Computer Systems Science & Engineering
DOI: 10.32604/csse.2023.026526
Article

Tech Science Press

Cyberattack Detection Framework Using Machine Learning and User Behavior Analytics

Abdullah Alshehri[1,*], Nayeem Khan[1], Ali Alowayr[1] and Mohammed Yahya Alghamdi[2]

[1]Information Technology Department, Faculty of Computer Science and Information Technology, Al Baha University, Al Baha, 65799, Saudi Arabia
[2]Computer Science Department, Faculty of Science and Arts at Buljurashi, Al Baha University, Al Baha, 65799, Saudi Arabia
*Corresponding Author: Abdullah Alshehri. Email: aashehri@bu.edu.sa
Received: 29 December 2021; Accepted: 27 February 2022

Abstract: This paper proposes a novel framework to detect cyber-attacks using Machine Learning coupled with User Behavior Analytics. The framework models the user behavior as sequences of events representing the user activities at such a network. The represented sequences are then fitted into a recurrent neural network model to extract features that draw distinctive behavior for individual users. Thus, the model can recognize frequencies of regular behavior to profile the user manner in the network. The subsequent procedure is that the recurrent neural network would detect abnormal behavior by classifying unknown behavior to either regular or irregular behavior. The importance of the proposed framework is due to the increase of cyber-attacks especially when the attack is triggered from such sources inside the network. Typically detecting inside attacks are much more challenging in that the security protocols can barely recognize attacks from trustful resources at the network, including users. Therefore, the user behavior can be extracted and ultimately learned to recognize insightful patterns in which the regular patterns reflect a normal network workflow. In contrast, the irregular patterns can trigger an alert for a potential cyber-attack. The framework has been fully described where the evaluation metrics have also been introduced. The experimental results show that the approach performed better compared to other approaches and AUC 0.97 was achieved using RNN-LSTM 1. The paper has been concluded with providing the potential directions for future improvements.

Keywords: Cybersecurity; deep learning; machine learning; user behavior analytics

1 Introduction

Cyber-attacks have become a major threat for network telecommunications due to rapid developments and growth in IT technology. The majority of cyber-attacks are carried out via breaking network security using malware that aims to compromise network security [1]. Malware assaults often compromise a secure network by introducing a harmful external component; thus, the attack originates from outside the network's perimeter security. Examples of malware attack tools include trojan horses, viruses, and worms [2]. Recall that the security breach in this context has a harmful effect on the victim machine, such as

altering and disrupting data, phishing sensitive transmitted data, or even flooding the network resources with an extraordinarily traffic to deny access to the network services known as denial of service (DoS) attack. Nevertheless, the common characteristic of malware attacks is that the potential attack is conceived from such an external source; ideally, it can be described as an external attack. As a result, an appropriate approach for securing the network is to propose a perimeter defense in the form of firewalls, antivirus software, and intrusion detection tools to prevent such a dangerous external component from violating security regulations.

Cyber-attacks, on the other hand, can be launched from an internal network resource which are known as insider threats [3]; also, it can be described as an insider attack. The most common type of insider attack is when the legitimate user engages in cyber-attacks. To put it another way, a valid user is well-versed in security standards, allowing them to gain access to network resources and quickly commit cybercrime. Moreover, the legitimate user may relatively utilize the granted credentials to perform the attack. Another form of insider attack is when the attack is occurred accidentally by the legitimate user. For example, if the user clicks on a spam link, the network can adversely be infected with malicious malware. Nevertheless, the insider attack is particularly deceiving since perimeter defenses have a hard time detecting attacks launched from within the network by actual users. Therefore, the insider attacks constitute a significant challenge in cyber security, making it desirable to propose powerful methods for detecting insider attacks.

Analyzing user activity is ideal for detecting insider threats [4]. Real users are mainly a key component of the network. They typically perform tremendous of tasks and activities on a daily basis [5]. This, in turn, composes frequent patterns of regular usage of variant tasks on the network. Hence, the regular activities and workflow tasks can underline an insightful pattern to map a distinguish user behavior. Intuitively, when the real user behavior is recognized, the irregular and abnormal behavior can be distinguished accordingly. To this end, the Machine Learning (ML) approach is most suitable candidate to detect cyber-attacks based on analyzing user behavior [6]. In concise, the conjecture is to learn the user's typical behavior, which is derived by the enormous daily tasks and activities. Thus, the system can effectively detect abnormal behaviors whereas the security protocols can render an alert of a potential attack.

