting insider attacks, such as command sequences [10], system call sequences [11], system logs [12], web requests [13], network observable actions [14], and so on. It has been proved that those behaviours are practical in detecting anomaly activities. However, there are two issues to apply anomaly detection in cloud environment. Firstly, behaviours mentioned above are hard to collect or illegal to collect. Secondly, anomaly detection methods are always adopted as a binary classifier that cannot reflect the changes of status of the users.

No doubt that trusted computing is a powerful tool to guarantee the security and trust of cloud environment, but the trust of cloud users cannot be assessed. The trust of users is not a simple binary problem with the labels of "trusted" or "untrusted". Note that the zero trust architecture [15] stated that a variable trust should be allocated to every entity in the network no matter the service provider or the service recipient; we believe that the trust of users should also be dynamic and real-time. Therefore, in this paper, combining the notions of variable trust and anomaly detection, we propose a method of building trust profiles to assess trust for cloud users based on system logs from virtual machines which are legitimate and easy to collect [16]. System logs, which record system events in some critical points, are excellent data source for monitoring system behaviours. Furthermore, system logs are generic for nearly all kinds of operating systems and with fixed format to analyze [12]. The contributions of this paper can be highlighted as follows:

A deep learning-based trust profile: the trust profile is capable of predicting future behavioural actions of the specific user, and by calculating the similarity between predicted actions and actual actions the trustworthiness of the user will be assessed with a baseline to detect the trust state of the cloud user dynamically

Convenient and legitimate data collection and processing: the data used to profile user's behavioural patterns is extracted from system logs which are easy and legitimate to collect for diagnosis

Experiments were conducted on an open dataset and the results indicated that our method is able to describe the real-time status of the user

This paper is organized as follows. Section 1 introduces the background and promotions with the related works in Section 2. The proposed assessment method is discussed in Section 3 and evaluated with experiments in Section 4. In the end, the conclusion is given in Section 6.

2. Related Work

TCG's trusted computing plays an important role in the study with respect to trust in cloud environment. Trusted computing is defined as follows: one entity is trusted if this entity acts toward prospective goal with prospective actions. Generally, there are several manners such as transitive trust, attestation, and sealing storage used to enhance security of the protected system in trusted computing. In cloud environment, trust relationships are established in many ways, e.g., reducing VMM function to gain trust [17], a trusted coordinator based TCCP (trusted cloud computing platform) [18], vTPM [19] and sHype [20] integrated TVDc [21], etc. In recent years, trusted computing is also adopted in cloud-edge systems to guarantee the efficiency, reliability, and resource allocation. However, most of the researches about trust in cloud focus on how to build trust in cloud environment, while the trust of users is always neglected.

For trust issue from the users' end, traditional security methods such as identification, authentication, and access control are widely employed, and traditional security countermeasures accompanied with novel techniques such as blockchain are also adopted to protect the security and privacy of cloud environments or keep cloud environment from certain attacks. Building trust for users is also an important approach. It was pointed out that the trust of users is still a hot topic in the study of cloud computing security [22]. In [23], a policy-based trust evaluation method was proposed by checking the policies violated by the user or not to tell the trustworthiness of the user, but the establishment of the policies highly depended on experience and expertise. Li et al [24] designed a scheme to determine the trust of user by evaluating the interactions between user and the network environment and user and other users, but the interaction data was hard to collect. In [25], a hierarchical trust assessment method for cloud users was proposed, in which a series of subtrust properties with respect to user behaviours were aggregated to a global trust, but the scheme was not authenticated with experiments. Fuzzy logic was adopted in [26] to compute a trust value for a user with evidence of user's behaviours such as login, security, operating, and performance. Chen et al. [27] proposed a trust evaluating scheme, with which user's behaviours such as authentication, downloads, uploads, and interactions were adopted to compute the direct/indirect trust, and the comprehensive trust was calculated with weighted sum. But both [26] and [27] only took advantages of limited user behaviours which are also hard to collect. To describe the participant in a computing environment, the notion of trust profile [28, 29] was introduced, which means the policy that a participant declares to the execution environment in order to have a trust based interaction and focuses on the entities in the environment instead of the environment itself. Different from trusted cloud computing, trust profile is applied as security constraint for both service suppliers and service users. A behaviour based trust profile for a specific user will be able to represent how this user would act when using the service.

To build trust profile, anomaly detection is a typical and powerful tool employed to analyze behaviours of a user. Anomaly detection is defined in [30] as a problem of finding patterns in data that do not conform to expected behaviour. To define a normal region that encompasses all normal activities, a series of methods have been adopted by researchers such as statistics, machine learning, and data mining. In the early time, anomaly detection was mainly based on statistics and pattern recognition [31, 32]. With the development of machine learning, new methods were proposed. In [33], Naive Bayes was applied to analyze anomalies in UNIX commands of multiple users. One-class support vector machine (SVM) was used to detect inside attackers in [34]. In [35], an eigen cooccurrence matrix based method was used to extract relationships in the commands' sequences. And, hidden Markov model was also employed to distinguish abnormal behaviours from normal behaviours [36]. Compared with schemes in [22–27], anomaly detection based trusted profile can utilize more features of users.

In recent years, with the rise of deep learning or deep neural networks, deep learning-based methods were proposed for detection [37, 38] or prediction [39]. A significant benefit of deep learning-based methods is that feature engineering in traditional machine learning is no longer indispensable. In [39], deep neural network was used to predict service stability of smart grid. In [37], a deep neural network-based method was proposed to detect inside threats with an LSTM-based automatic feature extraction and a CNN based classifier, but the sequential characteristics were only considered in feature extraction and it was still a classification problem in essence.

The related work is summarized in Table 1. In general, we have classified related work into four categories and each of them focuses on different problems. However, there are limitations for all of them. For example, for traditional security methods such as access control and authorization and TCG's trusted computing-based methods, the inside users are always considered trusted and therefore the user's trust is neglected. For nonlearning-based trust evaluation methods, a series of predefined trust properties are needed for the evaluation process and the properties are not comprehensive enough. For current learning-based methods, there are also limitations. First, the data used in those methods is neither hard to collect nor illegal to collect. Second, the algorithms adopted in those methods cannot capture some hidden features such as time dependence. Third, current learning-based methods always focus on binary classification for distinguishing abnormal or malicious targets but cannot describe the real-time trust status of an entity dynamically. To overcome those limitations, the advantages and merits of our methods would be focusing on user trust, making use of more convenient data source, and taking advantage of hidden features such as time-dependencies.

