Developing Novel Deep Learning Models to Detect Insider Threats and Comparing the Models from Different Perspectives

Araştırma Makalesi/Research Article

Yasin Görmez*[1], Halil Arslan[2], Yunus Emre Işık[1], Veysel Gündüz[3]

[1]Management Information System, Sivas Cumhuriyet University, Sivas, Türkiye
[2]Computer Engineering, Sivas Cumhuriyet University, Sivas, Türkiye [3]Basis Department, Detaysoft, İstanbul, Türkiye

yasingormez@cumhuriyet.edu.tr, harslan@cumhuriyet.edu.tr, yeisik@cumhuriyet.edu.tr, veysel.gunduz@detaysoft.com

(Geliş/Received:06.11.2023; Kabul/Accepted:16.01.2024)
DOI: 10.17671/gazibtd.1386734

Abstract— Cybersecurity has become an increasingly vital concern for numerous institutions, organizations, and governments. Many studies have been carried out to prevent external attacks, but there are not enough studies to detect insider malicious actions. Given the damage inflicted by attacks from internal threats on corporate reputations and financial situations, the absence of work in this field is considered a significant disadvantage. In this study, several deep learning models using fully connected layer, convolutional neural network and long short-term memory were developed for user and entity behavior analysis. The hyper-parameters of the models were optimized using Bayesian optimization techniques. Experiments analysis were performed using the version 4.2 of Computer Emergency and Response Team Dataset. Two types of features, which are personal information and numerical features, were extracted with respect to daily activities of users. Dataset was divided with respect to user or role and experiment results showed that user based models have better performance than the role based models. In addition to this, the models that developed using long short-term memory were more accurate than the others. Accuracy, detection rate, f1-score, false discovery rate and negative predictive value were used as metrics to compare model performance fairly with state-of-the-art models. According the results of these metrics, our model obtained better scores than the state-of-the-art models and the performance improvements were statistically significant according to the two-tailed Z test. The study is anticipated to significantly contribute to the literature, as the deep learning approaches developed within its scope have not been previously employed in internal threat detection. Moreover, these approaches have demonstrated superior performance compared to previous studies.

Keywords— user and entity behavior analysis, machine learning, deep learning, insider threat, cyber security

İç Tehditlerin Tespit Edilmesi için Özgün Derin Öğrenme Modellerinin Geliştirilmesi ve Modellerin Farklı Perspektiflerde Karşılaştırılması

Özet— Siber güvenlik, çok sayıda kurum, kuruluş ve devlet için zamanla hayati öneme sahip bir konu haline gelmiştir. Mevcut çalışmalar incelendiğinde, dış saldırıları önlemek için birçok çalışma yapıldığı, ancak iç tehditleri tespit etmeye yönelik çalışmaların yeterli olmadığı kanısına varılmaktadır. İç tehditlerden gelen saldırıların kurum itibarlarına ve mali durumlarına verdiği zararlarda göz önüne alındığında, bu alanda çalışma eksikliği büyük bir dezavantaj olarak değerlendirilmektedir. Bu çalışmada, kullanıcı ve varlık davranış analizi için tam bağlı katman, evrişimsel sinir ağı ve uzun kısa süreli hafıza kullanan çeşitli özgün derin öğrenme modelleri geliştirilmiştir. Modellerin hiper parametreleri Bayesian optimizasyon teknikleri kullanılarak optimize edilerek, analizler, Computer Emergency and Response Team Dataset veri kümesinin 4.2. sürümü kullanılarak yapılmıştır. Kullanıcıların günlük aktivitelerine göre kişisel bilgiler ve sayısal özellikler olmak üzere iki tür özellik çıkarılmıştır. Veri seti kullanıcı veya role göre bölünmüş ve deney sonuçlarına kullanıcı tabanlı modellerin rol tabanlı modellere göre daha iyi performansa sahip olduğunu gözlemlenmiştir. Ayrıca uzun kısa süreli hafızayı kullanarak geliştirilen modellerin diğerlerine göre daha başarılı sonuçlar elde ettiği gözlemlenmiştir. Model performansını literatürdeki çalışmalar ile adil bir şekilde karşılaştırmak için, başarı oranı, tespit oranı, fl puanı, yanlış keşif oranı ve negatif tahmin değeri metrikleri kullanılmıştır. Bu metriklerin sonuçlarına göre modelimiz, literatürde var olan modellere göre daha iyi performans skorları elde etmiş ve iki kuyruklu Z testine göre performans iyileştirmeleri istatistiksel olarak anlamlı bulunmuştur. Çalışma kapsamında geliştirilmiş olan derin öğrenme yaklaşımlarının daha önce iç tehdit tespitinde kullanılmamış olmasından ve önceki çalışmaların performanslarından yüksek bir performans elde etmesinden dolayı çalışmanın literatüre büyük bir katkı sağlayacağı kanaatine varılmıştır.

Anahtar Kelimeler— kullanıcı ve varlık davranış analizi, makine öğrenmesi, derin öğrenme, iç tehdit, siber güvenlik

1. INTRODUCTION

The concept of cyber security has become more and more audible, with technology taking place in more areas of our lives. Information is more important for institutions now and the institutions store their data on different servers. Although systems designed to access data from different locations provide convenience for institutions, they cause data security to become more complex. Some of these data such as employee, marketing strategy, application info and project documents have crucial importance for institutions. In this regard, information security practices and institutions' strategy should be compatible and information security applications must secure all systems of institutions [1]. If one of the institution's information system is vulnerable to cyber-attacks, all systems are at risk to attacks. Companies can experience financial losses or loss of reputation because of the information theft. Although most organizations use basic security precautions, the number of security incidents is increasing. Therefore, more resources, which are human, hardware and software, should be allocated for security systems.

Information technology systems of institutions are very resistant to external attacks such as DDOS, malware, phishing and password hacking thanks to advance applications. According to the research, more than 60% of businesses are using technical information security prevention practices such as antivirus program, firewall, intrusion detection system, virtual private networks and anti-spyware software. In addition to this, attacks from inside and outside are increasing [2]. Although these applications are very successful to external attacks, they are not yet sufficient to detect internal attacks. According to a study conducted by the Ponemon Institute in 2017 with 237 companies from 6 countries, it was concluded that insider threats are the most expensive cyber-attack situations and it is foreseen that insider threats will increase in the future [3]. Insider threats are typically perpetrated by existing employees or authorized individuals. In large organizations, user activities and network traffic become highly intricate. The human factor plays a more significant role in the occurrence of internal threats than external threats. However, not every person working in large organizations possesses the same level of experience in information security. Considering these complexities, detecting internal threats emerges as a formidable challenge for institutions.

Especially large companies use security software that collects logs and event data generated by all users, servers, network devices and firewalls to monitor and analyze all security-related events in their infrastructure. This system is called as security information and event management (SIEM). It is possible to make user and entity behavior analysis (UEBA), which is one of the most common approach to detect insider threads, thanks to data collected by SIEM systems.