This study proposes a framework for detecting cyber-attacks using coupled with user behavior analytics (UBA). The proposed framework has been motivated by the idea of applying deep learning methodology, a well-known disciplinary of ML, to recognize user behavior. More specifically, the framework proposes two models as follows. The first model is to represent the user activities as sequences of ordered event. These sequences are fitted into a deep learning model to extract features to map the regular behavior of user. The second model operates for detecting the potential attack by classifying the recognized behavior to either normal or abnormal behavior. Thus, the anticipated cyber-attack can be detected accordingly.

The contributions of the proposed work can be summarized as follows. (i) This paper utilizes user activities to map distinctive behavior patterns to detect cyber-attacks that are mostly triggered from inside the network. (ii) The study presents a novel method to model the user behavior in terms of handling the streams of activities. More specifically, the user behavior is represented as a sequence of events to map the user activities. The intuition is that event sequence frequencies would present accurate, readable patterns for the user behavior. (iii) Finally, the proposed framework has applied deep learning to extract features from the event streams representation. The importance is that user activities are stochastic and randomly featured, so deep learning is ideal for learning from unstructured data. This is a situation in which ideal features must be extracted from user activities to handle measured behavior.

The remainder of this paper is structured as follows. Section 2 presents a brief review of the related work in the literature. In Section 3, the proposed framework has been explained in detail. Next, in Section 4, the evaluation metrics have been introduced. Finally, Section 5 presents conclusions as well as future directions for the work.

2 Related Work

Amid the last decade, ML has been progressively being used to detect Cybersecurity-related attacks in response to the expanding variety and advancement of modern attacks. ML has proved to be advantageous in detecting and classifying malware attacks by providing exceptional flexibility in the automated detection of attacks [7–12]. Several studies have been conducted to detect Cybersecurity-related attacks using ML. A study by [13], primarily investigated the possibility of exploring security attacks using ML. Their breakthrough was limited up to the detection of malicious code attacks in webpages. The results obtained from this study shows that ML can be used for the detection of attacks with high detection of accuracy than signature-based methods and heuristics-based methods. A study by [14] used the 4-gram for feature extraction along with information gain entropy. The investigators use SVM, Naïve Bayes and Decision Trees classifiers to perform the classification of the malicious code. The results obtained show that the Area Under Curve (AUC) of 0.996 was achieved by using the J48 Decision Tree classifier. An approach for detecting malicious code attacks using API call sequence was proposed and implemented by [15]. Their study uses a small dataset for the classification of a fake medical website using K-NN ML Classifiers. A study by [16] implemented an approach for the detection of malicious JavaScript code attacks in webpages. Many studies have also been conducted for the detection of malware in Android mobile phones using Machine Learning with high accuracy [17–20].

UBA has been described by Gartner [21], as the subset and process for the detection of Cybersecurity threats, attacks, and monetary frauds. UBA works on the methods of analyzing the behavior of humans using statistical analysis and algorithms for the detection of Cybersecurity attacks. Several approaches have been proposed to counter Cybersecurity attacks [22–24]. The limitations with approaches are that they are limited to a specific group of people whose behavior is being analyzed for the detection of Cybersecurity attacks.

Keeping in view the advantages of using ML coupled with UBA towards detecting Cybersecurity attacks. We propose a framework based on ML and UBA by analyzing the behavior of common users for attack detection, which is the main contribution of this work.

3 Proposed Framework

This section introduces a throughout explanation of the proposed detection framework. Fig. 1 depicts an illustration of the proposed framework. Initially, the framework consists of two models as follows: (i) User Behavior Modelling and (ii) Attack Detection Modelling. The first model is concerned with representing the user activities as streams in which features are extracted to construct the user profile. The second model is concerned with training the user profile using the recurrent neural network (RNN) model to classify the user activities to either normal or abnormal behavior. Thus, the possible attack can be captured. The following subsections provide further details of each model as follows.

