Detecting Cybersecurity Threats Using AI Network

Kariveda Venkata Sri Ram

Reasearch Student, CloudAmigo

Abstract:
In the field of cybersecurity, one of the most important tasks is to find a way to automatically and very effectively find online risks. This study shows how artificial intelligence and artificial neural networks can be used in a new way to find online threats. The suggested method turns a large number of security events into individual event profiles and uses a deep learning-based detecting method to greatly improve the ability to find cyber threats. To do this, an AI-SIEM (Artificial Intelligence Security Information and Event Management) system has been created. This system combines event profiling for data preprocessing with different artificial neural network techniques, such as Fully Connected Neural Networks (FCNN), Convolutional Neural Networks (CNN), and Long Short-Term Memory (LSTM) networks. The main goal of this system is to tell the difference between real alerts and false ones, so that security experts can react quickly to new online risks. The writers did a lot of tests using real-world data and two standard datasets (NSLKDD and CICIDS2017) to see how well the system worked. Five common machine-learning methods (Support Vector Machines, k-Nearest Neighbors, Random Forests, Naive Bayes, and Decision Trees) were used in tests to compare how well they worked with known methods. The study's results show that the suggested methods work well for detecting network intrusions, even in real-world situations, where they do a better job than traditional machine-learning approaches.

Keywords: Cybersecurity, Cyber-Threat Detection, Artificial Intelligence, Artificial Neural Networks, Deep Learning, AI-SIEM System, Network Intrusion Detection, Machine Learning, True Positive Alerts, False Positive Alerts

I. INTRODUCTION

The use of artificial intelligence (AI) techniques has made learning-based methods for finding cyberattacks much more effective. Several studies have found that these methods work well. But the fact that cyber risks are always changing makes it very hard to protect IT systems from bad things happening in networks. Because network attacks and bad behavior happen all the time, it's very important to come up with effective defense mechanisms and security solutions [1, 2, 3, 4].

Usually, there are two main methods used to find cyber risks and network attacks. An Intrusion Prevention System (IPS) is set up in the business network. Most of the time, signature-based methods are used to look at the protocols and flows of the network. It sends attack reports, which are called security events, to another system, like a Security Information and Event Management (SIEM) system. SIEM systems are widely used to collect and manage the reports that IPSs send out. SIEM stands out as the most common and effective method for examining security events and logs that have been gathered [5]. Security experts are very important when it comes to looking into strange alerts. They do this by following rules and limits that have already been set and by studying links between events based on what they know about attack trends.

The use of learning-based methods to find where an attack happened in a big amount of data has been helpful for analysts who need to quickly evaluate many events. Analyst-driven solutions and machine learning-driven solutions are the two main types of information security solutions [10]. Analyst-driven solutions are based on rules that security experts, called analysts, have already set up. On the other hand, methods based on machine learning are designed to find unusual or rare patterns, which makes it easier to find new cyber dangers [10]. Even though learning-based techniques are useful for finding hacks in systems and networks, there are some problems with the methods that are already in place.

First of all, learning-based methods of spotting need tagged data to train and test models. It is often hard to get a large enough amount of named data for correct model training. Many commercial SIEM systems don't keep labeled data that can be used for supervised learning models, even though labeled data is important for these methods [10].

Second, many of the ways of learning that are suggested by research studies are not easy to use in the real world. Most network protection systems don't have these functions, which makes it hard to use them in real life. In recent intrusion detection study, automation through deep learning technologies has been looked into. Well-known datasets like NSLKDD [11], CICIDS2017 [12], and Kyoto-Honeypot [13] have been used to test performance. But because these standard samples only have a small number of features, they may not work well in the real world. To get around these problems, learning models should be tested with data taken from the real world.

II. DOMAIN INTRODUCTION

A Deep Neural Network (DNN) is a sophisticated type of neural network characterized by its complexity, specifically having more than two layers. These neural networks leverage advanced mathematical models to process data in intricate ways. Neural networks, in general, are technology designed to mimic human brain activities, particularly pattern recognition and the flow of input through layers of simulated neural connections.

Deep neural networks are typically defined as networks comprising an input layer, an output layer, and at least one hidden layer in between. Each layer serves a unique purpose in a process often referred to as "feature hierarchy," where data is sorted and organized hierarchically. Deep neural networks excel in handling unlabeled or unstructured data. The term "deep learning" is also commonly used to describe these networks, signifying a subset of machine learning that leverages artificial intelligence to classify and organize information in more complex ways than traditional input/output protocols.

Benefits of Deep Neural Networks

Neural networks intentionally incorporate randomness into their design to enhance their ability to learn and approximate complex functions effectively. Randomness is strategically employed in these machine learning algorithms because it often leads to improved performance.

One of the primary uses of randomness in neural networks is in the initial setup of network weights. While randomness can be applied in various aspects, here are a few key areas where it plays a role:

1. **Randomness in Initialization**: This involves the random assignment of weights in the neural network, which helps prevent the network from getting stuck in suboptimal solutions during training.

2. **Randomness in Regularization:** Techniques like dropout introduce randomness by temporarily removing some neurons during training, preventing overfitting and improving generalization.
3. **Randomness in Layers:** In tasks like natural language processing, randomness is introduced in layers like word embedding to capture the rich semantic relationships between words.

III. RELATED STUDY

This literature review delves into the extensive body of research surrounding the application of deep learning methods for intrusion detection. It provides a comprehensive overview of the key studies and developments in this field, examining the methodologies, models, and datasets used, as well as the performance and limitations of deep learning-based intrusion detection systems. By synthesizing the findings of these studies, this review aims to shed light on the current state of research, identify trends and challenges, and offer insights into the future directions of leveraging deep learning for the protection of digital ecosystems against cyber threats. Through this exploration, we seek to contribute to a deeper understanding of the efficacy and potential of deep learning techniques in the critical domain of intrusion detection.

Naseer et al. [1] introduced and trained intrusion detection models employing different deep neural network architectures, including Convolutional Neural Networks (CNNs), Autoencoders, and Recurrent Neural Networks (RNNs). Their models were trained on the NSLKDD training dataset and evaluated on test datasets provided by NSLKDD. Notably, the DCNN and LSTM models achieved commendable performance, with 85% and 89% accuracy, respectively, on the test dataset.

B. Zhang et al. [2] categorized network intrusion detection methods into two types: direct methods employing a single algorithm and combination methods that combine multiple approaches. They proposed a novel detection model based on a directed acyclic graph (DAG) and a belief rule base (BRB). Their results revealed that the DAG-BRB combination model outperformed conventional detection models, demonstrating a higher detection rate when using the KDD 99 dataset.

Wang et al. [3] put forth a hierarchical spatial and temporal features-based intrusion detection system (HAST-IDS), which autonomously learns network traffic features. The approach involves initially learning spatial features using deep Convolutional Neural Networks (CNNs) and subsequently incorporating temporal features via Long Short-Term Memory (LSTM) networks. Experiments were conducted using datasets provided by DARPA and ISCX.