3. Methodology

In this section, we propose the trust profile for cloud users and trust assessment method. First, the application scenario of the proposed method is introduced. In a typical cloud environment illustrated in Figure 1, virtual machine instances, which represent the cloud users, are generating system logs in their life cycles all the time. The logs are collected via a central log server and used for further analysis.

There are four phases in the scheme, the selection of data source, the extraction of behaviours, building trust profile with LSTM, and trust assessment. In the first phase, we choose system logs as data source due to the accessibility and legality. In the second phase, users' behaviours are extracted from well-formatted system logs. What is worth mentioning is that the system logs selected in the first phase and behaviours extracted in the second phase should be derived from trusted source and hence would be a reflection of the normal behaviour patterns of users. In the third phase, trust profile will be built using LSTM. And, in the last phase, an algorithm is developed to predict the future actions of a user in terms of the trust profile and the trust assessment will be performed based on the similarity of predicted action sequence and real action sequence which actually happened. With the similarity value, the trust status of the user would be assessed. The four phases of our method are illustrated in Figure 2.

3.1. Phase 1: Data Source. The most often used data sources of anomaly detection are command lines and system call sequences. However, in cloud environment, it is either hard or illegal to collect command line history or system call sequences in client's virtual instances, no matter IaaS, PaaS, or SaaS. In contrast to command lines and system call sequences, system log is a good alternative data source. There are several benefits of system logs. Firstly, system logs contain a lot of information that indicates the behaviours of the virtual instances, which can be translated into behaviours of cloud users. Secondly, system logs are universally available in every operating system. Besides, which is important, system logs are legitimate to collect by CSPs due to diagnosis needs. And, meanwhile, it is easy to collect system logs from all virtual instances in a centralized manner for whether Microsoft Windows or Linux. Hence, in this case, we choose system logs as data source to build the trust profiles for cloud users.

3.2. Phase 2: Behaviour Extraction. The first step of building trust profile for cloud user is to extract behaviours from system logs. Typically, there are two parts of system log entry, the "Event ID" and the "Event Content." Event ID tells what incident happened and Event Content tells the detailed information of this incident. We use that information to extract behaviours from log data. For every cloud user, it would be easy to obtain its log files and to assemble them. Suppose that there are K users that need to establish trust profile, denoted as $U = \{u_1, u_2, \ldots, u_k\}$. Let L_i^m be log file of user i ($i \in [1, K]$) where m is the number of log entries of this log file. As illustrated in Figure 3, there are three steps to extract behaviours.

Table 1: Summary of related work.

Category		Literature	Focus
Tradition security methods		[22]	Specific security issues
TCG's trusted computing		[17–21]	Building trust for cloud infrastructures
User trust	Nonlearning-based methods	[23–27]	Evaluating user's trust or reputation
	Learning-based methods	[33–37]	Building trust profiles for users

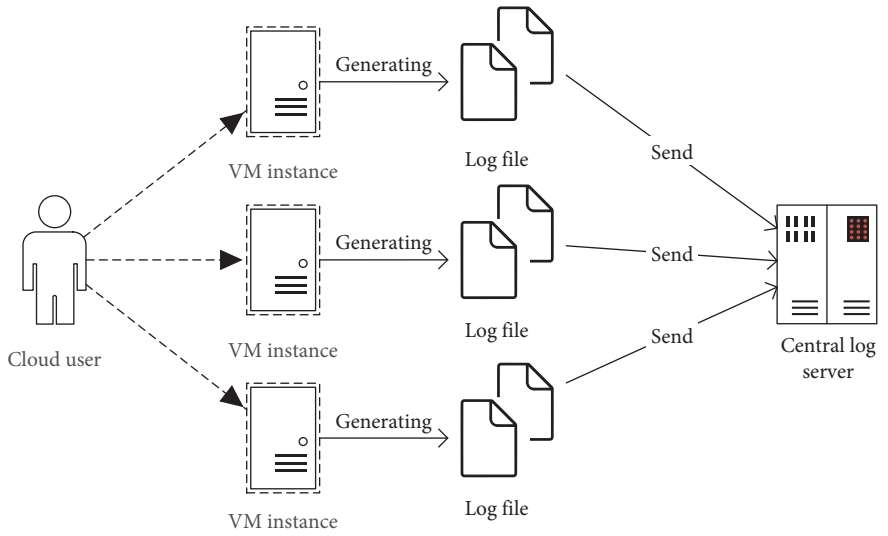

Figure 1: Cloud scenario for the proposed method.

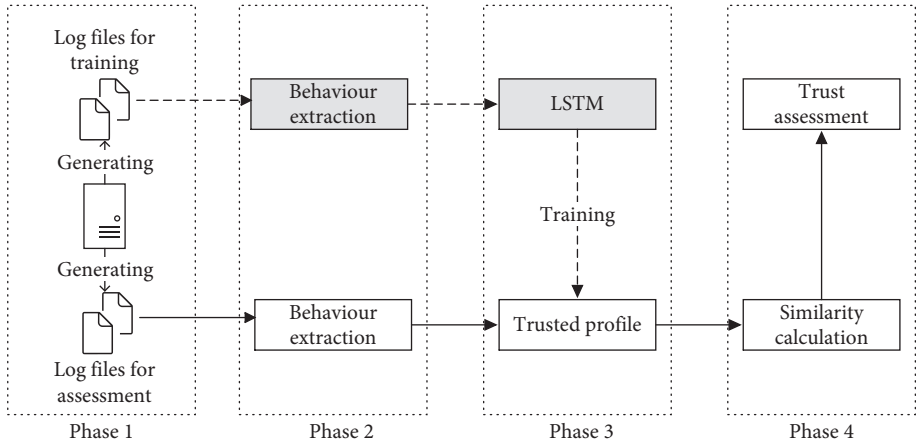

Figure 2: Phases of the proposed method.