UEBA aims to determine malicious actions that coming from the insiders. These actions can be done by outsiders who have impersonated employees or employees who acted maliciously or negligently. Instead of detecting malicious software or antivirus, it is desired to detect anomalies in the behavior of users with UEBA. This situation causes the problem to be difficult because what is an anomaly for one user may not be an anomaly for another user. For example, frequent review of employee information may not be an anomaly for someone who working in the human resources department, but it may be considered as an anomaly for those working in other departments. For this reason, UEBA systems are also developed separately for situations such as department, job description, user or role in the literature [4]. The data used within the scope of UEBA is important for the business processes of the companies and requires confidentiality. Therefore, it is very difficult to find data for academic studies. The Computer Emergency and Response Team Dataset (CERT), which generated synthetically thanks to support of Carnegie Mellon University, is frequently encountered in studies in UEBA field [5], [6].

To date, supervised or unsupervised machine learning algorithms have been used frequently for UEBA. Xiangyu et al. combined one class support machines, recurrent neural networks and isolation forest for UEBA and they reached 91.60%, 93.10% and 100% accuracy, precision and recall respectively [7]. Tuor et al. proposed long short-term memory and deep neural networks based unsupervised deep learning model for UEBA. The proposed model was trained on version 6.2 of CERT dataset and anomaly score for each sample was computed. Experiment results show that, proposed model outperformed the isolation forest, support vector machines and principal component analysis models [8]. Lin et al. applied proposed hybrid machine learning model based on deep belief network and one-class support vector machines on version 4.2 of CERT dataset. They firstly extracted hidden features using deep belief network, subsequently samples were classified using support vector machine model. As a result of experiment analysis, they obtained 87.70%, 81.04% and 12.18% accuracy, detection rate and false positive rate respectively [9]. Yuan et al. extracted user behavior features with user actions and abstracted temporal features using long short-term memory network networks. Afterwards, they converted extracted features to fixed size vector and they obtained 94.49% area under the ROC curve score by training convolutional neural network model [10]. Lo et al. detected the insider threads using distance measurements that were Damerau–Levenshtein, Jaccard and Cosine Distance, and they obtained 39%, 36% and 47% detection rate. They summarized that distance measurement techniques were better than the hidden markov model because they were faster to train and detect [11]. Li and Zincir-Heywood applied both supervised and unsupervised learning models on CERT dataset using numerical and sequential data. As a result of the experiments, they reached 79.75% and 73.10% detection rate with self-organizing maps and C4.5 decision tree respectively for weekly data; 84.60% and 99.87% detection rate with self-organizing maps and C4.5 decision tree respectively for daily data [12]. Igbe and Saadawi

applied artificial immune system algorithm on CERT dataset with 21 extracted features and they reached 83.45%, 4.60% and 89.00% true positive rates, false positive rates and accuracy respectively [13]. Hall et al. applied neural networks, support vector machines, naïve bayes, decision tree, random forest and logistic regression on version 4.2 of CERT dataset. According to the analysis results they obtained 95.8%, 91.3%, 95.4%, 97.5%, 96.1% and 96.5% accuracy for neural networks, naïve bayes, support vector machines, random forest, decision tree and logistic regression respectively and they increased accuracy to 95.8%, 97.2%, 97.2%, 97.2%, 97.2% and 96.8% by applying boosting on these algorithm [14]. Aldairi et al. achieved 93% accuracy and 92% precision on version 4.2 of CERT dataset using isolation forest and one-class support vector machine [15]. Le and Zincir-Heywood extracted features from version 4.2 and 5.2 of CERT dataset using autoencoder, principle component analysis and random projection and they applied logistic regression, artificial neural networks, naïve Bayes and random forest as a classifier. As a result of experiment analysis, they achieved 98.33% accuracy for version 4.2 and 99.04% accuracy for version 5.2 [16]. In other study, Le and Zincir-Heywood achieved 92.93%, 99.54% and 97.13% accuracies using logistic regression, random forest and artificial neural networks respectively with user-session based feature extraction techniques [17]. Nasser et al. obtained 92% accuracy using gated recurrent neural network architecture [18]. Sharma et al. proposed long short-term memory based autoencoder model for user behavior analytics and they achieved 90.17%, 91.03% and 9.84% accuracy, true positive rate and false positive rate respectively [19]. Tian et al. used several deep learning layers, which were long short-term memory, convolutional long short-term memory and multi-layer perceptron, to develop multi-model based system for insider thread detection [20]. Al-Shehari and Alsowail used synthetic minority oversampling technique to overcome imbalance problem of CERT dataset and they obtained 0.79, 1.00, 1.00, 0.84 and 0.99 area under curve score using logistic regression, decision tree, random forest, naïve Bayes and k nearest neighborhood respectively [21]. Nasir et al. achieved 90.60% accuracy, 97% precision and 94% F1 score by using long short-term memory based autoencoder model [22]. Su et al. utilized online version of recurrent neural networks for the version 6.2 of CERT dataset and they achieved 95.3% f-measure [23]. Dosh et al. achieved 94.68% accuracy using three traditional machine learning algorithms [4]. 96% accuracy was obtained by Pantelidis et al. using autoencoder and variational autoencoder based deep learning model [24]. Al-Mhiqani et al. achieved 97% accuracy, 2.88% false positive rate with proposed multilayer framework model [25]. AlSlaiman et al. et al. applied sentiment analysis techniques on HTTP traces, emails, and files of CERT dataset and they combined it with different data representation. For this purpose, they proposed a long short-term memory based deep learning model and they achieved 0.29% false positive rate, 2.27% false negative rate and 97% area under the curve [26]. Li et al. integrated memory-augmented network into autoencoder and they obtained 94.56% area under the curve score [27].

When reviewing the literature, it becomes evident that numerous studies have been conducted on UEBA, resulting in the development of various deep learning models. When examining studies focused on detecting abnormal behavior through the analysis of individual user patterns, it becomes apparent that some studies have achieved an accuracy level of approximately 90% [22]. However, in studies with a success rate of around 98%, the detection rate remained at 67% [16]. In this context, it was concluded that enhancements should be made to the deep learning models used for classifying insider threats. Therefore, our study aimed to develop deep learning models that achieve high accuracy and detection rates while maintaining a low false discovery rate.

It is known that cyber-security is one of the main issue in many area [28]. In this paper, malicious actions from insiders were predicted using CERT datasets. For this purpose, preprocessing steps were applied on CERT dataset and samples were divided into parts based on roles or users. After preprocessing steps, features were extracted and several deep learning models were developed. User behavior is time-dependent, and a malicious action is a combination of a user's activities over time. Therefore, Convolutional neural networks (CNN), long short-term memory (LSTM) and fully connected layer (FLC), which were suitable for time series problem, were used to develop deep learning models. The hyper-parameters are one of the most important factors for the performance of deep learning models. Because of this reason, Bayesian optimization technique, which is faster and more effective than traditional techniques, were used in this study to optimize hyper-parameters [29]–[32]. Bayesian optimization can explore larger spaces. Moreover, unlike grid search, which tests specific values, Bayesian optimization allows for trying any value within the specified range. Consequently, this method can identify more suitable values for hyper-parameters. The performance of the proposed models was measured using five different metrics that are accuracy, detection rate, f1-score, negative predictive value and false discovery rate.

The primary aim of the study is to predict internal threats with higher accuracy compared to the existing literature, owing to the novel approaches and models developed within this research. First innovative aspect of our work is the deep learning approaches used for UEBA were developed by us specifically for this study. In addition to this, to the best of our knowledge, it is first to use Bayesian optimization techniques with deep learning approaches for UEBA on CERT datasets.