3.1 User Behavior Model

The first step of this model is to handle user behavior profiling. The objective is to collect the various data events of different types related to the user's activities. User behavior profiling searches for such pattern usage, which will imply suspicious behavior irrespective of whether such events emanate from a cybercriminal, an insider, or malware. User behavior analysis does not thwart the attack but helps identify and evaluate any potential attack through user activities. The data collected from user behavior can be obtained from the current as well as past user activities. The collected data points may include different values, i.e., the user's ID and IP address. It is worth mentioning that the majority of work in the literature has handled aggregated vectors of set of features. However, in this study, we have considered the entire characteristics of the audit data. Thus, the obtained data is structured as a series of events in which each event composed one or many data. The aim is then to build a distinctive user profile based on auto

extraction of features. In this study, we first train the model to extract features and then to recognize normal patterns behavior of each user in the network.

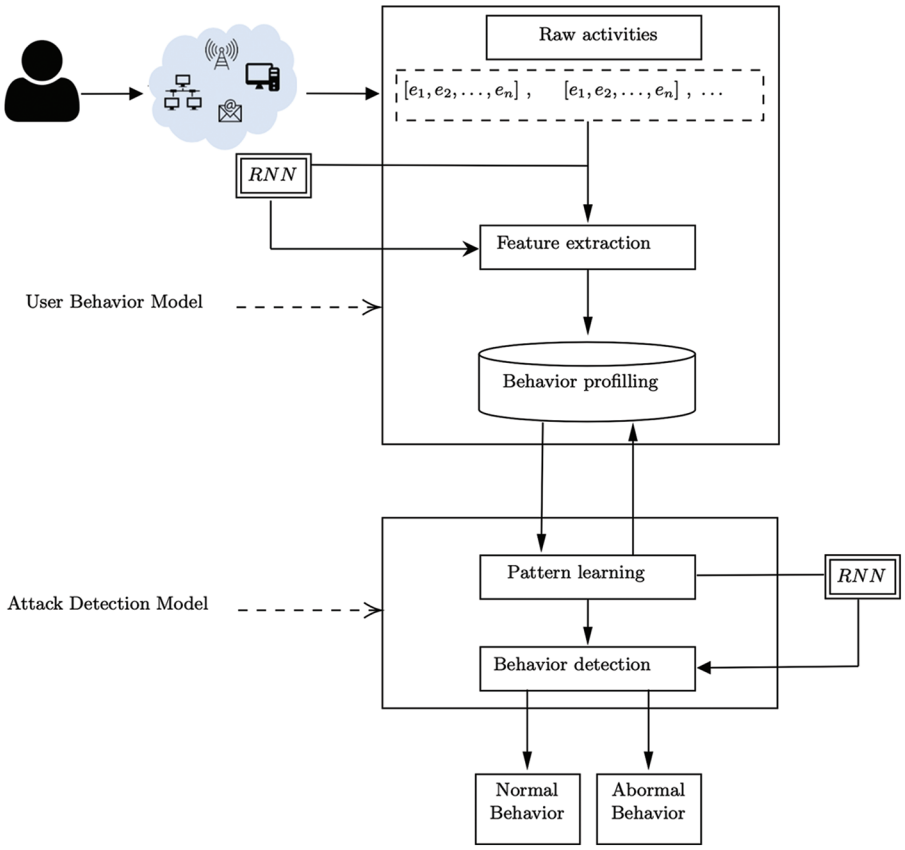

Figure 1: The basic models of the proposed framework

3.1.1 Feature Representation

The features are extracted from a sequence of activities for each user. Note that each sequence is indexed temporally by the time in which the sequence is acquired.

Given a series of user activity S, it is represented as a stream of events such that $S = \{e_1, e_2, \ldots, e_n\}$ where $n \in \mathbb{N}$ is the length of the stream. For an event e_i is a set of data points such that $e_i = [p_1, p_2, \ldots, p_l]$ where $p_j \in \mathbb{R}^n$ denotes a vector of input series. Thus, the feature can be extracted from the data stream to handle abstracted forms of features.