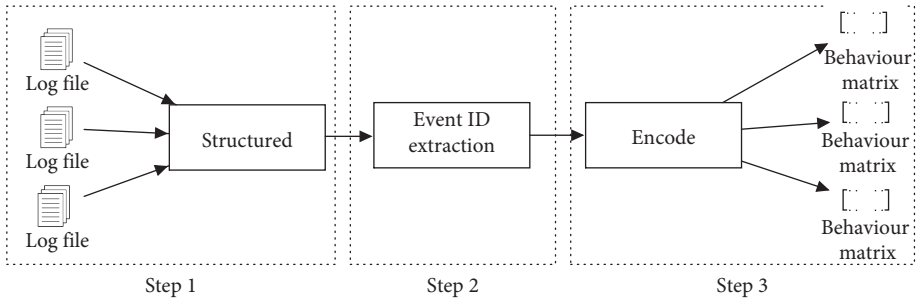

Figure 3: Process of behaviour extraction.

Step 1. The first step aims to transfer raw log data into structured data. Generally, raw log files are always disorganized and should be parsed before bringing into operation. In this paper, we use XML format to structure log files. We separate important information from raw log data such as Event Name, Event GUID, Event ID, corresponding process or thread id, opcode, timestamp, etc. The input of this step is L_i^m, and the output would be an XML file with m child nodes of root node, denoted as XML_i^m.

Step 2. The second step is about to extract necessary information from structured data. Due to the feature of multisource and heterogeneous of log data, we only use Event ID and timestamp to extract behaviours. And, the rest of the information is still stored for further research purposes. With XML format, it is easy to extract Event ID. Timestamp is also extracted to index corresponding Event ID. With XML_i^m as input of this step, a sequence of Event IDs with the length of m and sorted by time will be generated, denoted as $S_i^m = \{s_{i,1}, s_{i,2}, \ldots, s_{i,m}\}$ where $s_{i,t}\,(1 \le t \le m)$ is an event entry.

Step 3. The third step aims to encode Event ID sequence generated in step 2. In this paper, we choose one-hot encoding to finish this task. Because the amount of system events is limited, a finite-capacity vocabulary can be built. Suppose that the total number of system events is N; then let $V = \{v_1, v_2, \ldots, v_N\}$ be the vocabulary that contains all system events. When the sequence entry $s_{i,t}$ matches event v_j $(j \in (1, N))$, this entry will be a vector $s_{i,t} = [e_{i,t}^1, e_{i,t}^2, \ldots, e_{i,t}^N]$ where $e_{i,t}^j$ is set as 1 and the others are set as 0. Furthermore, the m-length sequence of Event IDs would be represented as a matrix $BehM_i = \begin{bmatrix} s_{i,1} \\ s_{i,2} \\ \cdots \\ s_{i,m} \end{bmatrix} = \begin{bmatrix} e_{i,1}^1 & \cdots & e_{i,1}^N \\ \vdots & \ddots & \vdots \\ e_{i,m}^1 & \cdots & e_{i,m}^N \end{bmatrix}$. This matrix will be the behaviour matrix of specific cloud user.

Every log file will be parsed to a one-hot encoded behaviour matrix after these three steps, which indicates that the behaviour pattern of cloud user is implied within this matrix. The behaviour matrix will be used as input of deep neural network.

3.3. Phase 3: Building Trust Profile with LSTM. With the behaviour matrix generated in above procedure, we can employ neural network to build trust profile. Not only the Event IDs but also the sequence of Event IDs indicates the pattern of users' behaviour. Therefore, to capture the sequential characteristic of users' behaviours, LSTM neural network is adopted in this paper.

LSTM takes a fixed-length sequence as input. In this paper, we use a w-length time window to split the Event IDs sequence into a series of subsequences with a stride of n. With one-hot encoding, the input of LSTM in this paper would be a $w \times N$ behaviour matrix. Suppose the LSTM network consists of one input layer, l LSTM layers, and one output layer, as shown in Figure 4. In Figure 4, $S_{i,1\ldots w}$, which indicates the input of the neural network, is the encoded Event IDs, and h^l is the hidden state in the output of a previous LSTM cell. The cells are connected with S_i and h^l to generate a w-length and l-layer network.

The output of LSTM shown in Figure 4 is calculated in the process of forward propagation, which is described as follows:

$$\begin{aligned} f_t^{(l)} &= \sigma\left(W_f^{(l)} \cdot \left[h_{t-1}^{(l)}, \mathrm{input}_{i,t}\right] + b_f^{(l)}\right), \\ i_t^{(l)} &= \sigma\left(W_i^{(l)} \cdot \left[h_{t-1}^{(l)}, \mathrm{input}_{i,t}\right] + b_i^{(l)}\right), \\ \widetilde{C_t^{(l)}} &= \tanh\left(W_C^{(l)} \cdot \left[h_{t-1}^{(l)}, \mathrm{input}_{i,t}\right] + b_C^{(l)}\right), \\ C_t^{(l)} &= f_t^{(l)} * C_{t-1}^{(l)} + i_t^{(l)} * \widetilde{C_t^{(l)}}, \\ o_t^{(l)} &= \sigma\left(W_o^{(l)}\left[h_{t-1}^{(l)} + b_o^{(l)}\right]\right), \\ h_t^{(l)} &= o_t^{(l)} * \tanh\left(C_t^{(l)}\right). \end{aligned} \tag{1}$$

$\sigma(\cdot)$ is sigmoid function, and $\mathrm{input}_{i,t}$ equals $s_{i,t}$ when $l = 1$. $W_f^{(l)}$, $b_f^{(l)}$, $W_i^{(l)}$, $W_C^{(l)}$, $b_i^{(l)}$, $b_C^{(l)}$, $W_o^{(l)}$, and $b_o^{(l)}$ are trainable parameters. The trust profile consists of those parameters along with some hyper parameters. An exclusive LSTM neural network would be built for every user in cloud environment as the user's trust profile.