Vinayakumar et al. [15] developed a hybrid intrusion detection system capable of analyzing both network and host-level activities. They harnessed distributed deep learning models, employing Deep Neural Networks (DNNs), to process and analyze extensive real-time data. The selection of the DNN model was informed by comprehensive evaluations against classical machine learning classifiers using benchmark IDS datasets, such as NSLKDD and UNSW-NB15.

Khan et al. [38] introduced an innovative two-stage deep learning model, built upon a stacked auto-encoder with a softmax classifier, designed for efficient network intrusion detection. Extensive experiments were conducted on two widely used public datasets: the benchmark KDD99 and UNSW-NB15 datasets. Remarkably, this study achieved remarkable results, with accuracy rates reaching up to 99.9% for the KDD99 dataset and 89.1% for the UNSW-NB15 dataset.

Liao et al. [39] proposed a novel algorithm based on the k-Nearest Neighbors (k-NN) classifier method using Term Frequency-Inverse Document Frequency (TF-IDF) for modeling program behavior in intrusion detection, particularly concerning system calls. This approach utilizes the frequencies of system calls to

characterize program behavior. They employ text categorization techniques, like TF-IDF, to transform system call data into vectors and calculate the similarity between program system call activities. The authors reported that the TF-IDF-based k-NN classifier demonstrates promise in the domain of intrusion detection, particularly in the context of malware detection.
These studies collectively highlight the diverse applications of deep learning techniques in intrusion detection, showcasing impressive results across different datasets and model architectures.

IV. METHODOLOGY

a. Existing System

Since there is no wait staff in an automated café, it is difficult for the management to gauge how well the concept and the cuisine are received. Existing rating systems, such as Google and Trip Advisor, only partially address this problem since they only cover a subset of the client's findings. A small percentage of the customers who review the cafe on independent websites really use these rating systems. Clients who leave feeling hopeless or dissatisfied are the primary target audience here.

b. Proposed System

All consumers need to be encouraged to leave a review in order to address the aforementioned issue. This article presents a strategy for a restaurant rating system that encourages as many customers as possible to rate their experience after dining there. The system's ratings are based on facial expression recognition via the use of pre-trained convolutional neural network (CNN) models, making it suitable for usage in unattended dining establishments. The consumer may take a photo of his face while rating the cuisine to indicate how he feels about it. There is far less data and no first-hand accounts of user experience than in a text-based rating

c. System Architecture

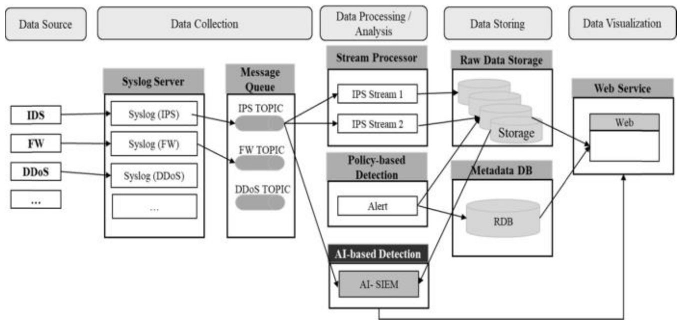


Figure 1: System Design

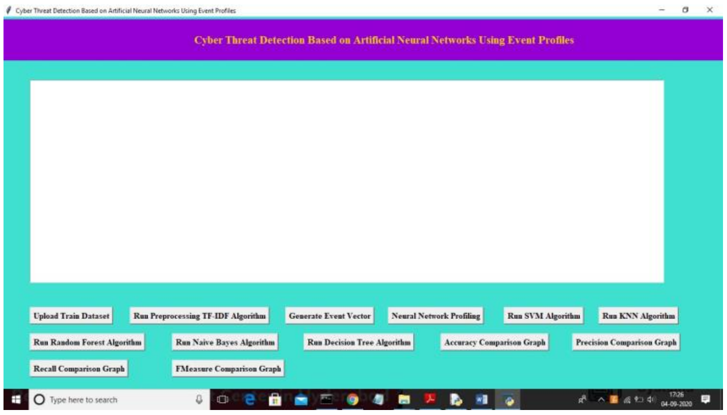
d. MODULES

The proposed algorithms consist of the following modules:
Data Parsing: In this module, the input dataset is processed to create a raw data event model.
TF-IDF: This module is used to convert the raw data into an event vector that contains normal and attack signatures.
Event Profiling Stage: The processed data is divided into training and testing sets based on profiling events.
Deep Learning Neural Network Model: Within this module, Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) algorithms are applied to both the training and testing data to generate a predictive model. The generated trained model is then applied to the test data to calculate prediction scores, Recall, Precision, and FMEA (Failure Modes and Effects Analysis) score. An algorithm that learns

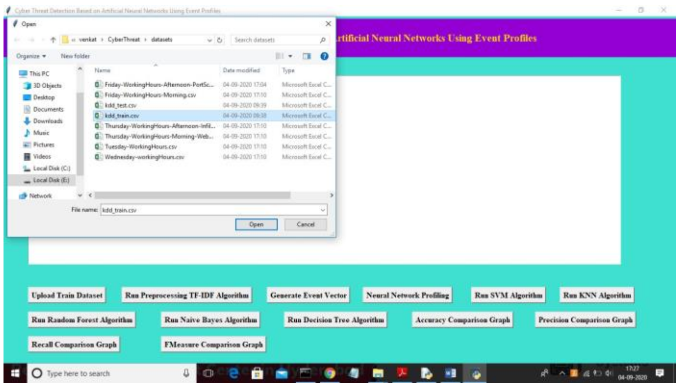
effectively will yield improved accuracy results, and this model is selected for deployment on a real system for attack detection.

Module Description

To construct the complete prediction model, it is imperative to execute all the required modules in a step-by-step manner. To initiate the project, simply double-click on the 'run.bat' file, which will open the screen as depicted below.


Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles
Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles
Upload Train Dataset
Run Preprocessing TF-IDF Algorithm
Generate Event Vector
Neural Network Profiling
Run SVM Algorithm
Run KNN Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Accuracy Comparison Graph
Precision Comparison Graph
Recall Comparison Graph
FMeasure Comparison Graph
Type here to search

To upload a dataset, use the 'Upload Train Dataset' button on the aforementioned interface.


Open

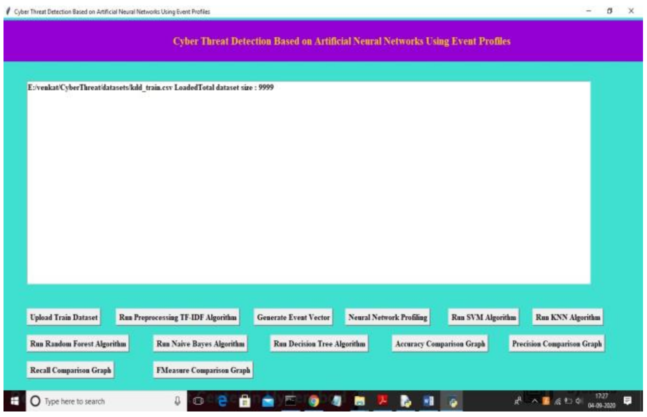
« venkat › CyberThreat › datasets

Name	Date modified	Type
Friday-WorkingHours-Afternoon-PortSc...	04-09-2020 17:04	Microsoft Excel C...
Friday-WorkingHours-Morning.csv	04-09-2020 17:10	Microsoft Excel C...
kdd_test.csv	04-09-2020 09:39	Microsoft Excel C...
kdd_train.csv	04-09-2020 09:38	Microsoft Excel C...
Thursday-WorkingHours-Afternoon-Infil...	04-09-2020 17:10	Microsoft Excel C...
Thursday-WorkingHours-Morning-Web...	04-09-2020 17:10	Microsoft Excel C...
Tuesday-WorkingHours.csv	04-09-2020 17:10	Microsoft Excel C...
Wednesday-workingHours.csv	04-09-2020 17:10	Microsoft Excel C...