Prior to feature extraction, it is important to proceed with data normalization for better feature extraction at RNN [25]. The reason is that the outputs of the activation function reach a saturation point, after which they remain constant. As a result, when utilizing RNN cells, it is necessary to ensure that the inputs are properly normalized so that the outputs do not fall into the saturated range. Thus, the next step is to normalize the instances of the numerical data to a range from 0 to 1. To this end, we use the common min-max scaling normalization formula. Thus, given a vector p_j, which represents a series of certain data points entry, the vector is scaled to $p' \in [0, 1]$ as given in Eq. (1):

$$p' = \frac{p_j - \min(p_j)}{\max(p_j) - \min(p_j)} \tag{1}$$

The activity sequence is sent to a RNN as a vector for extracting features from the raw data, so the idea is to build an abstracted feature space. Recall that RNNs are universal approximators similar to artificial neural networks (ANNs); however, RNN has the advantage where the feedback loops of recurrent cells handle the temporal order as well as the temporal dependencies of the sequences from the start. To satisfy optimal feature extraction for the sequential structure of the obtained data, we employ long short-term memory (LSTM) to map an ideal feature space. Thus, LSTM has been employed to obtain abstracted feature vector using the deep layer from the raw data.

The LSTM receives inputs as a series of activities $[S_1, S_2, \ldots, S_N]$ for each user. Thus, given a user series S_i^u, the layer input is a vector of normalized features p' which represents a stream of user's activity. Afterward, the model maps a set of hidden state vectors $[h_1^{u,p}, h_2^{u,p}, \ldots, h_j^{u,p}]$ to show the probable association between each input vector. Recall that LSTM has memory to build hidden state vectors based on prior inputs, unlike standard deep-learning neural networks, which only find the computed hidden state vector as a function of the current input sequence. The hidden state function is triggered using three vectors as follows: remember vector r, save vector s, focus vector f. As a result, a long-term memory cell c_t will be formed after each hidden layer, which will be merged with the hidden state vectors, to remember the influence of earlier inputs. Formally, the Eqs. (2)–(6) show the mathematical notation of each gate vector with the generated cell in the hidden layers as follows:

$$r_t^u = \alpha\left(W_{(r,p)}[h_{t-1}, p_t] + b_r\right) \tag{2}$$

$$s_t^u = \alpha\left(W_{(s,p)}[h_{t-1}, p_t] + b_s\right) \tag{3}$$

$$f_t^u = \alpha\left(W_{(f,p)}[h_{t-1}, p_t] + b_f\right) \tag{4}$$

$$c_t^u = r_t \odot c_{t-1} + s_t \odot \alpha_c(W_c[p_t] + U_c[h_{t-1}] + b_c \tag{5}$$

$$h_t^u = f_t \odot \alpha_h c_t) \tag{6}$$

where $\alpha(.)$ is the activation function, in which in this case is the sigmoid function, and $\odot$ indicated the element wise multiplication.

3.1.2 Pattern Learning

The further step is to learn the user's behavior whenever the features are extracted, as explained in the previous stage as shown in Fig. 2. Thus, the extracted features are fed into LSTM for the training process. Recall that LSTM has the advantage that it can grasp sequential dependencies, so it has been well employed for temporal data analysis. This is especially the case of the user's behavioral analytics, where the pattern in the abstracted features can be recognized for distinctive behavior. The input of the LSTM is a vector of abstracted features for each user, such that $\left[s'_{u,1}, s'_{u,2}, \ldots, S'_{u,N}\right]$. The input vector is converted to a one-hot vector where each transformed vector is fitted to the LSTM to produce an output prediction. Therefore, we compute the loss function between input and output values in all iterations to measure the error rate using the Mean Square Error (MSE) as given in Eq. (7).

$$MSE = \frac{1}{N}\sum_i^N |x_i - y_i|^2 \tag{7}$$

Note that x_i denotes the encoded input value while y_i denotes the decoded output value in the model parameters. The MSE is determined in the following way. Note that the MSE is kept to be minimized

during the training process. In this context, MSE has been used as the loss function where it is subject to the minimization at each subsequent iteration in the model using Adam optimizer [26]. Thus, the weights at each layer are updated accordingly.