3.4. Phase 4: Trust Assessment. Trained with normal behaviours of specific user, the trust profile of this user is generated and applied to predict future actions of this user. An algorithm is adopted to predict next x-step actions.With the predicted action sequence, the key to assess the user's trustworthiness is to compute the mean similarity of predicted sequence and actual sequence. Firstly, we calculate the similarity of individual action of the two sequences. As every action is encoded as a one-hot vector, we adopt cosine similarity for single action:

$$\mathrm{sim}_i = \left(\left(\overrightarrow{\mathrm{predicted}_i} \cdot \overrightarrow{\mathrm{real}_i}\right) \Big/ \left(\overrightarrow{\mathrm{predicted}_i} \times \overrightarrow{\mathrm{real}_i}\right)\right). \tag{2}$$

Then, the mean similarity is calculated as

$$\mathrm{Sim}\left(\mathrm{Seq}_p, \mathrm{Seq}_r\right) = \frac{1}{x} \sum_{i=1}^{x} \mathrm{sim}_i\left(\overrightarrow{\mathrm{predicted}_i}, \overrightarrow{\mathrm{real}_i}\right), \tag{3}$$

where Seq_p refers to the predicted action sequence by Algorithm 1 and Seq_r refers to the action sequence that really happened derived from the future system logs. And $\mathrm{predicted}_i$ and real_i refer to the vectorized single step action in Seq_p and Seq_r

Furthermore, a threshold θ is set to determine the trustworthiness of user u_i:

$$T_{u_i} = \begin{cases} 1, & \mathrm{Sim}\left(\mathrm{Seq}_p, \mathrm{Seq}_r\right) \ge \theta, \\ 0, & \mathrm{Sim}\left(\mathrm{Seq}_p, \mathrm{Seq}_r\right) < \theta, \end{cases} \tag{4}$$

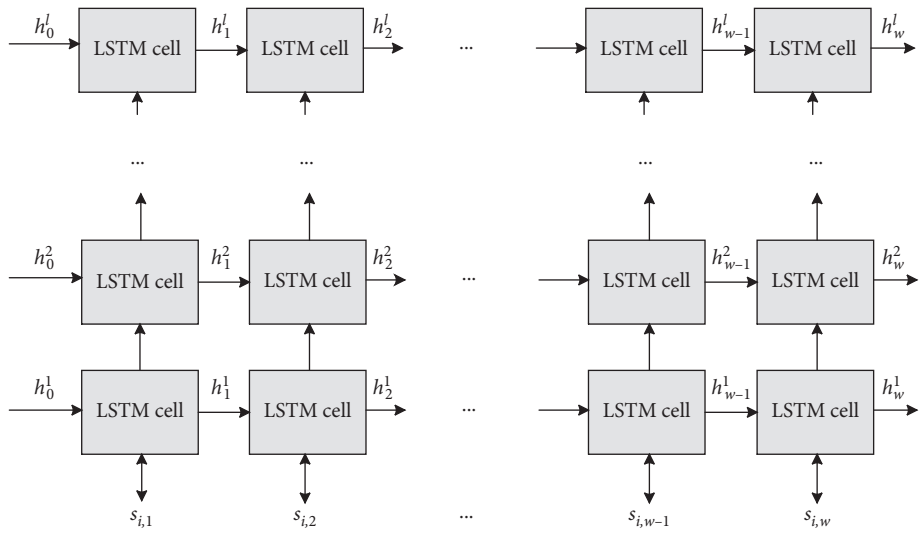

Figure 4: LSTM structure.

```
(i) Input: current action sequence to Predict, steps to predict x
    Output: predicted action sequence predicted
(1)   function PREDICTACTIONS (to predict)
(2)     nextAction = LSTM (to Predict)
(3)     return next Action
(4)   end function
(5)   iter = 0
(6)   while (TRUE) do
(7)     if iter ≤ x then
(8)       nextAction = PredictAction (to Predict)
(9)       predicted = to Predict.remove (the first element)
(10)      predicted = predicted.append (next Action)
(11)    else
(12)      return predicted
(13)    end if
(14)  end while
```

Algorithm 1: Predict sequence.

which means when user's activities deviate from the predicted activities which are conceived normal behaviours, the user would be considered in untrusted state.

4. Evaluation

4.1. Dataset Description. The dataset we perform experiments on is CSE-CIC-IDS2018 [40]. This dataset contains a series of attack scenarios such as DDoS, Brute force attack, Web attack, and Botnet, etc. We choose Botnet attack which fits the cloud inside risk for the experiment. The Botnet software was deployed on 10 Microsoft Windows servers and the system logs are collected from the 10 servers. The servers were kept running for 720 hours and in the last 3 hours the Botnet software was activated. Therefore, the dataset has recorded system logs of 720 hours, in which the first 717 hours were when the servers were in normal or benign state and in the last 3 hours the servers were in abnormal or malicious state. Hence, the dataset can be separated into two parts, the part in the normal state and the part of malicious state, which were connected but can be distinguished with timestamps in the logs.

4.2. Data Preprocessing and Experiment Setup. We extract behaviours from the logs with phases in Section 3.2, as shown in Figure 5.

We extract about 9,000 normal behaviours from the normal state and about 250 behaviours (such as '104', '7036', and '51047' in Figure 5) within the period of attack which are considered untrusted behaviours for every server. Those data are used to train the learning models to build trust profile for each server. In the training process, we only use normal behaviours as training data to determine the parameters of models and we use untrusted behaviours to test the trust profile. Corresponding parameters are chosen as in Table 2.

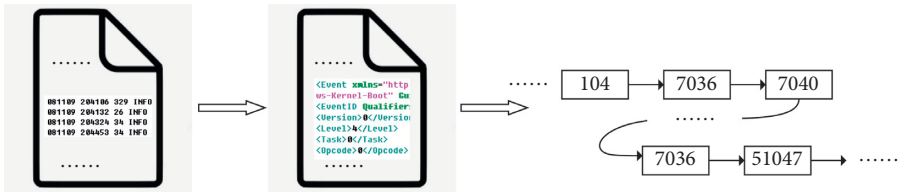

Figure 5: Behaviours extraction from the dataset, in which '104' and '7036' indicate the Event IDs contained in the log files.

Table 2: Parameters in experiment.