File name: kdd_train.csv

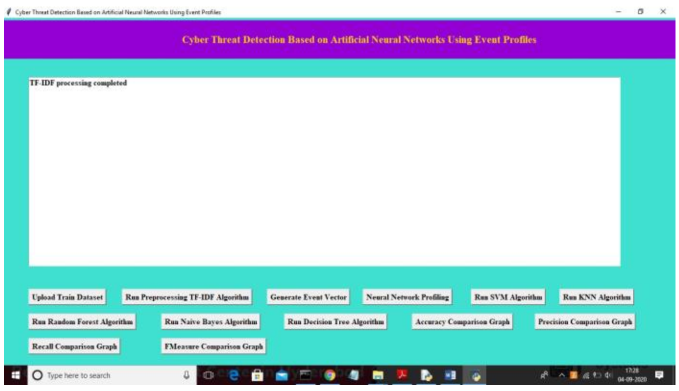
After selecting the 'kdd_train.csv' dataset in the previous screen, the next screen will appear.

e. The TF-IDF Algorithm


E:/venkat/CyberThreat/datasets/kdd_train.csv LoadedTotal dataset size : 9999

Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles
Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles
Upload Train Dataset
Run Preprocessing TF-IDF Algorithm
Generate Event Vector
Neural Network Profiling
Run SVM Algorithm
Run KNN Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Accuracy Comparison Graph
Precision Comparison Graph
Recall Comparison Graph
FMeasure Comparison Graph
Type here to search

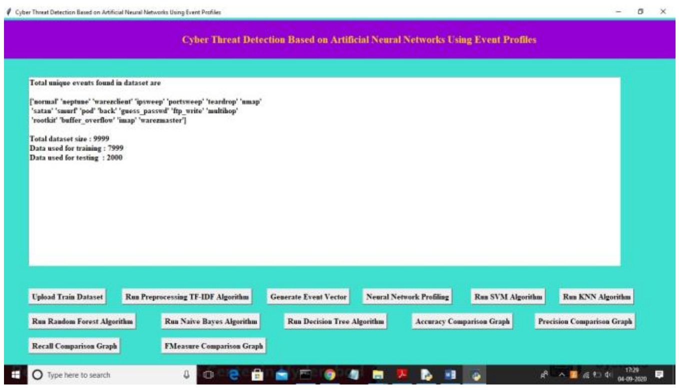
To transform the raw dataset into TF-IDF values, as seen on the previous screen, click the 'Run Preprocessing TF-IDF Algorithm' button.


Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles
Upload Train Dataset
Run Preprocessing TF-IDF Algorithm
Generate Event Vector
Neural Network Profiling
Run SVM Algorithm
Run KNN Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Accuracy Comparison Graph
Precision Comparison Graph
Recall Comparison Graph
FMeasure Comparison Graph

```
TF-IDF processing completed
```

Select the 'Generate Event Vector' button on the previous page to generate a vector from TF-IDF including various events.

f. Generate Even Vector


Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles
Upload Train Dataset
Run Preprocessing TF-IDF Algorithm
Generate Event Vector
Neural Network Profiling
Run SVM Algorithm
Run KNN Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Accuracy Comparison Graph
Precision Comparison Graph
Recall Comparison Graph
FMeasure Comparison Graph

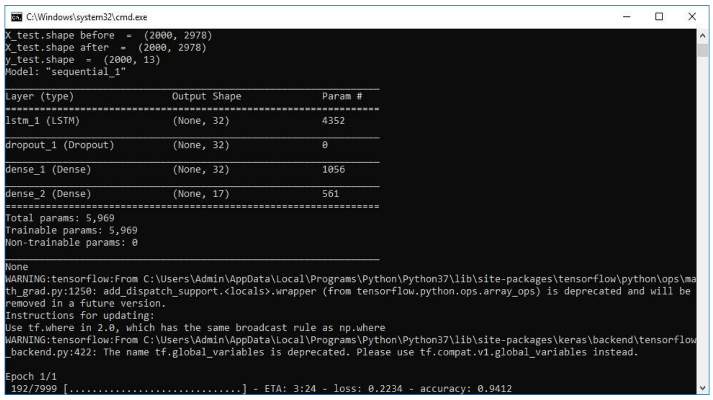
```
Total unique events found in dataset are

['normal' 'neptune' 'warezclient' 'ipsweep' 'portsweep' 'teardrop' 'nmap'
 'satan' 'smurf' 'pod' 'back' 'guess_passwd' 'ftp_write' 'multihop'
 'rootkit' 'buffer_overflow' 'imap' 'warezmaster']

Total dataset size : 9999
Data used for training : 7999
Data used for testing  : 2000
```

You can see the complete amount of the dataset and how the application is using 80% of it for training (7999 records) and utilizing 20% for testing (2000 records) in the window to the right. After getting your datasets ready for training and testing, you may use the 'Neural Network Profiling' button to create an LSTM or CNN model.

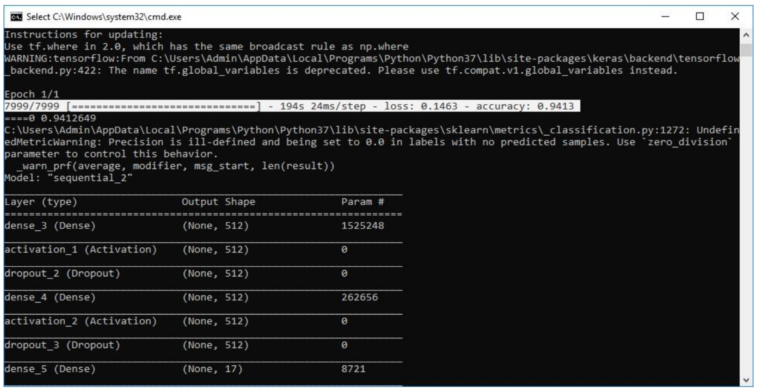
g. Neural Network Profring



```
X_test.shape before  =  (2000, 2978)
X_test.shape after  =  (2000, 2978)
y_test.shape  =  (2000, 13)
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 32)                4352
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                1056
_________________________________________________________________
dense_2 (Dense)              (None, 17)                561
=================================================================
Total params: 5,969
Trainable params: 5,969
Non-trainable params: 0
_________________________________________________________________
None
WARNING:tensorflow:From C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\ops\ma
th_grad.py:1250: add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be
removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow
_backend.py:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

Epoch 1/1
 192/7999 [..............................] - ETA: 3:24 - loss: 0.2234 - accuracy: 0.9412
```

The accompanying screen depicts the generation and epoch running of an LSTM model with an initial accuracy of 0.94. It may take some time for the LSTM and CNN training processes to finish before the whole dataset can be run. There are 7999 records in this dataset, and LSTM will loop over all of them to apply filters and construct the model.