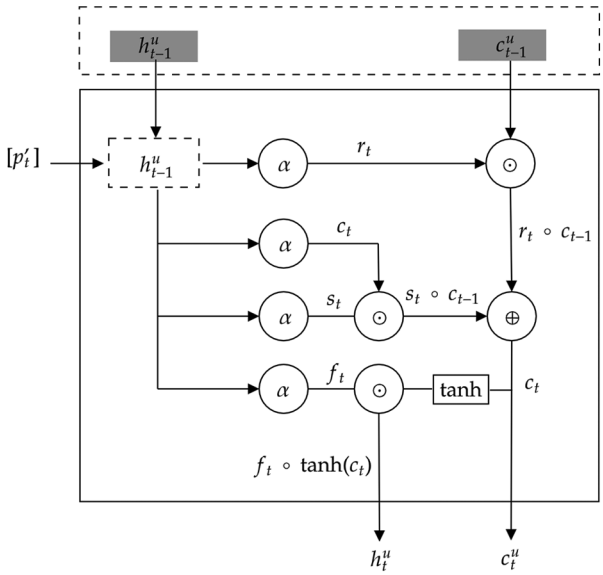

Figure 2: The structure of utilized LSTM for feature extraction

3.2 Attack Detection Model

In this study, we employ LSTM to perform the sequential classification on the input user behavior to detect whether the behavior is normal or malicious. Because the LSTM network is trained on normal behavior, the model learns the users' normal behavior. The LSTM network is given both normal and anomalous sequences throughout the testing phase. Typically, the model produces a low MSE rate for the normal data (normal behavior) while resulting in a large error rate in abnormal data (malicious behavior) since it learns on a normal pattern. Thus, we dedicate a threshold ϑ to classify output to either normal or abnormal behavior. The parameter ϑ is defined in Eq. (8).

$$\vartheta = \gamma\left(\frac{1}{N}\sum\nolimits_i^N \varepsilon_i\right) \tag{8}$$

where γ is some constant and ε_i is the ESM for the i^{th} vector. Thus, the model would predict two classes as given in Eq. (9).

$$y = \begin{cases} abnormal - behavior, & if\ \varepsilon > \vartheta \\ normal - behavior, & otherwise \end{cases} \tag{9}$$

where y is the predicted class in the model.

4 Performance Evaluation

This section presents an evaluation to determine the effectiveness of the proposed model. The evaluation is concerned with examining the performance of the proposed model at the hyper-parameter tune. Moreover, the evaluation is conducted to compare the performance of the proposed model with traditional anomaly detection approaches such as support vector machine (SVM). Computer emergency and response team

(CERT) dataset [27] has been used for the evaluation process in this study. Thus, we first begin introducing details of the dataset, including the structure of the data and.

4.1 Dataset

The dataset is publicly available on [27]. The latest version released is CERT v4.2 which has been used in this study. It is worth mentioning that CERT v4.2 has more instances of abnormal activities than the previous versions. It is synthetic data consisting of user activities for 1000 users with generated events of ≈ 32M (log lines) in 502 days. Inside the total recorded activities, there are about 7K log lines which represent anomalous activities; these logs have been inserted manually by experts to the data records.