The paper is organized as follows: Section II explained the benchmark datasets, feature extraction steps and architectures of proposed models. Section III presents the experimental results, encompassing hyper-parameter optimization, the performance metrics of the trained models, a comparison with state-of-the-art methods, and a two-tailed Z-test. Finally, the paper is concluded and future works were explained in the section IV.

2. MATERIALS AND METHODS

2.1. Benchmark Dataset

In this study, the Computer Emergency and Response Team (CERT) dataset, which is insider threat benchmark datasets and generated synthetically, were used to train the proposed models and to assess the performance of the proposed model [6]. This dataset consists of device connection, sent or received email, transferred or copied file, http and login activities from 1000 users for 500 days. In addition to this, CERT dataset also contains LDAP files and detailed personal information for each user. Data imbalance has a detrimental impact on model performance in machine learning studies [33]. Ideal conditions for high-performance models involve having a balanced distribution of samples from different classes in the training dataset. In this study, version 4.2 of the CERT dataset was utilized to mitigate the effect of data imbalance on model performance, as it contains the largest number of samples related to malicious actions. Many actions in these datasets are standard actions and also the Computer Emergency and Response Team determined three scenarios to define malicious action. Explanation of these scenarios were given below. Note that, these scenarios were taken directly from the dataset information package [5].

- *"User 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. Leaves the organization shortly thereafter [5]."*
- *"User begins surfing job websites and soliciting employment from a competitor. Before leaving the company, they use a thumb drive (at markedly higher rates than their previous activity) to steal data [5]."*
- *"System administrator becomes disgruntled. Downloads a keylogger and uses a thumb drive to transfer it to his supervisor's machine. The next day, he uses the collected keylogs to log in as his supervisor and send out an alarming mass email, causing panic in the organization. He leaves the organization immediately [5]."*

In other versions of the dataset, these scenarios may change. If an employee has a malicious action, these actions are specified separately in each scenario. The details about this dataset can be found on CERT information package [5].

2.2. Feature Extraction

As mentioned earlier, the CERT datasets consisting of raw information from 5 different activities and users' personal information. In order to make UEBA, this data must be combined into a single file in a way that the machine learning model can understand. Because of this reason, some preprocessing steps were applied on these files then features were extracted. Firstly, each activity type in separate files was combined on a per-user basis and sorted by date. After that, based on one-day activities for all users, features were extracted. In this study a samples represent the daily activities of users. Considering the daily activities, two types of features, which are personal information and numerical features, were extracted. Personal information consists of user role, user functional unit, user team and "o", "c", "e", "a" and "n" results of OCEAN test. The details about OCEAN test can be found at the test information page [34]. The numerical features were computed considering the number of daily activities with respect to activity types. In this context, 24 values, which are listed below, were computed for each sample.

- n_of_logOnLogOf_work_on_own_pc: Number of logon-logoff in work on time with user own personal computer.
- n_of_logOnLogOf_work_off_own_pc: Number of logon-logoff in work off time with user own personal computer.
- n_of_logOnLogOf_work_on_other_pc: Number of logon-logoff in work on time with other user's personal computer.
- n_of_logOnLogOf_work_off_other_pc: Number of logon-logoff in work off time with other user's personal computer.
- n_of_email_work_on_own_pc: Number of email in work on time with user own personal computer.
- n_of_email_work_off_own_pc: Number of email in work off time with user own personal computer.
- n_of_email_work_on_other_pc: Number of email in work on time with other user's personal computer.
- n_of_email_work_off_other_pc: Number of email in work off time with other user's personal computer.
- n_of_device_work_on_own_pc: Number of device in work on time with user own personal computer.
- n_of_device_work_off_own_pc: Number of device in work off time with user own personal computer.
- n_of_device_work_on_other_pc: Number of device in work on time with other user's personal computer.
- n_of_device_work_off_other_pc: Number of device in work off time with other user's personal computer.
- n_of_file_work_on_own_pc_toMedia: Number of file copy to temporary device from company's

computer in work on time with user own personal computer.

- n_of_file_work_off_own_pc_toMedia: Number of file copy to temporary device from company's computer in work off time with user own personal computer.
- n_of_file_work_on_other_pc_toMedia: Number of file copy to temporary device from company's computer in work on time with other user's personal computer
- n_of_file_work_off_other_pc_toMedia: Number of file copy to temporary device from company's computer in work off time with other user's personal computer.
- n_of_file_work_on_own_pc_fromMedia: Number of file transfer to company's computer from temporary device in work on time with user own personal computer.
- n_of_file_work_off_own_pc_fromMedia: Number of file transfer to company's computer from temporary device in work off time with user own personal computer.
- n_of_file_work_on_other_pc_fromMedia: Number of file transfer to company's computer from temporary device in work on time with other user's personal computer.
- n_of_file_work_off_other_pc_fromMedia: Number of file transfer to company's computer from temporary device in work off time with other user's personal computer.
- n_of_http_work_on_own_pc: Number of website activity in work on time with user own personal computer.
- n_of_http_work_off_own_pc: Number of website activity in work off time with user own personal computer.
- n_of_http_work_on_other_pc: Number of website activity in work on time with other user's personal computer.
- n_of_http_work_off_other_pc: Number of website activity in work off time with other user's personal computer

In feature extraction stage, numerical features were calculated considering two important situations: activity time and computer used for activity. It is thought that the increase in the use of computers of other employees and the activities performed in work off time are two important factors in determining the insider malicious actions. In addition to this, the information of whether data is received from the media or data is transferred to the media was also used to compute numerical features for file activities. Considering all these situations, four numerical features were computed for logon, email, device and http activities separately and eight numerical feature were computed for file activities. Therefore, a total of 24 numerical features were computed using five different activity records.

After feature extraction steps, each sample was labeled as non-malicious (0) or malicious (1) using the answers file of CERT dataset. The answers file consists of malicious event from 5 different activities according to 3 scenarios of CERT datasets. It is assumed that, if any sample contains one of these malicious activities, this sample labeled with "1". Otherwise the sample was labeled with "0". As a result of preprocessing and feature extraction steps, 330452 samples, which consist of 986 malicious and 329466 non-malicious, and 29 features were obtained from 1000 users. Comprehensive details regarding dataset preprocessing and feature extraction can be found in the study proposed by Görmez et al. [35]. The code for these processes is available on the Detaysoft GitHub page [36].

2.3. Proposed Models

In this study, two different approaches, which were role based and user based, were used to develop proposed deep learning models. In these approaches, small datasets were generated by dividing the dataset according to role type or user. In the role based models, dataset divided into 42 parts and in the user based models, dataset divided into 1000 parts. In these approaches, models consisting of input layer and hidden layers for each section were generated separately, then combined in a single output layer to create the main model. There are two reasons to use these approach. There are cases where an action that is malicious for one person or group is not malicious action by another person or group. For example, a behavior that is normal for a person working in the IT department may be malicious for the human resources department. Therefore, training the UEBA models separately based on user or role affects the accuracy of model positively. Considering this situation, the input and hidden layers of our models were created user or role based separately. In this case, in order to prevent overfitting situations that may occur in the model and to generalize the model, the models created separately on the basis of role or person were combined in a single output layer. A total six model were proposed using these approaches. One of them was used only for role based, three of them were used only for user based and remaining were used both role and user based dataset. The architectures of these models are shown in figures below (Figure 1-5). The models can be summarized as traditional artificial neural network (MLP), convolutional neural network model (CNN), long short-term memory model in deep two (LSTM_2), long short-term memory model in deep four (LSTM_4), combination of convolutional neural network and long short-term memory in deep two (CNN_LSTM_2) and combination of convolutional neural network and long short-term memory in deep four (CNN_LSTM_4).