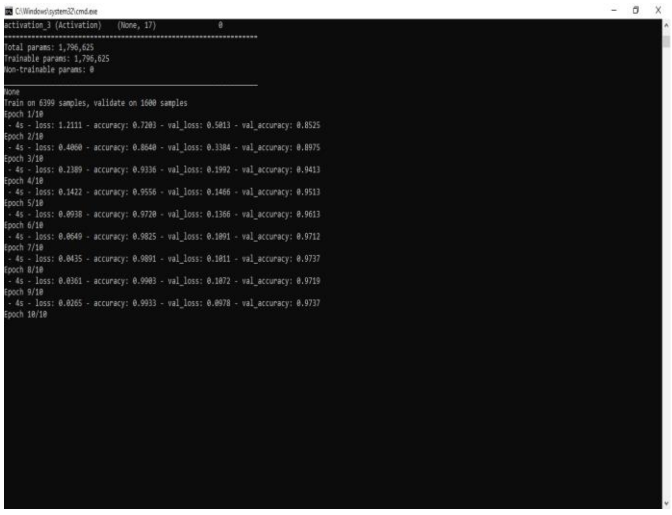


```
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow
_backend.py:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

Epoch 1/1
7999/7999 [==============================] - 194s 24ms/step - loss: 0.1463 - accuracy: 0.9413
====0 0.9412649
C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\metrics\_classification.py:1272: Undefin
edMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 512)               1525248
_________________________________________________________________
activation_1 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               262656
_________________________________________________________________
activation_2 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_5 (Dense)              (None, 17)                8721
```

Here, we see that the LSTM model has finished all of its iterations and that the CNN model has begun running.

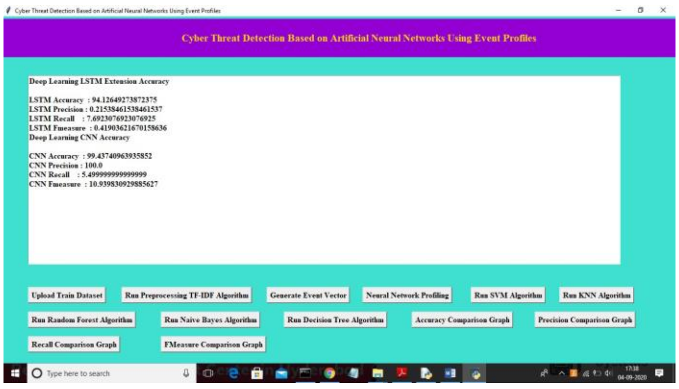
h.SVM Algorithm



```
activation_3 (Activation)    (None, 17)                0
=================================================================
Total params: 1,796,625
Trainable params: 1,796,625
Non-trainable params: 0
_________________________________________________________________
None
Train on 6399 samples, validate on 1600 samples
Epoch 1/10
 - 4s - loss: 1.2111 - accuracy: 0.7203 - val_loss: 0.5013 - val_accuracy: 0.8525
Epoch 2/10
 - 4s - loss: 0.4060 - accuracy: 0.8640 - val_loss: 0.3384 - val_accuracy: 0.8975
Epoch 3/10
 - 4s - loss: 0.2389 - accuracy: 0.9336 - val_loss: 0.1992 - val_accuracy: 0.9413
Epoch 4/10
 - 4s - loss: 0.1422 - accuracy: 0.9556 - val_loss: 0.1466 - val_accuracy: 0.9513
Epoch 5/10
 - 4s - loss: 0.0938 - accuracy: 0.9720 - val_loss: 0.1366 - val_accuracy: 0.9613
Epoch 6/10
 - 4s - loss: 0.0649 - accuracy: 0.9825 - val_loss: 0.1091 - val_accuracy: 0.9712
Epoch 7/10
 - 4s - loss: 0.0435 - accuracy: 0.9891 - val_loss: 0.1011 - val_accuracy: 0.9737
Epoch 8/10
 - 4s - loss: 0.0361 - accuracy: 0.9903 - val_loss: 0.1072 - val_accuracy: 0.9719
Epoch 9/10
 - 4s - loss: 0.0265 - accuracy: 0.9933 - val_loss: 0.0978 - val_accuracy: 0.9737
Epoch 10/10
```

Even while CNN's initial accuracy is just 0.72 on this screen, after 10 cycles of filtering and training, it reaches an impressive 0.99, which when multiplied by 100 yields an impressive 99%. So, as you can see in the accompanying GUI screen, CNN is providing more accuracy than LSTM.

1. KNN Algorithm

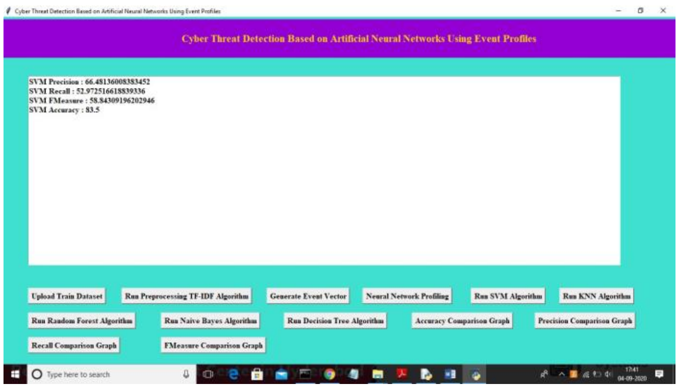

Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles
Upload Train Dataset
Run Preprocessing TF-IDF Algorithm
Generate Event Vector
Neural Network Profiling
Run SVM Algorithm
Run KNN Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Accuracy Comparison Graph
Precision Comparison Graph
Recall Comparison Graph
FMeasure Comparison Graph

```
Deep Learning LSTM Extension Accuracy

LSTM Accuracy  : 94.12649273872375
LSTM Precision : 0.21538461538461537
LSTM Recall    : 7.6923076923076925
LSTM Fmeasure  : 0.41903621670158636
Deep Learning CNN Accuracy

CNN Accuracy  : 99.43740963935852
CNN Precision : 100.0
CNN Recall    : 5.499999999999999
CNN Fmeasure  : 10.939830929885627
```

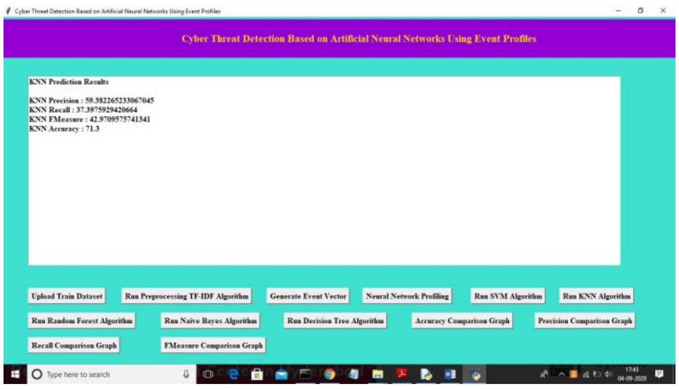
The above window displays the FMEA sure values, accuracy, precision, recall, and recall rates of both methods. Next, choose the option to "Run SVM Algorithm" to use the pre-built SVM algorithm.


Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles
Upload Train Dataset
Run Preprocessing TF-IDF Algorithm
Generate Event Vector
Neural Network Profiling
Run SVM Algorithm
Run KNN Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Accuracy Comparison Graph
Precision Comparison Graph
Recall Comparison Graph
FMeasure Comparison Graph

```
SVM Precision : 66.48136008383452
SVM Recall : 52.972516618839336
SVM FMeasure : 58.84309196202946
SVM Accuracy : 83.5
```

You can see the results of the SVM algorithm above, and you can start the KNN algorithm by clicking the button labeled "Run KNN Algorithm."

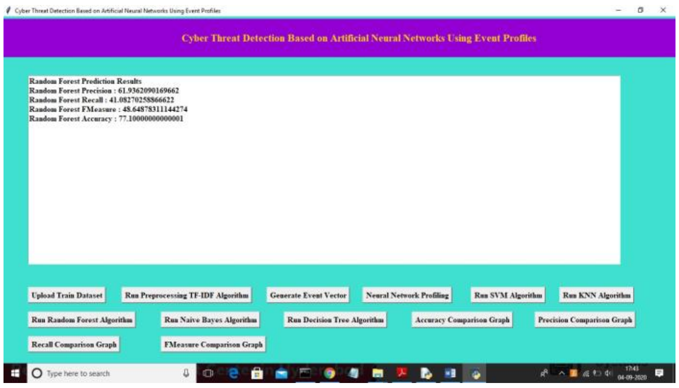
2. Random Forest Algorithm


Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles
Upload Train Dataset
Run Preprocessing TF-IDF Algorithm
Generate Event Vector
Neural Network Profiling
Run SVM Algorithm
Run KNN Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Accuracy Comparison Graph
Precision Comparison Graph
Recall Comparison Graph
FMeasure Comparison Graph

```
KNN Prediction Results

KNN Precision : 59.382265233067045
KNN Recall : 37.3975929420664
KNN FMeasure : 42.9709575741341
KNN Accuracy : 71.3
```

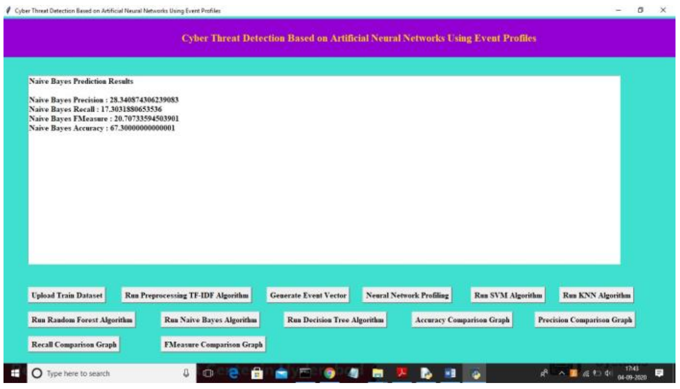
KNN algorithm result values are shown on the previous screen; to proceed with Random Forest algorithm, choose 'Run Random Forest Algorithm' button.


Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles
Upload Train Dataset
Run Preprocessing TF-IDF Algorithm
Generate Event Vector
Neural Network Profiling
Run SVM Algorithm
Run KNN Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Accuracy Comparison Graph
Precision Comparison Graph
Recall Comparison Graph
FMeasure Comparison Graph

```
Random Forest Prediction Results
Random Forest Precision : 61.9362090169662
Random Forest Recall : 41.08270258866622
Random Forest FMeasure : 48.64878311144274
Random Forest Accuracy : 77.10000000000001
```

The result values of the Random Forest algorithm have been shown; to proceed with the Naive Bayes algorithm, choose the button labeled "Run Naive Bayes Algorithm" from the previous screen.

3. Naïve Base Algorithm

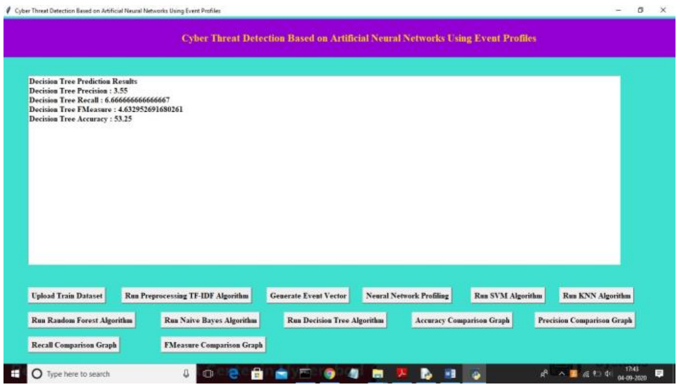

Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles
Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles
Upload Train Dataset
Run Preprocessing TF-IDF Algorithm
Generate Event Vector
Neural Network Profiling
Run SVM Algorithm
Run KNN Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Accuracy Comparison Graph
Precision Comparison Graph
Recall Comparison Graph
FMeasure Comparison Graph
Type here to search
17:43
04-09-2020

```
Naive Bayes Prediction Results

Naive Bayes Precision : 28.340874306239083
Naive Bayes Recall : 17.3031880653536
Naive Bayes FMeasure : 20.70733594503901
Naive Bayes Accuracy : 67.30000000000001
```

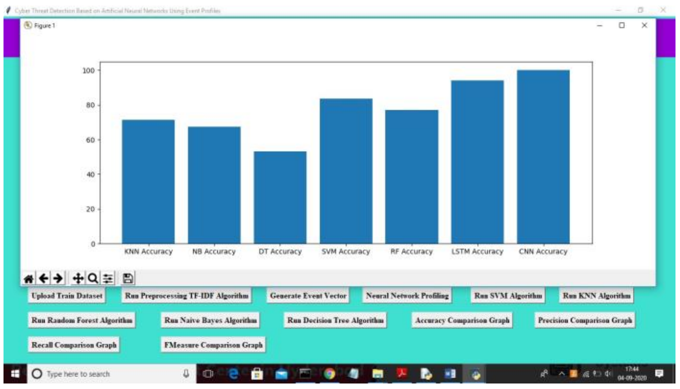
The result values of the Naive Bayes algorithm are shown on the preceding screen; to proceed with the Decision Tree Algorithm, click the "Run" button.