Parameter name	Parameter value
Time window	$w = 99$
Stride	$n = 1$
To predict steps	$x = 30$
Event numbers (collected from log files)	$N = 63$

4.3. Comparison with Previous Methods and Model Selection. We studied representative anomaly detection methods such as One-class SVM (OC-SVM) in [34] and a state-of-the-art deep learning-based method in [37] to compare the performance among the proposed method and related work. We briefly describe them as follows:

Method in [34]: one-class SVM model was employed in detecting masquerade intrusions. Non-intruded data was adopted as training data for the model, and the trained model was used to detecting "unusual" behaviours of a specific user. Only non-intruded data was collected and labelled as negative; when outlier behaviour was detected, it would be reported as positive.

Method in [37]: an LSTM-CNN (long short-term memory-convolutional neural network) was adopted to detected insider threat without feature engineering. The LSTM was used to extract features automatically. The extracted features would be transformed into matrices and the CNN was used to classify them as normal or abnormal.

Method in this paper: we take advantage of LSTM to model the normal behavioural patterns of cloud users as trusted profile. And we use regression method to predict the future behaviours of the users. The similarity between the "what-should-do" behaviours and "what-really-do" behaviours is calculated to assess the trustworthiness of the users.

Considering the dataset is a segment of system behaviours cut off from the complete behavioural workflows, we also investigated other LSTM-based models in our experiments, stacked LSTM and bidirectional LSTM. The stacked LSTM is a neural network with multiple hidden LSTM layers. As an expansion of LSTM, stacked LSTM is with greater model complexity and it can create a more complex feature representation. And the bidirectional LSTM (Bi-LSTM) takes the input data twice for training from two directions. Bi-LSTM is able to improve learning long-term dependencies and thus consequently will improve the accuracy of the model. In the experiments, the kernel function of one-class SVM was Radial Basis Function (RBF) and the γ was set to 0.5. And for LSTM-based models, we adopted a 3-layer stacked LSTM and a 2-layer Bi-LSTM. The 9,000 normal behaviours were dynamically generated into several mini-batches with the size of 100 to train the models for 50 epochs with RMSProp. The performance of model accuracy and training time was compared in our experiments as illustrated in Figure 6. For OC-SVM, traditional LSTM, Bi-LSTM, and 3-LSTM, the accuracy on training data is 58.1%, 94.7%, 80.1%, and 98.2%, respectively. With the growth of the scale of the parameters of the models, the training time of the four models is increasing as well, recorded as 369685 ms, 920128 ms, 950012 ms, and 110192 ms.

Due to the high-dimension and sequential characteristics of the input data, the LSTM-based models outperformed the OC-SVM in accuracy, in which 3-LSTM has achieved the optimal performance. Hence, we have chosen the stacked LSTM to model the trust profile.

4.4. Experiment Result. With the trained LSTM, we experiment on untrusted behaviours to evaluate the performance of proposed method. We choose the last 99 behaviours in normal state as current actions to predict next 30 actions and then compute the similarity value between each action in the predicted sequence and each action in the real sequence that happened in the attacking state. To provide a comparison, we also predict some sequences in the normal state. The results are illustrated in Figure 7.

In Figure 7, the x-axis is the action sequence number while the y-axis is similarity value between the predicted action and real action, and the horizontal line *similarit* $y = 1$ is the baseline that indicates that the predicted sequence is identical to the real sequence. When the predicted action is different from the real action, the corresponding point will deviate from the baseline. The broken line in Figure 7 indicates the similarity value between each predicted action and real action calculated with (7), and the mean similarity is also calculated with (8) and shown on top of every subfigure. The first 3 subfigures show that the mean similarity is close to 1 when predicting sequences in normal state, while the last subfigure shows that the similarity is far away from the baseline when the server enters Botnet state. It can be seen intuitively that the broken line in normal state (the first 3 subfigures) and in Botnet state (the last subfigure) are significantly different, which verified that the similarity value is useful to describe the state of target server.

Also, an experiment was performed on the effectiveness of trust profile. We picked up three subsequences from the

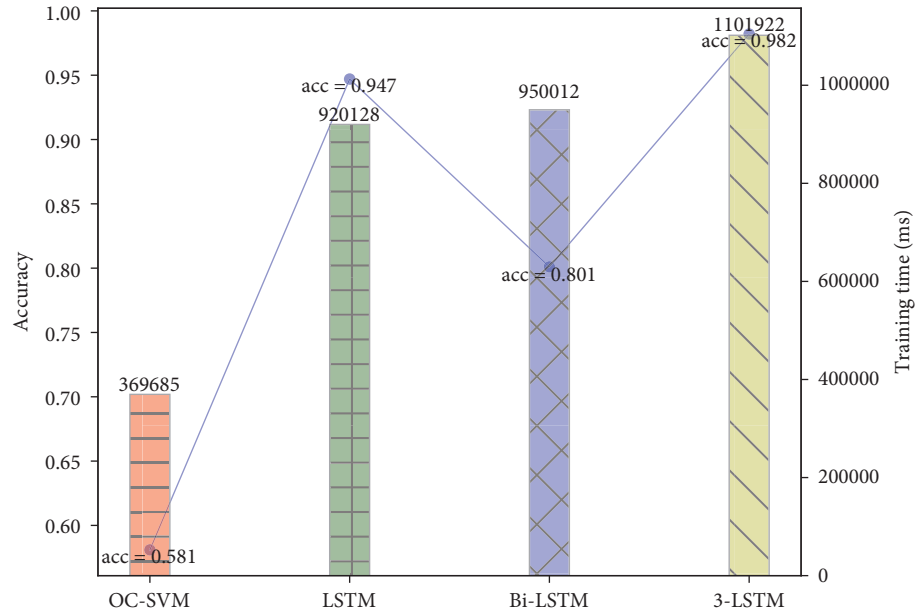

Figure 6: Performance comparisons between different models.