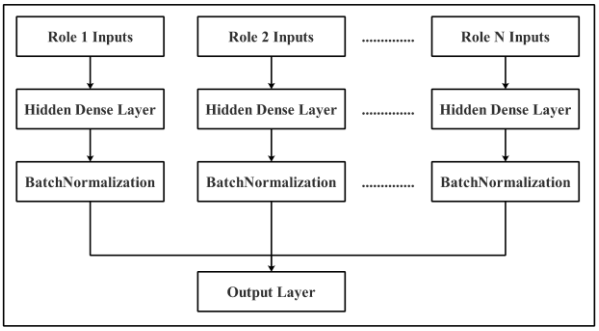

Figure 1. Architecture of role based multi-layer perceptron model

MLP model, which was used only for role based approach, was shown in the figure 1. This model has contained 42 input layer, because there were 42 different roles in the CERT dataset. A fully connected and Batch Normalization layers were added after each input layer. In fully connected layer, relu was used as an activation function and number of neurons were optimized. Note that, all hidden layers are identical which means that the number of neurons were same in each fully connected layer. All of this layers were concatenated at the output layer with softmax function to generate generalized classification layer.

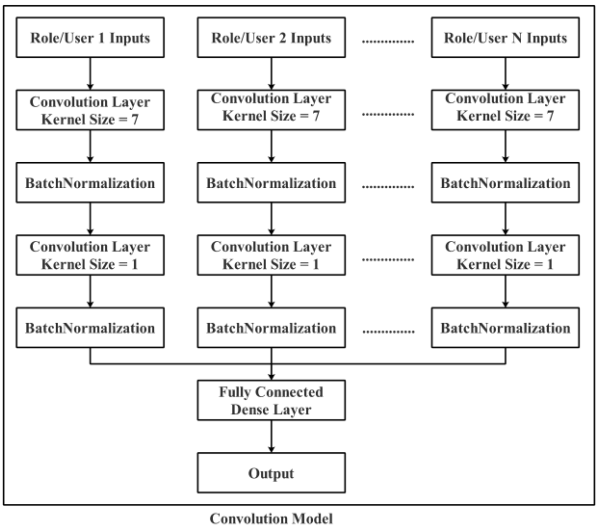

Figure 2. Architecture of role and user based convolution model

Convolution model, which were used for both role or user based approaches, was shown in the figure 2. In this model, 42 input layers were used for role based approach and 1000 input layers were used for user based approach. Note that, two different models were generated for user and role based approaches separately. In this model, each input layer was followed by convolutional layer with kernel size 7, Batch Normalization layer, convolutional layer with kernel size 1 and Batch Normalization layer in sequential order. After that, all of these layers for each user or role were concatenated. Later, followed by the shared fully connected and classification layer and the model was completed. Note that, similar to MLP model, layers created for each user or role are identical. Hyper-parameter optimization was applied user based and role based models separately.

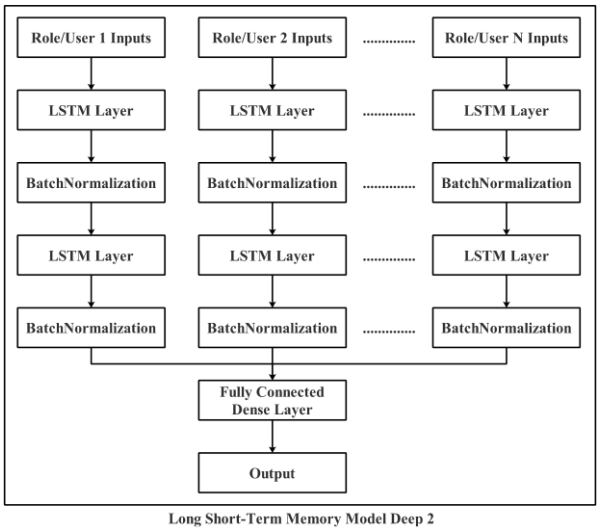

Figure 3. Architecture of role and user based long short-term memory model at depth 2

Long short-term memory model in deep 2, which were used for both role or user based approaches, was shown in the figure 3. Similar to convolution model, also in this model, 42 input layers were used for role based approach, 1000 input layers were used for user based approach and two different models were generated for user and role based approaches separately. In this model, each input layer was followed by one directional long short-term memory layer, Batch Normalization layer, one directional long short-term memory layer and Batch Normalization layer in sequential order. After that, all of these layers for each user or role were concatenated. Later, followed by the shared fully connected and classification layer and the model was completed. Because there was a two long short-term memory layer, this model was named as LSTM_2. Similar to other models, layers created for each user or role are identical and hyper-parameter optimization was applied user based and role based models separately.

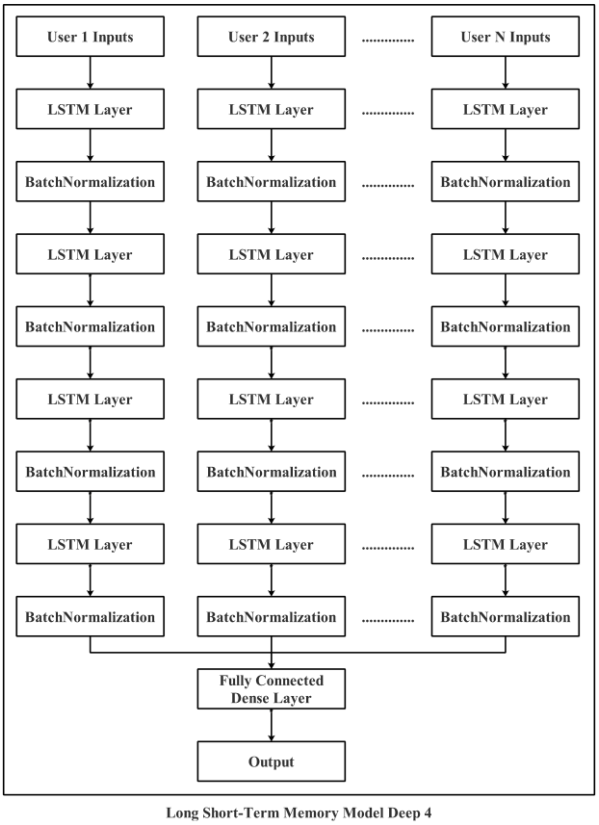

Figure 4. Architecture of user based long short-term memory model at depth 2

Long short-term memory model in deep 4 was shown in the figure 4. This model has contained 1000 input layer, because it was only used for user based approach. The layers of this model was exactly same with the LSTM_2 model. Because this model contained four one directional long short-term memory layer, only difference from LSTM_2 model was the depth of the model. For this reason, the model was named as LSTM_4. This model was used in this study to measure the effect of depth on the performance of model. The depth was only measure for LSTM model, because the initial results showed that the best approach was user based LSTM model for UEBA. Similar to other models, layers created for each user are identical.