4. Decision Tree Algorithm

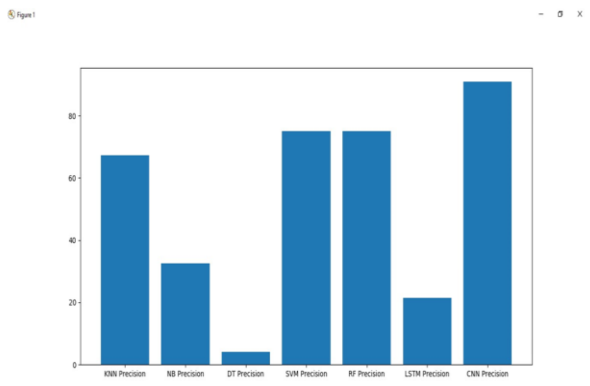

Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles
Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles
Upload Train Dataset
Run Preprocessing TF-IDF Algorithm
Generate Event Vector
Neural Network Profiling
Run SVM Algorithm
Run KNN Algorithm
Run Random Forest Algorithm
Run Naive Bayes Algorithm
Run Decision Tree Algorithm
Accuracy Comparison Graph
Precision Comparison Graph
Recall Comparison Graph
FMeasure Comparison Graph
Type here to search
17:43
04-09-2020

```
Decision Tree Prediction Results
Decision Tree Precision : 3.55
Decision Tree Recall : 6.666666666666667
Decision Tree FMeasure : 4.632952691680261
Decision Tree Accuracy : 53.25
```

To see a graph comparing how well different algorithms perform, click the "Accuracy Comparison" button.



From the preceding graph, where the x-axis indicates algorithm names and the y-axis represents accuracy, it is clear that LSTM and CNN are effective methods. Below is a graph showing the results when you click "Precision Comparison Graph."



Having looked at the preceding graph, you may now go to the 'Recall Comparison Graph' to see how well CNN is doing.

System Study:

The system study phase involves a thorough analysis of the project's feasibility and the presentation of a business proposal with a preliminary project plan and cost estimates. During this phase, the feasibility of the proposed system is critically examined to ensure it does not pose an undue financial burden on the organization. To conduct a feasibility analysis, it is crucial to have a clear understanding of the system's major requirements. The three primary considerations in feasibility analysis include:

1. Economic Feasibility:

This study assesses the economic impact that the system will have on the organization. It involves an examination of the funds available for research and development, considering budget limitations. The system's development should align with the budgetary constraints. This goal is achieved by leveraging freely available technologies, with only customized components incurring costs.

2. Technical Feasibility:

Technical feasibility analysis evaluates whether the system's technical requirements align with the organization's available resources. It is imperative that the system does not place excessive demands on the existing technical infrastructure, as this could strain the client's resources. The ideal system should have modest technical requirements, necessitating minimal or no changes to be made for its implementation.

3. Social Feasibility:

Social feasibility assessment aims to gauge the level of acceptance of the system by its users. This includes the process of training users to effectively utilize the system. Users should not perceive the system as a threat; rather, they should embrace it as a necessary tool. User acceptance depends on the methods employed to educate users about the system and make them familiar with it. Boosting user confidence is crucial, as it encourages constructive criticism, which is valuable given that users are the ultimate beneficiaries of the system.

Input and Output Design:

Input Design:

Input design serves as the vital link between the information system and the user. It encompasses the development of specifications and procedures for data preparation, ensuring that transaction data is in a usable form for processing. Data input can occur through various means, such as reading data from written or printed documents using computerized scanning or having individuals manually enter data into the system. The primary focus of input design is to control the volume of input required, minimize errors, prevent delays, streamline processes, and maintain simplicity. Input design also takes into account security and user-friendliness, ensuring data privacy and ease of use.

Output Design:

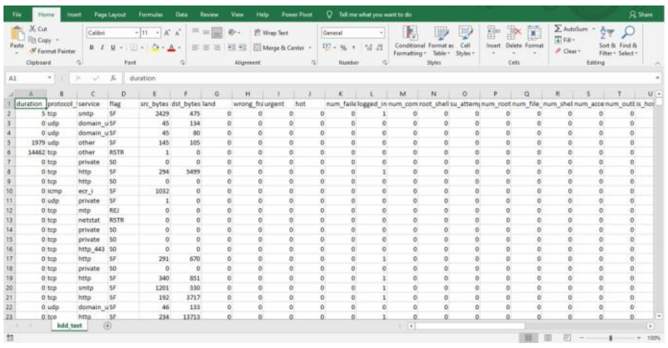
Output design is essential for providing users with information that meets their requirements and presents data clearly and effectively. In any information system, the results of processing are communicated to users and other systems through outputs. Output design determines how information is displayed for immediate use and also considers hard-copy outputs. High-quality output design is critical for facilitating user decision-making. When designing computer output, it's essential to:

- Organize the design process systematically, ensuring that the right output is developed.
- Choose appropriate methods for presenting information.
- Create documents, reports, or other formats containing information produced by the system.

The objectives of output design include conveying information about past activities, current status, or future projections, signaling important events or triggers, and confirming actions. Well-structured output enhances the system's ability to assist users in making informed decisions.

V. RESULT AND DISCUSSION

Datasets



duration	protocol_	service	flag	src_bytes	dst_bytes	land	wrong_fra	urgent	hot	num_faile	logged_in	num_com	root_shel	su_attem	num_root	num_file_	num_shel	num_acce	num_outb	is_hos
5	tcp	smtp	SF	2429	475	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
0	udp	domain_u	SF	45	134	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	udp	domain_u	SF	45	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1979	udp	other	SF	145	105	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
14462	tcp	other	RSTR	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	tcp	http	SF	294	5499	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
0	tcp	http	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	icmp	ecr_i	SF	1032	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	udp	private	SF	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	tcp	mtp	REJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	tcp	netstat	RSTR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	tcp	http_443	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	tcp	http	SF	291	670	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	tcp	http	SF	340	851	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
0	tcp	smtp	SF	1201	330	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
0	tcp	http	SF	192	3717	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
0	udp	domain_u	SF	46	133	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	tcp	http	SF	234	13713	0	0	0	0	0	1	0	0	0	0	0	0	0	0	

Our data comes from two major commercial systems (ESX1 and ESX-2) that we've given the descriptive names ESX1 and ESX-2. Over a period of 5 months for ESX-1 and 30 days for ESX-2, respectively, security raw events were gathered, with the detecting threat information recorded independently by SOC security experts whenever a network intrusion occurred. Time of threat occurrence, related attacks, attack type, response content, attack IP address, and victim network details are all included in the list of threat detection data.

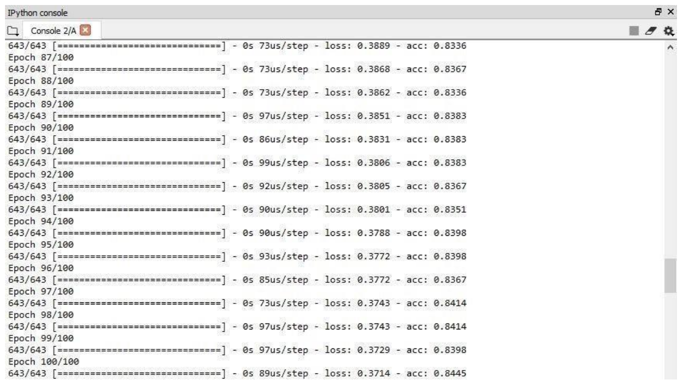
We looked over 798 cyber threat detection cases in ESX-1 that span the whole collecting window. There were 240 scanning attacks, 547 system hack-king assaults, and 11 worm attacks among the total number of registered cyber threats. Comparatively, there are 941 scans, 3,077 hacks, and 51 worm assaults in ESX-2. SOC analysts manually sorted attacks into these categories. System hacking attacks may be broken down into many types, such as cross-site scripting, distributed denial of service, brute force, and injection. Scanning attacks include trojans, backdoors, and other malware. In all, 4,079 cyber threats were identified.