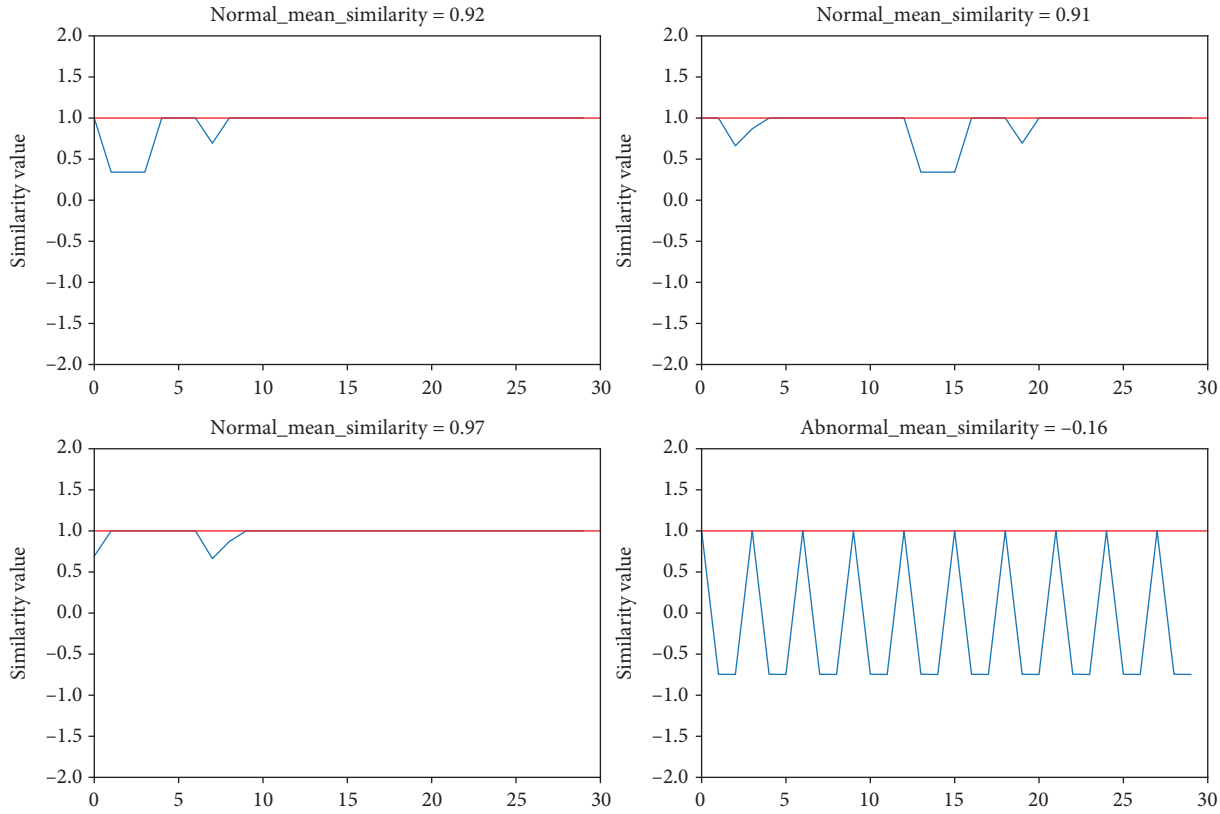

Figure 7: Similarity of predicted sequences and real sequences.

dataset. Those subsequences were extracted from system logs of three different users, i.e., three different cloud hosts. All three subsequences contain normal behaviours and malicious behaviours but those behaviours are distributed in different time period. The structures of the three subsequences are shown in Table 3.

For convenience, the number of behaviours in different periods is modified based on the parameters in Table 2. The

Table 3: Subsequence structure.

	Period 1	Period 2	Period 3
Subsequence 1	Normal	Malicious	Normal
Subsequence 2	Malicious	Normal	Normal
Subsequence 3	Malicious	Normal	Malicious

similarity values on three subsequences are calculated to check whether the trust profile will be able to determine the trust state or not. The result of this experiment is shown as Figure 8.

In Figure 8, the x-axis is the number of behaviours sequence and the y-axis is the similarity value between every predicted action and real action. When the user is in normal state, the similarity value would oscillate around 1 while it would oscillate around 0 when entering malicious/untrusted state. Figure 8 illustrates that when user enters different state, no matter from normal state to malicious state or from malicious state to normal state, it will be reflected by the similarity calculated with trust profile. Therefore, trust profile is useful to detect malicious or abnormal state for cloud users, and the CSP would be able to take response measures such as sending alert, adding to black or blocking at network level.

4.5. Computation of Trustworthy Threshold θ. It is important to determine the trustworthy threshold θ. The value of θ is highly associated with the user's usage, business, active time, etc. Consequently, θ should be established for every single user and determined by the historical experience of specific user. In our experiment, the threshold is determined with

$$\theta^{(u)} = \frac{1}{k}\sum_{i=1}^{k}\theta_i^{(u)}, \tag{5}$$

$\theta^{(u)}$ is the trustworthy threshold for specific user$_u$, and it is calculated as mean value of multiple times of single-time threshold $\theta_i^{(u)}$. $\theta_i^{(u)}$ is calculated with two steps. Firstly, we randomly choose one action from the action sequence extracted from the trusted system logs and it would be the starting point of the time window. Secondly, the similarity will be calculated with the parameters selected in Table 2 and the value of similarity will be $\theta_i^{(u)}$ in the ith time, which means $\theta_i^{(u)} = \mathrm{Sim}_i$. Obviously, the bigger k is, the more accurate $\theta^{(u)}$ would be.

In the experiment, we performed the computation of trustworthy threshold θ for one user. The value of k is set to 10,000. The mean value of those 10,000 similarity values would be θ and the value computed is 0.8199. Hence, in this instance, the trustworthy threshold θ would be set as about 0.8.

5. Discussion

5.1. Qualitative Analysis. The result of the experiments indicated the effectiveness of the proposed method. We also quantitatively compared our method with similar schemes for building trust in cloud environment and summarized in Table 4.