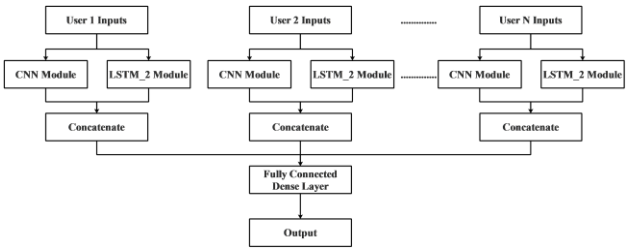

Figure 5. Architecture of user based convolution and long short-term memory model at depth 2

Combination of convolution and long short-term memory model in deep 2, which were used for only user based approach, was shown in figure the figure 5. In this model, input layers, which were generated for each user separately, were connected in parallel to convolution and LSTM modules. The convolution module was exactly same with the CNN model and the LSTM module was exactly same with the LSTM_2 model. At the final stage, modules generated for each user separately were concatenated and they followed by fully connected and classification layer. Similar to other models, layers created for each user are identical. This model was called as CNN_LSTM_2, because LSTM_2 was used to generate LSTM module. Combination of convolution and long short-term memory model in deep 4, which were used for only user based approach were also developed. This model was similar to CNN_LSTM_2 model where only difference was LSTM_4 model was used in order to LSTM_2 model for LSTM modules. Similar to other models, layers created for each user are identical. This model was called as CNN_LSTM_4, because LSTM_4 was used to generate LSTM module.

These models were developed using keras deep learning library of python [37]. In these models, hidden dense layer represents the traditional artificial neural networks. In the convolutional neural network models, two layers, which have different kernel sizes as seven and one, were connected each other serially. Note that, the convolution layers in these models consisted of 1D convolution operation. The horizontal width of the kernel was 29 that is equal to the total number of features. The convolution process was performed between activities performed by the same person on previous days. In the LSTM models return_sequences parameter was set to True. Similar to convolution process, the LSTM layer also revealed the time series relationship by looking at the previous day activities of the same person. In all layers, relu was used as activation function, glorot_uniform with seed 1 was used as weight initializers, softmax was used as activation function at classification layer, Adam was used as optimizer and categorical_crossentropy was used as loss function. During the experiment, the layer specific hyper-parameters of model were optimized using Bayesian optimization techniques. The details about the hyper-parameter optimization process was given in the experiment results section

3. EXPERIMENT RESULTS

In the first phase of the experiment, four datasets, which are training, testing, training for optimization (trainingOpt) and testing for optimization (testingOpt), were generated by dividing the main dataset into parts with respect to user and sample date. Firstly, 30% percent of the main dataset (last samples by date for each user) selected to generate testing dataset and training dataset was generated using the rest of main dataset. After this phase, 25% percent of the training dataset (similar to testing dataset, these are the last samples by date of training dataset for each user) selected to generate testingOpt dataset, trainingOpt dataset was generated using the rest of training dataset. TrainingOpt and testingOpt datasets were used for hyper-parameter optimization phase and training dataset was used to train model with optimum hyper-parameter and testing dataset

was used to assess performance of trained model with optimum hyper-parameter.

3.1. Hyper-Parameter Optimization

Hyper-parameters are one of the most important factor that affect the performance of the deep learning models. Because number of hyper-parameters are high and hyper-parameter space are large, traditional hyper-parameter optimization techniques are not effective for deep learning models [29]–[32]. Because of these reasons, in this study, hyper-parameters of the models were optimized using Bayesian optimization techniques. This method uses a Gaussian process with two aims: Modeling the surrogate function and optimizing the expected probability which is based on improving the existing best solutions through new trials. It assumes that the function values track a multivariate Gaussian distribution. A Gaussian kernel designates the covariance of the function values among the parameters. In each iteration, the next value of a parameter is selected through the acquisition function over the Gaussian prior.

In this study, Bayesian Optimization technique was implemented using skopt library of python language [37]. From this library, gp_minimize function was used with hyper-parameter spaces using following parameters settings: n_cals = 250 and acq_func = "EI". This library operates as a unique parameter space type. The model was optimized according to the mean of true negative rate and accuracy. In this context, according to the hyper-parameter type, three different variable types, which were Real, Categorical and Integer, were used. In these types, Real and Integer needs a low and high value and Categorical needs a hyper-parameter space as in grid search technique. For this purpose, these spaces were generated by determining the hyper-parameters to be optimized separately for each model and hyper-parameters of each model were optimized using trainingOpt and testingOpt datasets. Hyper-parameter value type, space, and optimum value were shown in table 1 for each hyper-parameter of role based model and in table 2 for user based model. In this step, learning rate of the model (lr), number of epoch (epoch), batch size (batch), number of hidden unit in dense layer (n_unit_dense), number of kernel in convolutional layer (n_unit_conv_1) and dimension of output space of LSTM layers (n_unit_lstm_1-4) were optimized, if model need any of them.

Table 1. Hyper-parameters value type, space and optimum value for role based models

Model Name	Parameter Name	Value Type	Parameter Space	Optimum Value
MLP	lr	Real	low = 10^{-10}, high = 10^{-1}	0.059841909
	epoch	Integer	low = 10, high = 1500	1458
	batch	Categorical	{2,4,8,16,32,64,128,256,512,1024,2048}	1024
	n_unit_dense	Integer	low = 20, high = 5000	4076
CNN	lr	Real	low = 10^{-10}, high = 10^{-1}	0.0000004771
	epoch	Integer	low = 10, high = 1500	758
	batch	Categorical	{2,4,8,16,32,64,128,256,512,1024,2048}	2048
	n_unit_dense	Integer	low = 20, high = 5000	77
	n_unit_conv_1	Integer	low = 20, high = 150	101
LSTM_2	lr	Real	low = 10^{-10}, high = 10^{-1}	0.0000022047
	epoch	Integer	low = 10, high = 1500	780
	batch	Categorical	{2,4,8,16,32,64,128,256,512,1024,2048}	512
	n_unit_dense	Integer	low = 20, high = 5000	2767
	n_unit_lstm_1	Integer	low = 10, high = 100	34
	n_unit_lstm_2	Integer	low = 10, high = 100	28

As evident from the table data, an extensive search was conducted for hyper-parameters, particularly real values. The optimum values obtained could be any value within the specified range, unlike grid search. This is notably observed in the lr hyper-parameter. Upon examining the table data, it is noted that the epoch and batch parameters generally have high values. The lr parameter varies from model to model, but it is mostly close to the lower limit.