Data Visualization

t-SNE (t-distributed stochastic neighbor embedding) serves a dual purpose in data analysis. It's not only a common tool for visualizing vector data but also functions as an embedding tool to portray high-dimensional data effectively. t-SNE achieves this by transforming high-dimensional data into twodimensional maps while preserving the neighborhood relationships among data points. In this context, the t-SNE method is employed to visualize datasets like CICIDS 2017 and ESX-2. The t-SNE plots illustrate that both normal and attack data points are closely situated in the same space, making it challenging to distinguish between them as either normal or malicious. Despite the clustering of normal and attack data in the t-SNE plots, it's evident that they aren't linearly separable. Deep learning approaches are commonly utilized for handling high-dimensional data with non-linear characteristics, which is one of the reasons why deep learning methods are chosen for cyber threat detection.

Experimental Results

The experimental results yield two significant conclusions. First, the proposed machine learning models show promise as effective tools for network intrusion detection. When evaluated using well-established benchmark datasets such as NSLKDD and CICIDS 2017, these models perform on par with conventional machine learning techniques. This implies that the methods integrated into the AI-SIEM system have the potential for learning-based network intrusion detection. Second, when conventional learning-based methods that perform well on benchmark datasets are applied in real-world scenarios, their overall accuracy tends to be less reliable. However, the accuracy of the three EP-ANN models proposed in this study does not exhibit significant degradation, even when handling substantial amounts of data and lacking benchmark dataset features, as seen in the results for ESX-2. In contrast, the accuracy of conventional methods decreases from approximately 0.90 to 0.85 in such real-world scenarios. These findings underscore the robustness and practicality of the EP-ANN models in real-world network intrusion detection applications.



```
643/643 [==============================] - 0s 73us/step - loss: 0.3889 - acc: 0.8336
Epoch 87/100
643/643 [==============================] - 0s 73us/step - loss: 0.3868 - acc: 0.8367
Epoch 88/100
643/643 [==============================] - 0s 73us/step - loss: 0.3862 - acc: 0.8336
Epoch 89/100
643/643 [==============================] - 0s 97us/step - loss: 0.3851 - acc: 0.8383
Epoch 90/100
643/643 [==============================] - 0s 86us/step - loss: 0.3831 - acc: 0.8383
Epoch 91/100
643/643 [==============================] - 0s 99us/step - loss: 0.3806 - acc: 0.8383
Epoch 92/100
643/643 [==============================] - 0s 92us/step - loss: 0.3805 - acc: 0.8367
Epoch 93/100
643/643 [==============================] - 0s 90us/step - loss: 0.3801 - acc: 0.8351
Epoch 94/100
643/643 [==============================] - 0s 90us/step - loss: 0.3788 - acc: 0.8398
Epoch 95/100
643/643 [==============================] - 0s 93us/step - loss: 0.3772 - acc: 0.8398
Epoch 96/100
643/643 [==============================] - 0s 85us/step - loss: 0.3772 - acc: 0.8367
Epoch 97/100
643/643 [==============================] - 0s 73us/step - loss: 0.3743 - acc: 0.8414
Epoch 98/100
643/643 [==============================] - 0s 97us/step - loss: 0.3743 - acc: 0.8414
Epoch 99/100
643/643 [==============================] - 0s 97us/step - loss: 0.3729 - acc: 0.8398
Epoch 100/100
643/643 [==============================] - 0s 89us/step - loss: 0.3714 - acc: 0.8445
```

Epoch

In the context of neural networks, an epoch is a single iteration over the whole training set. A single forward pass and a single reverse pass are the iterations. The effectiveness of a classification system may be summarized with the help of a confusion matrix. Count numbers summarize the number of accurate and wrong predictions and are further broken down by class. This solves the mystery of the muddled matrix. The proportion of correctly identified cyber threats is the classification accuracy. Large numbers are difficult for computers to store. Instead, computers only have a certain amount of storage capacity for each individual number. Therefore, difficulties may arise when the number of time units that have passed since a system's epoch surpasses the maximum number that can fit in the space allocated for the time representation. Overflow may cause unpredictable behavior in certain systems, but in others, the time value will be reset to zero and the computer will treat the current time as if it were the epoch time again.

VI. Conclusion

In the realm of cybersecurity, Intrusion Detection Systems (IDS) and skilled network security auditors play a crucial role in promptly identifying and mitigating threats. These systems and experts are entrusted with the responsibility of sifting through a multitude of security events to pinpoint the most critical threats in real-time. Their primary objective is to discern the key elements of an attack and translate them into IDS signatures. Threat detection stands as a cornerstone in the cybersecurity strategy of IT organizations, particularly those reliant on cloud infrastructure. It encompasses the ability of IT entities to swiftly and accurately recognize threats directed at the network, applications, or other assets within the network.

The proposed approach revolves around the transformation of a large volume of collected security events into individual event profiles. It employs a deep learning-based detection method to enhance the identification of cyber threats. In this context, an AI-Security Information and Event Management (AI-SIEM) system has been developed, incorporating event profiling for data preprocessing and harnessing various artificial neural network techniques such as Fully Connected Neural Networks (FCNN), Convolutional Neural Networks (CNN), and Long Short-Term Memory (LSTM). The primary objective of this system is to distinguish between true positive and false positive alerts, thereby enabling security analysts to respond swiftly to digital threats.

In conclusion, this paper introduces an AI-SIEM system that leverages event profiles and artificial neural networks. The innovation lies in the ability to condense vast amounts of data into event profiles and apply deep learning techniques for more effective cyber-threat detection. The AI-SIEM system empowers security analysts to efficiently handle significant security alerts, facilitating the comparison of long-term security data. By reducing false positive alerts, it enables rapid responses to cyber threats embedded within a multitude of security events. Performance evaluation involved benchmark datasets (NSLKDD, CICIDS2017), as well as real-world datasets. Results indicate that this technology outperforms traditional machine learning methods in terms of accurate classifications.

Future Work

Looking ahead, future work will concentrate on improving early threat prediction through multiple deep learning approaches aimed at identifying long-term patterns in historical data. Additionally, efforts will be directed towards enhancing the precision of labeled datasets for supervised learning, involving the meticulous labeling of raw security events by SOC analysts over several months. The creation of purpose-built test beds, encompassing big data platforms and AI-SIEM systems, will continue to play a role in performance evaluations. Furthermore, the collection of real-world Intrusion Prevention System (IPS) data over an extended period will provide valuable insights in the context of cybersecurity defense mechanisms

VII. References

1. S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, K. Han, "Enhanced Network Anomaly Detection Based on Deep Neural Networks," IEEE Access, vol. 6, pp. 48231-48246, 2018.
2. B. Zhang, G. Hu, Z. Zhou, Y. Zhang, P. Qian, L. Chang, "Network Intrusion Detection Based on Directed Acyclic Graph and Belief Rule Base", ETRI Journal, vol. 39, no. 4, pp. 592-604, Aug. 2017
3. W. Wang, Y. Sheng and J. Wang, "HAST-IDS: Learning hierarchical spatialtemporal features using deep neural networks to improve intrusion detection," IEEE Access, vol. 6, no. 99, pp. 1792-1806, 2018.