The recorded activity is logline consists of data pertaining to logon/logoff, device, and hypertext transfer protocol (HTTP). Fig. 3 shows a snippet of the data fields which represent the activities of users. In the experiment, each user activity is parsed to a vector, including id, date, user, pc, and activity type.

```
|    Data-frame     :                    Fields                          |
|- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -|
|*  logon.csv       : id | date | user | pc | activity (Logon/Logoff)    |
|                                                                        |
|*  device.csv      : id | date | user | pc | activity (connect/disconnect)  |
|                                                                        |
|*  HTTP.csv        : id | date | user | pc | url                        |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Figure 3: Snippet of the data fields that represent the user activities in the dataset

The dataset is divided into two parts: training and testing. The training dataset is 70% of the dataset in which it is directed at model training with hyper-parameter adjustments. The testing subset is 30% of the dataset so it has been utilized to assess the performance of the proposed model.

4.2 Evaluation

We use several evaluation methods to evaluate the proposed model. Confusion Matrix (CM) is used to measure the performance of classification for the proposed model. The classification can either be predicted correctly or incorrectly. In concise, CM handles four different conditions for binary classification problems, as applied in this study, as follows: True Positive (TP), False Negative (FN), True Negative (TN), and True Negative (TN). Thus, TP represents whether the model correctly predicts the instance as positive while FN represents the number of positive instances that are predicted as negative. On the other hand, FP presents if the model wrongly predicts an instance as positive, whereas TN calculates the instances predicted as negative correctly. Consequently, CM is used to measure the Receiver Operator Characteristics (ROC) curves to evaluate the performance of the proposed models.

4.3 Results

The conducted experiments have been implemented using the well-known machine learning library TensorFlow [28]. The default parameters have been configured to 5 for the hidden layers and 50 for the patch size, where the epoch is set to 10. The learning rate is also set to 1. Note that the parameters are configured according to the RNN with LSTM regularization as proposed in [29]; therefore, the parameters are tested and tuned to obtain the best performance. Tab. 1 shows the different parameter configurations for three RNN-LSTM models. Thus, the conducted experiments evaluate the ROC curves for each of the RNN-LSTM models. Fig. 4 depicts the ROC results concerning each model. It can be seen from the Fig. 4 that different configurations yield different results. The best performance is recorded for RNN-LSTM 1 with AUC 0.97. The other models recorded AUC with 0.96 and 0.73 respectively. The

overall result indicated promising performance for the proposed RNN-LSTM model which considers the streaming of user activities to represent the use's behavior for protentional cyber-attacks in the network.

Table 1: Parameter settings for the RNN-LSTM models.

RNN model	#Units per layer	#Mini-batch	#Epoc
RNN-LSTM1	55	20	10
RNN-LSTM2	39	20	10
RNN-LSTM3	20	20	10

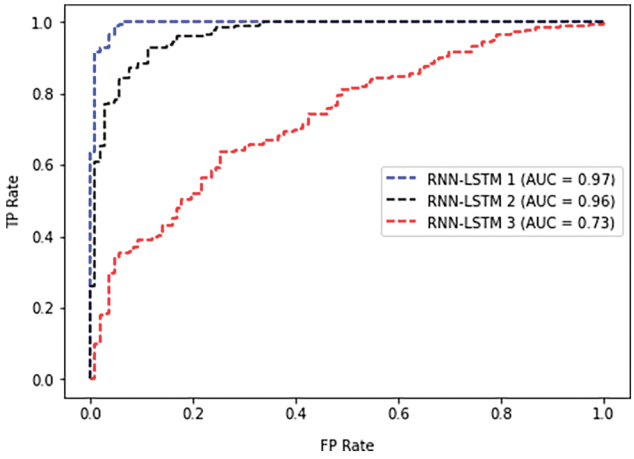

Figure 4: Illustration for the ROC curves of the RNN-LSTM models

The evaluation is also conducted to measure the proposed RNN-LSTM model's performance to other literature models. We used SVM as it is used in the literature [30] as a baseline for anomaly detection. It learns a decision hypersphere to classify positive instances from the anomalies in the data suitable to detect abnormal user behaviour. Fig. 5 shows the ROC results of the best performing RNN-LSTM 1 compared with SVM. The result show better performance for the proposed RNN-LSTM 1 comparing with SVM.

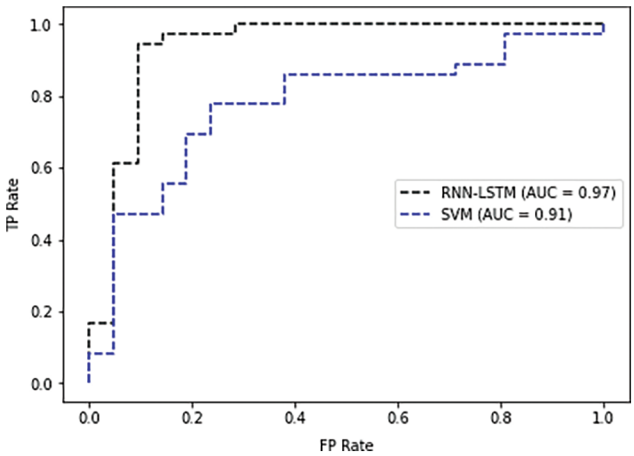

Figure 5: ROC result of best RNN-LSTM comparing with SVM baseline

5 Conclusion