Table 2. Hyper-parameters value type, space and optimum value for user based models

Model Name	Parameter Name	Value Type	Parameter Space	Optimum Value
CNN	lr	Real	low = 10^{-10}, high = 10^{-1}	0.0000005749
	epoch	Integer	low = 10, high = 1500	1301
	batch	Categorical	{2,4,8,16,32,64,128,256,512,1024,2048}	16

	n_unit_dense	Integer	low = 20, high = 5000	602
	n_unit_conv_1	Integer	low = 20, high = 150	68
LSTM_2	lr	Real	low = 10^{-10}, high = 10^{-1}	0.00000000143
	epoch	Integer	low = 10, high = 1500	167
	batch	Categorical	{2,4,8,16,32,64,128,256,512,1024,2048}	8
	n_unit_dense	Integer	low = 20, high = 5000	1482
	n_unit_lstm_1	Integer	low = 10, high = 100	51
	n_unit_lstm_2	Integer	low = 10, high = 100	37
LSTM_4	lr	Real	low = 10^{-10}, high = 10^{-1}	0.00012994
	epoch	Integer	low = 10, high = 1500	399
	batch	Categorical	{2,4,8,16,32,64,128,256,512,1024,2048}	4
	n_unit_dense	Integer	low = 20, high = 5000	639
	n_unit_lstm_1	Integer	low = 10, high = 100	10
	n_unit_lstm_2	Integer	low = 10, high = 100	25
	n_unit_lstm_3	Integer	low = 10, high = 100	29
	n_unit_lstm_4	Integer	low = 10, high = 100	18
CNN_LSTM_2	lr	Real	low = 10^{-10}, high = 10^{-1}	0.0001284
	epoch	Integer	low = 10, high = 1500	1065
	batch	Categorical	{2,4,8,16,32,64,128,256,512,1024,2048}	2048
	n_unit_dense	Integer	low = 20, high = 5000	617
	n_unit_conv_1	Integer	low = 20, high = 150	148
	n_unit_lstm_1	Integer	low = 10, high = 100	28
	n_unit_lstm_2	Integer	low = 10, high = 100	57
CNN_LSTM_4	lr	Real	low = 10^{-10}, high = 10^{-1}	0.00000002271
	epoch	Integer	low = 10, high = 1500	1500
	batch	Categorical	{2,4,8,16,32,64,128,256,512,1024,2048}	1024
	n_unit_dense	Integer	low = 20, high = 5000	20
	n_unit_conv_1	Integer	low = 20, high = 150	34
	n_unit_lstm_1	Integer	low = 10, high = 100	16
	n_unit_lstm_2	Integer	low = 10, high = 100	10
	n_unit_lstm_3	Integer	low = 10, high = 100	14
	n_unit_lstm_4	Integer	low = 10, high = 100	26

Based on the results observed in the table, unlike role-based models, hyper-parameters in user-based models may exhibit more variation from model to model. Therefore, it cannot be interpreted as being close to the lower or upper limit for any hyper-parameter value. Particularly when examining hyper-parameter distributions with real values such as lr, the benefits of the Bayesian optimization method become evident once again.

3.2. Performance Measures of Model Trained with Optimum Hyper-Parameters

After hyper-parameter optimization process, models were trained using optimum hyper-parameters with training dataset and performance of models were assessed using testing dataset. Two callback functions, which were learning rate callback and early stopping callback, were added to model to improve the performance of our model. In the learning rate callback function, learning rate of the model was divided by two if there is no improvement on model loss for three epochs. Although number of epoch is optimized, sometimes stopping the model early before reaching the epoch number can affect the model performance positively. Because of this reason, the training process was stopped using early stopping callback, if there is no improvement on model loss for six epochs. For this purpose, testingOpt dataset was used as validation dataset during the training phase. Accuracy is the one of the most common performance metrics to measure the performance of the machine learning models. However, the accuracy score can be very misleading for the unbalanced datasets. Because the number of malicious actions are much less than the number of normal actions, our dataset also considered as unbalanced datasets. Because of these reasons accuracy, detection rate (DR), f1-score, false discovery rate (FDR) and negative predictive value (NPV), which were shown in table 3, were computed to assess the performance of our model. The details about these metrics can be found in the information page [39].

Table 3. Performance measures of proposed model that trained using optimum hyper-parameters

Model Name	Model Base	Accuracy	DR	F1-Score	FDR	NPV
MLP	Role	86.27%	90.67%	92.25%	13.76%	95.09%
CNN	Role	83.07%	86.05%	90.37%	16.95%	89.05%
LSTM_2	Role	79.84%	89.88%	88.40%	20.24%	**99.89%**
CNN	User	98.86%	97.17%	99.19%	1.122%	95.47%
LSTM_2	User	**99.27%**	98.07%	**99.44%**	**0.724%**	96.86%
LSTM_4	User	99.18%	**98.89%**	99.38%	0.820%	98.60%
CNN_LSTM_2	User	98.85%	98.21%	99.18%	1.144%	97.56%
CNN_LSTM_4	User	98.94%	98.25%	99.24%	1.057%	97.56%

According to these results, it is desired that all scores except FDR are high and FDR is low. When the experiment results are examined, it is seen that user based LSTM model in deep two obtained the best scores for all metrics except NPV and DR. Role based LSTM model in deep two obtained the best score for NPV metric and user based LSTM model in deep 4 obtained the best score for DR metric. When these results are interpreted, it is understood that solo LSTM models were the most successful predictor for UEBA and the effect of increasing the depth on the performance was negative. In addition to this, user based models have been much more successful than role based models. Considering all these situations, it is thought that the most suitable model for a UEBA platform is the user based LSTM model. The bar chart for each metric is presented in figure 6 to better visualize the performance between models. In this figure, bar charts were shown for each performance metrics separately and different colors were assigned to each model separately. The model shown with the same color in bar charts of all performance metrics.

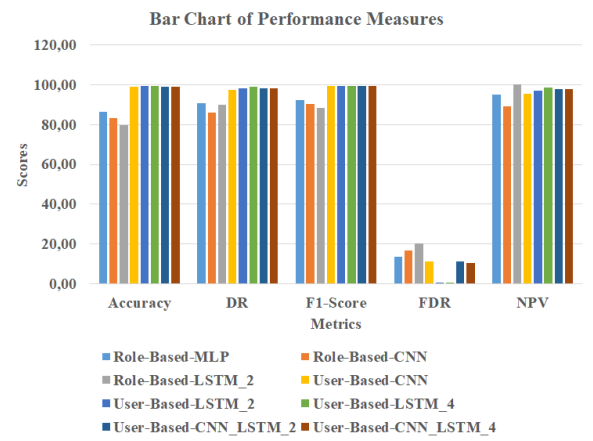

Figure 6. Bar chart of each model based on performance metrics

3.3. Comparison with the State-of-the-art

In order to better measure the contribution of the proposed models to the literature, it has great importance to compare them with the existing studies in the literature. Because of this reason, our model was compared with the four models, which were proposed by Le and Zincir-Heywood [16], Al-Shehari and Alsowail [21], Nasir et al. [22] and Al-Mhiqani et al. [25], from the literature. Table 4 shows the comparison result of these models with our model with respect to accuracy, DR, FDR or F1-score if they exist in state-of-the-art models. For the state-of-the-art comparison our LSTM model with deep 2 was used because it was the model with the best performance. These state-of-the-art models were selected because they were also developed based on user and they trained model on version 4.2 of CERT dataset.