4. M. K. Hussein, N. Bin Zainal and A. N. Jaber, "Data security analysis for DDoS defense of cloud-based networks," 2015 IEEE Student Conference on Research and Development (Scored), Kuala Lumpur, 2015, pp. 305-310.
5. S. Sandeep Sekaran, K. Kandasamy, "Profiling SIEM tools and correlation engines for security analytics," In Proc. Int. Conf. Wireless Com., Signal Prove. and Net. (Wisp NET), 2017, pp. 717-721.
6. Hubbell and V.Surya narayana False alarm minimization techniques in signaturebased intrusion detection systems: A survey,'' Compute. Common., vol. 49, pp. 117, Aug. 2014.
7. A. Naser, M. A. Majid, M. F. Zolile and S. Anwar, "Trusting cloud computing for personal files," 2014 International Conference on Information and Communication Technology Convergence (ICTC), Busan, 2014, pp. 488-489.
8. Y. Shen, E. Marconi, P. Verviers, and Gianluca Stringham, "Tiresias: Predicting Security Events Through Deep Learning," In Proc. ACM CCS 18, Toronto, Canada, 2018, pp. 592-605.
9. Kyle Soaks and Nicolas Christin, "Automatically detecting vulnerable websites before they turn malicious,", In Proc. USENIX Security Symposium., San Diego, CA, USA, 2014, pp.625-640.
10. K. Veerama channid, I. Arnaldo, V. Koraput, C. Basis, K. Li, "AI2: training a big data machine to defend," In Proc. IEEE Bigdata Security HPSC IDS, New York, NY, USA, 2016, pp. 49-54
11. Mahmood Lavallee, Ebrahim Bagheri, Wei Lu and Ali A. Ghobadi, "A detailed analysis of the kid cup 99 data set," In Proc. of the Second IEEE Int. Conf. Comp. Int. for Sec. and Def. App., pp. 53-58, 2009.
12. I. Sharfuddin, A. H. Lashari, A. A. Ghobadi, "Toward generating a new intrusion detection dataset and intrusion traffic characterization", Proc. Int. Conf. Inf. Syst. Scur. Privacy, pp. 108- 116, 2018.
13. [online] Available: http://www.takakura.com/Kyoto_data/
14. N. Shone, T. N. Ngoc, V. D. Phail and Q. Shi, "A deep learning approach to network intrusion detection," IEEE Trans. Emerge. Topics Compute. Intel., vol. 2, pp. 41-50, Feb. 2018
15. R. Vijayakumar, Mamoon Alazar, K. P. Soman, P. Poorna Chandran, Ameer AlNamrata and Sit Lakshmi Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," IEEE Access, vol. 7, pp. 41525-41550, Apr. 2019.
16. W. Hu, W. Hu, S. Maybank, "Ad boost-based algorithm for network intrusion detection," IEEE Trans. Syst. Man B Cybernet., vol. 38, no. 2, pp. 577-583, Feb. 2008.
17. T.-F. Yen et al., "Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks", Proc. 29th Annul. Compute. Security Appl. Conf., New York, NY, USA, 2013, pp. 199-208.
18. K.-O. Darken, T Rix, C Kleiner, B Hellmann, L. Renner's, "Seem approach for a higher level of its security in enterprise networks", In Proc. IDAACS, Warsaw, Poland, 2015, pp. 322-327.
19. en.wikipedia.org "Security information and event management," 2016 [Online] Available information and event management.
20. Y. Lacuna, L. Bottom, Y. Bagnio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.
21. C. Dong, C. C. Loy, K. He and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 2, pp. 295- 307, 1 Feb. 2016.
22. A. Apathy, "Connecting images and natural language," Ph.D. dissertation, Fac. Compute. Sci., Stanford Univ., Stanford, CA, USA, 2016.

23. A. Krushinski, I. Subsieve, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," In Proc. Of the 25th Int. Confront Neural Inf. Proc. Systems -Volume 1, ser. NIPS’12, 2012, pp. 1097-1105.
24. Q. Zhu, X. Li, A. Conesa, and C. Pereira, “Gram-CNN: a deep learning approach with local context for named entity recognition in biomedical text,” Bioinformatics, vol. 34, no. 9, pp. 1547–1554, 2017.
25. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, “Malware traffic classification using convolutional neural network for representation learning,” In Proc. Int. Conf. on Infor. Net. (ICOIN), Da Nang, Vietnam, Jan. 2017, pp. 712–717.
26. Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, “Intrusion detection using convolutional neural networks for representation learning,” In Proc. Int. Conf. Neural Information Springer, 2017, pp. 858–866.
27. M. Alazar, S. Venkatraman, P. Watters, and M. Alazar, “Zero-day malware detection based on supervised learning algorithms of API call signatures,” In Proc. 9th Australis. Data Mining Conf., vol. 121. Ballarat, Australia, Dec. 2011, pp. 171182.
28. E. Raff, J. Sylvester, and C. Nicholas, “Learning the PE header, malware detection with minimal domain knowledge," In Proc. 10th CalWORKs Atif. Intel. secure. New York, NY, USA, Nov. 2017, pp. 121-132.
29. J. Gu et al., "Recent advances in convolutional neural networks", Corer, pp. 187-332, Dec. 2017
30. Kene Wu, Zune Chen, Wei Li, "A Novel Intrusion Detection Model for a Massive Network Using Convolutional Neural Networks", Access IEEE, vol. 6, pp. 5085050859, 2018
31. Taejon Kim, Sang C. Suh, Hyunjoo Kim, Jongmyo Kim and Jingo Kim, "An Encoding Technique for CNN-based Network Anomaly Detection," In Proc. IEEE International Conference on Big Data (IEEE Bigdata), Seattle, WA, USA, Jan. 2019, pp. 2960-2965.
32. R. Vijayakumar, M. Alazar, K. P. Soman, P. Poorna Chandran and S. Venkatraman "Robust Intelligent Malware Detection Using Deep Learning," IEEE Access, vol. 7, pp. 46717-46738, Apr. 2019
33. Y. Lacuna, Y. Bagnio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436-444, 2015.
34. S. Hoch Reiter, Y. Bagnio, P. Francona, and J. Schmid Huber, “Gradient flow in recurrent nets: the difficulty of learning long-term dependencies,” 2001.
35. D. Similarities and Olivier Marko witch, "Reducing Data Complexity in Feature Extraction and Feature Selection for Big Data Security Analytics", In Proc. Int. Conf. Data Intel. and Sec. (ICDIS), South Padre Island, TX, USA, May 2018, pp. 43-48.
36. V. N. Inoculum, S. Aris, S. R. Ravia, "Security issues associated with big data in cloud computing", International Journal of Network Security & Its Applications, vol. 6, no. 3, pp. 45, 2014.