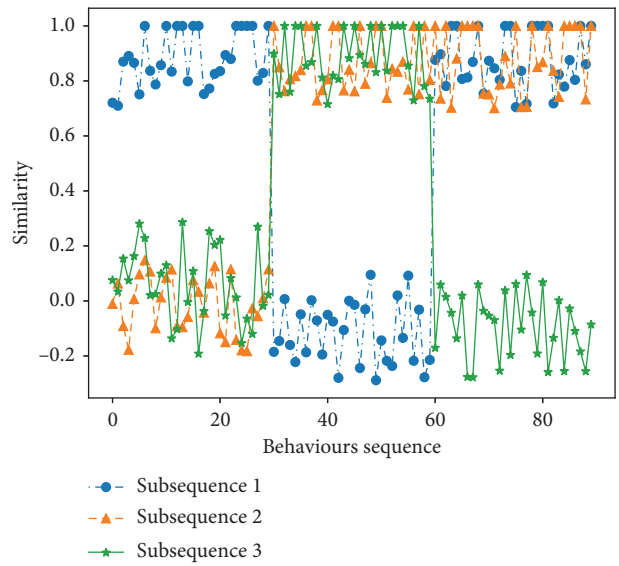

Figure 8: Variation of similarity curve in different scenarios.

We compared the trusted computing-based methods and learning-based methods from 5 aspects. Compared with trusted computing-based methods, our method can build trust for cloud users without requirements of special hardware. Compared with other learning-based schemes such as [34] and [37], our method benefits from the behavioural actions of cloud users extracted from system logs with LSTM to build a trust profile rather than feature extraction, which utilizes the sequential characteristics more comprehensively. And furthermore, our method focuses on the real-time trust assessment instead of binary classification.

5.2. Time Consumption and Complexity Analysis. Note that the time consumption in the training phase is a factor to be reckoned with. As shown in Figure 6, all LSTM-based methods would take more than 900000 milliseconds to be trained. However, with our method, the training phase and trust assessment phase are separated; i.e., we train the trust profile for every cloud user offline and assess the trustworthiness online. The time consumption in the training phase would not affect the efficiency of the assessment phase. The overhead in the assessment phase is considered with respect to the complexity of Algorithm 1. In Algorithm 1, a sequence of x steps of actions is generated with a loop from line 6 to line 14. The time complexity of algorithm 1 is $O(n)$ and the space complexity of algorithm is $O(n)$ as well, depending on the value of x. In our experiment, the time used in the assessment phase is about 5 milliseconds, which is totally acceptable.

And meanwhile, due to the fact that the training data may not cover all the patterns of normal behaviours, a dynamical update mechanism is adopted. With this mechanism, the CSP can provide feedback by checking the result of the trust assessment. When the result is a false positive, the CSP would take the input as training data for incremental training to update the parameters of the LSTM

Table 4: Comparisons of different methods.

		Special hardware required	Trust of users	Behavioural sensitiveness	Behavioural comprehensiveness	Technical features
Trusted computing-based methods		Yes	No	No	No	Transitive trust
Learning-based methods	Other schemes [34, 37]	No	Yes	Yes	No	Binary classification
	Our method	No	Yes	Yes	Yes	Real-time trust assessment

neural network. The incremental training process is also offline and the model would be updated continuously to improve the accuracy of trust assessment. In the case of the trustworthy threshold θ, the situation is similar. The value of θ is determined with multiple times of calculating similarities. Hence, at the beginning of trust assessment phase, the value θ is set manually while it can be updated dynamically in the follow-up trust assessments.

6. Conclusion

In this paper, we proposed a deep learning based trust assessment method for cloud users. With system logs collected in cloud environment, we extract user behaviours and build LSTM neural network as trust profile for this user. The trust profile is to determine the trustworthiness of the user by predicting the future actions and then computing the similarity between predicted actions and actual actions. With the calculated similarity and a trustworthy threshold, our method can assess the user's real-time trust status dynamically.

7. Case Study

The experiments verified the validity of our method. The trained trusted profile can describe the behavioural pattern of a cloud user and can be used as a baseline for detecting anomaly actions. However, regardless of the advantages, there are also limitations to our method. First, in the experiment we extract only 250 behaviours from the dataset, and the number is not so big for global system activities. Hence, our method is more suitable for building trust profile for cloud users that are with higher stability, as the behavioural actions or Event IDs are only sufficient to describe cloud users which undertake monotonous tasks. Second, the trustworthy threshold θ is not easy to determine. In Section 4, we have explained the computation process of θ. However, in the process, the final value of θ is highly dependent on the training data; i.e., the more training data is, the more accurate θ is. The scale of training data determines the accuracy of θ, as well as the performance of our method.

8. Future Work

Considering the limitations of our method, the future work is summarized. Firstly, it is necessary to find more generic properties to profile the users, and secondly, we should incorporate other types of neural networks to achieve better performance with smaller data scale. And thirdly, adopting the method to other fields such as Internet of things is also in consideration for future work.

Data Availability

The datasets used and/or analyzed during the current study are available on the website https://www.unb.ca/cic/datasets/ids-2018.html.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This paper was supported by the Henan Programs for Science and Technology Development (212102210100) and Henan Key Research Projects of Universities (20A413008, 21B520022, 20B520040, and 20A520043).

References

[1] Tcg. http://www.trustedcomputinggroup.org.

[2] W. Ma, Y. Liu, and C. Lv, "A novel service oriented approach of trustworthiness attestation in cloud computing," *International Conference on Machine Learning and Cybernetics*, vol. 4, pp. 1567–1671, 2013.

[3] W. Ma, Z. Han, and Y. Cheng, "Research on multi-level management mechanism in trusted cloud computing," *Netinfo Security*, vol. 7, pp. 20–25, 2015.

[4] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud," in *Proceedings of the 16th ACM conference on Computer and communications security*, pp. 199–212, ACM, Chicago, IL, USA, November 2009.

[5] H. Liu, "A new form of DOS attack in a cloud and its avoidance mechanism," in *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*, pp. 65–76, ACM, Chicago, IL, USA, October 2010.

[6] G. Gavai, "Supervised and unsupervised methods to detect insider threat from enterprise social and online activity data," in *Proceedings of the 7th ACM CCS international workshop on managing insider security threats*, pp. 47–63, Denver, CO, USA, October 2015.

[7] A. Tuor, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," in *Proceedings of the Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, San Francisco, CA, USA, February 2017.

[8] T. A. Tang, "Deep learning approach for network intrusion detection in software defined networking," in *Proceedings of the International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pp. 258–263, IEEE, Fez, Morocco, October 2016.