Table 4. Comparison between proposed and state-of-the-art methods on version 4.2 of CERT dataset

Model	Accuracy	DR	F1-Score	FDR
Model of Le and Zincir-Heywood [16]	98.33%	69.51%	--------	**0.350%**
Model of Al-Shehari and Alsowail [21]	--------	--------	99.00%	--------
Model of Nasir et. al. [22]	90.00%	--------	94.00%	--------
Model of Al-Mhiqani et al. [25]	97.00%		84.00%	2.88%
Proposed Model	**99.27%**	**98.07%**	**99.44%**	0.724%

Based on these results, our model outperformed the state-of-the-art model for accuracy, DR and F1-score metrics. However, autoencoder based model proposed by Le and Zincir-Heywood [16] were obtained better FDR score than our model. Considering the proportion taken for FDR, it is seen that both models achieve very low results. Considering that the proposed model achieves much better results especially for the DR score, it is thought that our

model is better than model proposed by Le and Zincir-Heywood [16] according to the average metric results.

3.4. Two-tailed Z-test

Based on the experiment results, it is concluded that user based models have better results than the role based models and the proposed model out-performed the state-of-the-art. In this phase, it was checked whether these improvements were significant. For this purpose, two-tailed z test was applied on our experiment results and also our experiment results were compared with the state-of-the-art based on two-tailed z test score. The details about two-tailed z test can be found at information page [40]. Table 5 shows the p values between the best model (LSTM model in deep two) and the other model and table 6 shows the p values between LSTM_2 model and state-of-the-art based on comparison metrics. A p-value of less than 0.01 indicates that the proposed model is 99% reliable, according to the two-tailed Z test, and statistically outperforms the models in the literature. Thus, a statistical comparison of the model with the literature can be made by examining the results in Table 6.

Table 5. P-values between LSTM_2 model and the other model calculated with two-tailed Z test.

Model Name	**Model Base**	**P-Value of Accuracy**	**P-Value of DR**	**P-Value of F1-Score**	**P-Value of FDR**	**P-Value of NPV**
MLP	Role	0. 00001	0. 00001	0. 00001	0. 00001	0. 00001
CNN	Role	0. 00001	0. 00001	0. 00001	0. 00001	0. 00001
LSTM_2	Role	0. 00001	0. 00001	0. 00001	0. 00001	**0. 00001**
CNN	User	0. 00001	0. 00001	0. 00001	0. 00001	0. 00001
LSTM_4	User	0. 00001	**0. 00001**	0. 00001	0. 00001	0. 00001
CNN_LSTM_2	User	0. 00001	0. 00001	0. 00001	0. 00001	0. 00001
CNN_LSTM_4	User	0. 00001	0. 00001	0. 00001	0. 00001	0. 00001

Table 6. P-values between LSTM_2 model and state-of-the-art model with two-tailed Z test.

Model	**Accuracy**	**DR**	**F1-Score**	**FDR**
Model of Le and Zincir-Heywood [16]	0. 00001	0. 00001	--------	**0. 00001**
Model of Al-Shehari and Alsowail [21]	--------	--------	0. 00001	--------
Model of Nasir et. al. [22]	0. 00001	--------	0. 00001	--------
Model of Al-Mhiqani et al. [25]	0. 00001	--------	0. 00001	0. 00001

Considering these results, it is seen that all improvements are statistically significant according to the two-tailed Z test at $p < 0.01$. The bold p-values represent the scores where the LSTM_2 model was worse than the related model. According to the results of two-tailed Z test, the model proposed by Le and Zincir-Heywood obtained statistically significant better results than our model for false discovery rate results. However, when models are considered based on all other metrics, our model seems to be superior. Especially in the detection rate score, it is observed that our model is significantly better than the model of Le and Zincir-Heywood [16].

4. CONCLUSION AND FUTURE WORK

In this study, several deep learning models, which were developed using fully connected layer, convolutional neural network and long short-term memory network, were proposed for user and entity behavior analysis. Our models used two different approaches: role based and user based. In user based models, the dataset is split so that each piece contains the data of one user. In role based approach, users with the same role were included in the same partition. Experiment analysis showed that user based models are better than the role based models.

According to the analysis results, the MLP model achieved the highest accuracy among role-based models, with a rate of 86.27%. In addition, its DR, F1-Score, FDR, and NPV scores were measured at 90.67%, 92.25%, 13.76%, and 95.09%, respectively. Surprisingly, the accuracy of this model was even lower than that of the CNN_LSTM_2 model, which has the lowest accuracy among user-based models. The CNN_LSTM_2 model obtained values of 98.21%, 99.18%, 1.144%, and 97.56% for DR, F1-Score, FDR, and NPV scores, respectively, outperforming the role-based MLP model in all metric scores. Considering these results, it is concluded that developing an insider threat system for an institution using a user-based approach is more effective in terms of performance metric scores. However, it's worth noting that user-based models may face challenges in corporate settings due to potential changes in an individual's authority or department. Such changes can lead to shifts in employee activities, potentially affecting the model's predictive accuracy. To address this issue, fine-tuning of the model is necessary, which may result in a decrease in model efficiency. In light of these considerations, it is concluded that role-based models are more compatible with corporate organizations. Therefore, future studies should focus on improving the accuracy of role-based models.

The best proposed model, which uses two serial connected LSTM layers, outperformed the state-of-the-art models and the improvement was statistically significant according to the two-tailed Z test at $p < 0.01$. Although our model obtained 99.27% accuracy on version 4.2 of CERT dataset, it is thought that the model should be improved in other metric scores such as negative predictive value and detection rate. Because of this reason, in the future work, more complex deep learning models will be developed using different approaches such as user, role, department and team based. In addition to this, several feature selection techniques will be applied on CERT dataset. In this way, best deep learning structures for UEBA and the features that can best identify malicious action will be determined. A new UEBA approach will be proposed using this information and a model will be proposed using the data of our institution operating in the field of technology. For this purpose, daily activities of employees will be collected with a SIEM application used in our company. In line with the determined scenarios, these activities will be labeled and a specific dataset will be created for our institution. The proposed deep learning models will be also applied for this dataset and the results will be compared with the performance measures of CERT dataset. After all these analyzes, we are planning to integrate proposed model to our SIEM systems to analyze real time performance of UEBA systems

KAYNAKLAR (REFERENCES)

[1] N. R. Mosteanu, "Artificial Intelligence and Cyber Security – Face To Face With Cyber Attack – A Maltese Case Of Risk Management Approach", *Ecoforum Journal*, 9(2), 2020.

[2] D. Ghelani, **Cyber Security, Cyber Threats, Implications and Future Perspectives: A Review**, Authorea Preprints, 2022

[3] Y. Hashem, H. Takabi, R. Dantu, and R. Nielsen, "A Multi-Modal Neuro-Physiological Study of Malicious Insider Threats", **International Workshop on Managing Insider Security Threats**, New York, NY, USA, 33-44, October 2017.

[4] M. Dosh, "Detecting insider threat within institutions using CERT dataset and different ML techniques", *Periodicals of Engineering and Natural Sciences*, 9(2), 873-884, 2021.

[5] Insider Threat Test Dataset, https://kilthub.cmu.edu/articles/dataset/Insider_Threat_Test_Dataset/12841247/1, 21.01.2024.

[6] W. R. Claycomb and A. Nicoll, "Insider Threats to Cloud Computing: Directions for New Research Challenges", **36th Annual Computer Software and Applications Conference**, İzmir, Turkey, 387,394, July 2012.