This study presents a novel framework for detecting cyber-attacks combining ML, and UBA is provided. The main contribution is that the proposed framework represents user behavior as a series of whole events to represent user activities in a network. Thus, the model can express latent patterns, unlike the case of representing abstracted features. The user sequences are fed into RNN model to automatically extracting features. To maintain the sequential extraction of data, we used LSTM. In this context, the model can recognize the frequency of routine activities to profile a distinctive user's behavior. Following that, LSTM is also used to train the model for detecting unseen behavior to either regular or irregular behavior. The importance of the suggested framework stems from the rise in cyberattacks, particularly when the attack originates from inside the network. Insider attacks are typically more difficult to detect since security protocols struggle to distinguish assaults from trusted network resources, such as users. As a result, user activity can reveal interesting patterns, with regular behavior reflecting typical network workflow, whereas abnormal behavior triggers an alarm for a possible cyberattack. The MSE between the input and output was kept to a minimum during the training phase. The model produced low MSE for normal patterns and high MSE for anomaly patterns throughout the testing phase. Therefore, a threshold value was used to manage the performance parameters.

In this study, the features are learnt from the raw data in which it can incorporate the latent behavior of the user. However, the relations between different features can be considered to extract a more meaningful representation of the user's behavior. The latent behavior of the user, from which a more accurate feature representation can be extracted, is known as distributed behavior. However, traditional sequential processing cannot capture this phenomenon and its complexity. This is especially required in the case of insider thread detection due to the sparsity of user's actions and activates recorded from different sources, which can make considering the entire series increase the complexity of the model. Thus, in the future, it is worth discussing how the relations between various activities would be learnt to improve feature extraction for better model classification.

Acknowledgement: The authors acknowledge the support received from Al Baha University, Saudi Arabia, under the fund 8/1440.

Funding Statement: The study is supported by the fund received from Al Baha University, 8/1440.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Jang-Jaccard and S. Nepal, "A survey of emerging threats in cybersecurity," *Journal of Computer and System Sciences*, vol. 80, no. 5, pp. 973–993, 2014.

[2] A. Naway and Y. Li, "A review on the use of deep learning in android malware detection," *arXiv preprint* vol. 1812.10360, pp. 1–15, 2018.

[3] F. Yuan, Y. Cao, Y. Shang, Y. Liu, J. Tan *et al.,* "Insider threat detection with deep neural network," in *Int. Conf. on Computational Science*, Berlin, Springer, pp. 43–54, 2018.

[4] P. A. Legg, "Visualizing the insider threat: Challenges and tools for identifying malicious user activity," in *2015 IEEE Symp. on Visualization for Cyber Security (VizSec)*, New Jersey, IEEE, pp. 1–7, 2015.

[5] N. Khan, J. Abdullah and S. A. Khan, "Defending malicious script attacks using machine learning classifiers," *Wireless Communications and Mobile Computing*, vol. 17, no. 6, pp. 1–9, 2017.

[6] N. A. Khan, M. Y. Alzahrani and H. A. Kar, "Hybrid feature classification approach for malicious javaScript attack detection using deep learning," *International Journal of Computer Science and Information Security*, vol. 18, no. 5, pp. 79–90, 2020.

[7] L. Chen, Y. Ye and T. Bourlai, "Adversarial machine learning in malware detection: Arms race between evasion attack and defense," in *2017 European Intelligence and Security Informatics Conf. (EISIC)*, New Jersey, IEEE, pp. 99–106, 2017.

[8] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian *et al.,* "Automated classification and analysis of internet malware," in *Int. Workshop on Recent Advances in Intrusion Detection*, Berlin, Springer, pp. 178–197, 2007.