[9] M. B. Salem, S. Hershkop, and S. J. Stolfo, "A survey of insider attack detection research," *Insider Attack and Cyber Security*, pp. 69–90, Springer, Berlin, Germany, 2008.

[10] M. Schonlau, "Computer intrusion: detecting masquerades," *Statistical Science*, vol. 16, no. 1, pp. 58–74, 2001.

[11] N. Nguyen, P. Reiher, and G. H. Kuenning, "Detecting insider threats by monitoring system call activity," in *Proceedings of the IEEE Systems, Man and Cybernetics SocietyInformation Assurance Workshop*, pp. 45–52, IEEE, West Point, NY, USA, June 2003.

[12] M. Du, "Deeplog: anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1285–1298, ACM, Dallas, TX, USA, October 2017.

[13] H. Kim, "Use of support vector machine (svm) in detecting anomalous web usage patterns," *Symposium on Information and Communications Technology*, 2004.

[14] M. A. Maloof and G. D. Stephens, "Elicit: a system for detecting insiders who violate need-to-know," *International Workshop on Recent Advances in Intrusion Detection*, pp. 146–166, Springer, Berlin, Germany, 2007.

[15] S. Rose, *Zero Trust Architecture*, National Institute of Standards and Technology, Gaithersburg, MD, USA, No. NIST Special Publication (SP) 800-207 (Draft), 2019.

[16] H. Singh, "Using CloudWatch with SageMaker," *Practical Machine Learning with AWS*, Apress, Berkeley, CA, USA, pp. 155–165, 2021.

[17] D. G. Murray, G. Milos, and S. Hand, "Improving Xen security through disaggregation," in *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pp. 151–160, ACM, Seattle WA USA, March 2008.

[18] N. Santos, K. P. Gummadi, and R. Rodrigues, "Towards trusted cloud computing," in *Proceedings of the 2009 conference on Hot topics in cloud computing*, Berkeley, CA, USA, June 2009.

[19] R. Perez, R. Sailer, and L. Van Doorn, "VTPM: virtualizing the trusted platform module," in *Proceedings of the 15th Conf. on USENIX Security Symposium*, pp. 305–320, Vancouver, Canada, July 2006.

[20] R. Sailer, "sHype: secure hypervisor approach to trusted virtualized systems," vol. 5, 2005 Technical. Report. RC23511.

[21] S. Berger, R. Cáceres, D. Pendarakis et al., "TVDc," *ACM SIGOPS - Operating Systems Review*, vol. 42, no. 1, pp. 40–47, 2008.

[22] S. Basu, "Cloud computing security challenges & solutions-A survey," in *Proceedings of the IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 347–356, IEEE, Las Vegas, NV, USA, January 2018.

[23] Y. Bendale and S. Shah, "User level trust evaluation in cloud computing," *International Journal of Computer Application*, vol. 69, no. 24, pp. 31–35, 2013.

[24] Li Wen, "Trust model of users' behavior in trustworthy internet,"vol. 1, pp. 403–406, in *Proceedings of the WASE International Conference on Information Engineering*, vol. 1, pp. 403–406, IEEE, Taiyuan, China, July 2009.

[25] T. Li-qin, C. Lin, and Y. Ni, "Evaluation of user behavior trust in cloud computing," in *Proceedings of the International Conference on Computer Application and System Modeling (ICCASM 2010)*, vol. 7, pp. 567–572, Taiyuan, China, October 2010.

[26] M. Alruwaythi and K. E. Nygard, "Fuzzy logic approach based on user behavior trust in cloud security," in *Proceedings of the IEEE International Conference on Electro Information Technology (EIT)*, pp. 1–6, IEEE, Brookings, SD, USA, May 2019.

[27] Z. Chen, L. Tian, and C. Lin, "Trust evaluation model of cloud user based on behavior data," *International Journal of Distributed Sensor Networks*, vol. 14, no. 5, pp. 1–10, 2018.

[28] S. Galizia, A. Gugliotta, and J. Domingue, "A trust based methodology for web service selection," in *Proceedings of the International conference on semantic computing (ICSC 2007)*, pp. 193–200, IEEE, Irvine, CA, USA, September 2007.

[29] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision Support Systems*, vol. 43, no. 2, pp. 618–644, 2007.

[30] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[31] B. D. Davison and H. Hirsh, "Predicting sequences of user actions," in *Proceedingds of the Workshop on Predicting the Future: AI Approaches to Time-Series Analysis*, pp. 5–12, Menlo Park, CA, USA, July 1998.

[32] T. Lane and C. E. Brodley, "Sequence matching and learning in anomaly detection for computer security," in *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, pp. 43–49, 1997.

[33] R. A. Maxion and T. N. Townsend, "Masquerade detection using truncated command lines," in *Proceedings International Conference on Dependable Systems and Networks*, pp. 219–228, IEEE, Washington, DC, USA, June 2002.

[34] B. K. Szymanski and Y. Zhang, "Recursive data mining for masquerade detection and author identification," in *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop*, pp. 424–431, IEEE, West Point, NY, USA, June 2004.

[35] M. Oka, Y. Oyama, and K. Kato, *Eigen Co-occurrence Matrix Method for Masquerade Detection*, Publications of the Japan Society for Software Science and Technology, Japan, 2004.

[36] T. Rashid, I. Agrafiotis, and R. C. Jason, "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*, pp. 47–56, ACM, Vienna Austria, October 2016.

[37] F. Yuan, "Insider threat detection with deep neural network," in *Proceedings of the International Conference on Computational Science*, pp. 43–54, Springer, Cham, Berlin Germany, June 2018.

[38] M. Numan, "A systematic review on clone node detection in static wireless sensor networks," *IEEE Access*, vol. 8, Article ID 65450, 2020.

[39] M. Alazab, S. Khan, S. S. R. Krishnan, Q. V. Pham, M. P. K. Reddy, and T. R. Gadekallu, "A multidirectional LSTM model for predicting the stability of a smart grid," *IEEE Access*, vol. 8, Article ID 85454, 2020.

[40] Canadian Institute for Cybersecurity, https://registry.opendata.aws/cse-cic-ids2018DataSet:CSE-CIC-IDS2018, 2018.