[7] X. Xiangyu et al., "Method and System for Detecting Anomalous User Behaviors: An Ensemble Approach", **30th International Conference on Software Engineering and Knowledge Engineering**, San Francisco, California, USA , 263-307, July 2018.

[8] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, **Deep Learning for Unsupervised Insider Threat Detection in Structured Cybersecurity Data Streams**, arXiv, 2017.

[9] L. Lin, S. Zhong, C. Jia, and K. Chen, "Insider Threat Detection Based on Deep Belief Network Feature Representation", **International Conference on Green Informatics**, Fuzhou, China, 54-59, August 2017.

[10] F. Yuan, Y. Cao, Y. Shang, Y. Liu, J. Tan, and B. Fang, "Insider Threat Detection with Deep Neural Network", **Computational Science**,Wuzi, China, 43-54, 2018.

[11] O. Lo, W. J. Buchanan, P. Griffiths, and R. Macfarlane, "Distance Measurement Methods for Improved Insider Threat Detection", *Security and Communication Networks*, 2018(e5906368), 1-18, 2018.

[12] D. C. Le and A. N. Zincir-Heywood, "Evaluating Insider Threat Detection Workflow Using Supervised and Unsupervised Learning", **IEEE Security and Privacy Workshops**, San Francisco, CA, USA, 270-275, May 2018.

[13] O. Igbe and T. Saadawi, "Insider Threat Detection using an Artificial Immune system Algorithm", **9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference**, New York, USA, 297-302, November 2018.

[14] A. J. Hall, N. Pitropakis, W. J. Buchanan, and N. Moradpoor, "Predicting Malicious Insider Threat Scenarios Using Organizational Data and a Heterogeneous Stack-Classifier", **IEEE International Conference on Big Data**, Seattle, WA, USA, 5034-5039, December 2018.

[15] M. Aldairi, L. Karimi, and J. Joshi, "A Trust Aware Unsupervised Learning Approach for Insider Threat Detection", **IEEE 20th International Conference on Information Reuse and Integration for Data Science**, Los Angeles, California, USA, 89-98, July 2019.

[16] D. C. Le and N. Zincir-Heywood, "Exploring anomalous behaviour detection and classification for insider threat identification", *International Journal of Network Management*, 31(4), 2021.

[17] D. C. Le and A. Nur Zincir-Heywood, "Machine learning based Insider Threat Modelling and Detection", IFIP/IEEE Symposium on Integrated Network and Service Management, Washington DC, USA, 1-6, April.

[18] M. Nasser Al-mhiqani, R. Ahmad, Z. Zainal Abidin, W. Yassin, A. Hassan, and A. Natasha Mohammad, "New insider threat detection method based on recurrent neural networks", *Indonesian Journal of Electrical Engineering and Computer Science*, 17(3), 1474, 2020.

[19] B. Sharma, P. Pokharel, and B. Joshi, "User Behavior Analytics for Anomaly Detection Using LSTM Autoencoder - Insider Threat Detection", **11th International Conference on Advances in Information Technology**, New York, USA, 1-9, July 2020.

[20] Z. Tian, C. Luo, H. Lu, S. Su, Y. Sun, and M. Zhang, "User and Entity Behavior Analysis under Urban Big Data", *ACM Transactions on Data Science*, 1(3), 1-16, 2020.

[21] T. Al-Shehari and R. A. Alsowail, "An Insider Data Leakage Detection Using One-Hot Encoding, Synthetic Minority Oversampling and Machine Learning Techniques", *Entropy*, *23(10)*, no. 10, 2021.

[22] R. Nasir, M. Afzal, R. Latif, and W. Iqbal, "Behavioral Based Insider Threat Detection Using Deep Learning", *IEEE Access*, *9(1)*, 143266–143274, 2021.

[23] D. Sun, M. Liu, M. Li, Z. Shi, P. Liu, and X. Wang, "DeepMIT: A Novel Malicious Insider Threat Detection Framework based on Recurrent Neural Network", **24th International Conference on Computer Supported Cooperative Work in Design**, Dalian, China, 335-341, May 2021.

[24] E. Pantelidis, G. Bendiab, S. Shiaeles, and N. Kolokotronis, "Insider Threat Detection using Deep Autoencoder and Variational Autoencoder Neural Networks", **IEEE International Conference on Cyber Security and Resilience**, Rhodes, Greece, 129-134, July 2021.

[25] M. N. Al-Mhiqani et al., "A new intelligent multilayer framework for insider threat detection", *Computers & Electrical Engineering*, 97(1), 107597, January 2022.

[26] M. AlSlaiman, M. I. Salman, M. M. Saleh, and B. Wang, "Enhancing false negative and positive rates for efficient insider threat detection", *Computers & Security*, 126(1), 103066, March 2023.

[27] D. Li, L. Yang, H. Zhang, X. Wang, and L. Ma, "Memory-Augmented Insider Threat Detection with Temporal-Spatial Fusion", *Security and Communication Networks*, 2022(1), e6418420, 2022.

[27] T. Karayel, A. Akbıyık, "A Global Perspective of Cybersecurity Research: Publication Trends and Research Directions", *Journal of Information Technologies*, 16(3), 221 – 235, 2023.

[29] Y. Gormez, Z. Aydin, R. Karademir, and V. C. Gungor, "A deep learning approach with Bayesian optimization and ensemble classifiers for detecting denial of service attacks", *International Journal of Communication Systems*, 33(11), e4401, 2020.

[30] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms", **Advances in Neural Information Processing Systems**, Nevada, USA, 2012.

[31] A. Salama, A. E. Hassanien, and A. Fahmy, "Sheep Identification Using a Hybrid Deep Learning and Bayesian Optimization Approach", *IEEE Access*, 7(1), 31681–31687, 2019.

[32] J. Snoek et al., "Scalable Bayesian Optimization Using Deep Neural Networks," **32nd International Conference on Machine Learning**, Lille, France, 2171-2180, Jun 2015.

[33] H. Kaur, H. S. Pannu, and A. K. Malhi, "A Systematic Review on Imbalanced Data Challenges in Machine Learning: Applications and Solutions", *ACM Computing Survey*, 52(4), 1-36, August 2019.

[34] Big Five personality traits: https://en.wikipedia.org/w/index.php?title=Big_Five_personality_traits&oldid=1114671408, 21.01.2024.

[35] Y. Görmez, H. Arslan, Y. E. Işik, and İ. E. Dadaş, "A User and Entity Behavior Analysis for SIEM Systems: Preprocessing of The Computer Emergency and Response Team Dataset," *Journal Soft Computing*, 4(1), 2023.

[36] Arge-Preprocessing-CERT: https://github.com/Detaysoft/Arge-Preprocessing-CERT, 21.01.2024

[37] Keras: the Python deep learning API: https://keras.io/, 21.01.2024.

[38] scikit-optimize:https://scikit-optimize.github.io/stable/, 21.01.2024.

[39] Precision and recall: https://en.wikipedia.org/w/index.php?title=Precision_and_recall&oldid=1122267443, 21.01.2024.

[40] Z Score Calculator for 2 Poulation Proportions, https://www.socscistatistics.com/tests/ztest/default2.aspx, 21.01.2024.