[9] R. A. Dunne, *A Statistical Approach to Neural Networks for Pattern Recognition*. vol. 702. New York: John Wiley & Sons, 2007.

[10] J. Z. Kolter and M. A. Maloof, "Learning to detect malicious executables in the wild," in *Proc. of the 10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Seattle, USA, pp. 470–478, 2004.

[11] Z. Ma, H. Ge, Y. Liu, M. Zhao and J. Ma, "A combination method for android malware detection based on control flow graphs and machine learning algorithms," *IEEE Access*, vol. 7, pp. 21235–21245, 2019.

[12] A. Kwan, "Malware detection at the microarchitecture level using machine learning techniques," vol. 2005.12019, pp. 1–28, 2020.

[13] M. -Y. Kan and H. O. N. Thi, "Fast webpage classification using URL features," in *Proc. of the 14th ACM Int. Conf. on Information and Knowledge Management*, Bremen, Germany, pp. 325–326, 2005.

[14] J. Z. Kolter and M. A. Maloof, "Learning to detect and classify malicious executables in the wild," *Journal of Machine Learning Research*, vol. 7, no. 12, pp. 2721–2744, 2006.

[15] A. Abbasi, F. M. Zahedi and S. Kaza, "Detecting fake medical web sites using recursive trust labeling," *ACM Transactions on Information Systems (TOIS)*, vol. 30, no. 4, pp. 1–36, 2012.

[16] I. A. AL-Taharwa, H. -M. Lee, A. B. Jeng, K. -P. Wu, C. -S. Ho *et al.,* "Jsod: Javascript obfuscation detector," *Security and Communication Networks*, vol. 8, no. 6, pp. 1092–1107, 2015.

[17] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An *et al.,* "Significant permission identification for machine-learning-based android malware detection," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3216–3225, 2018.

[18] A. Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp *et al.,* "Yes, machine learning can be more secure! a case study on android malware detection," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 4, pp. 711–724, 2017.

[19] M. K. Alzaylaee, S. Y. Yerima and S. Sezer, "Dl-droid: Deep learning based android malware detection using real devices," *Computers & Security*, vol. 89, no. 5, pp. 101663, 2020.

[20] H. Han, S. Lim, K. Suh, S. Park, S. J. Cho *et al.,* "Enhanced android malware detection: An SVM-based machine learning approach," in *2020 IEEE Int. Conf. on Big Data and Smart Computing (BigComp)*, New Jersey, IEEE, pp. 75–81, 2020.

[21] A. Litan and M. Nicolett, "Market guide for user behavior analytics," 2014. [Online]. Available: https://www.gartner. com/doc/2831117/market-guide-userbehavior-analytics.

[22] S. Chen, S. Chen, N. Andrienko, G. Andrienko, P. H. Nguyen *et al.,* "User behavior map: Visual exploration for cyber security session data," in *2018 IEEE Symp. on Visualization for Cyber Security (VizSec)*, New Jersey, IEEE, pp. 1–4, 2018.

[23] X. Zhao, M. Z. A. Bhuiyan, L. Qi, H. Nie, W. Tang *et al.,* "Trcmp: A dependable app usage inference design for user behavior analysis through cyber-physical parameters," *Journal of Systems Architecture*, vol. 102, no. 1, pp. 101665, 2020.

[24] S. L. Pfleeger and D. D. Caputo, "Leveraging behavioral science to mitigate cyber security risk," *Computers & Security*, vol. 31, no. 4, pp. 597–611, 2012.

[25] H. Hewamalage, C. Bergmeir and K. Bandara, "Recurrent neural networks for time series forecasting: Current status and future directions," *International Journal of Forecasting*, vol. 37, no. 1, pp. 388–427, 2021.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint*, vol. 1412.6980, pp.1–16, 2014.

[27] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *2013 IEEE Security and Privacy Workshops*, New Jersey, IEEE, pp. 98–104, 2013.

[28] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis *et al.,* "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symp. on Operating Systems Design and Implementation (OSDI-16)*, Savannah, GA, USA, pp. 265–283, 2016.

[29] W. Zaremba, I. Sutskever and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint*, vol. 1409.2329, pp. 1–8, 2014.

[30] S. Yuan, P. Zheng, X. Wu and Q. Li, "Insider threat detection via hierarchical neural temporal point processes," in *2019 IEEE Int. Conf. on Big Data (Big Data)*, New Jersey, IEEE, pp. 1–8, 2019.