

Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology Degree in

Information Technology

By

GOUTHAM KUMAR.J

(REG.NO.37120037)



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600119. TAMILNADU.

MARCH – 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(Established under Section 3 of UGC Act, 1956)

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600119

www.sathyabamauniversity.ac.in



SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bona-fide work of **GOUTHAM KUMAR J (Reg. No. 37120037)** who have done the project in “**Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles**” under our supervision from to June 2019 to March 2021

Internal Guide

Dr. S GOWRI

Head of the Department

Dr. R. SUBHASHINI, M.E., Ph.D.,

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I, GOTHAM KUMAR.J (Reg. No. 37120037) hereby declare that the Professional Training Report on **“Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles”** done by me under the guidance of Dr. Gowri, at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. Shashikala M.E., Ph.D., Dean**, School of Computing, **Dr.R.Subhashini M.E., Ph.D. , Head** of the Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide Dr. S Gowri for his valuable guidance, suggestions and constant courage paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

ABSTRACT

One of the major challenges in cybersecurity is the provision of an automated and effective cyber-threats detection technique. In this paper, we present an AI technique for cyber-threats detection, based on artificial neural networks. The proposed technique converts multitude of collected security events to individual event profiles and use a deep learning-based detection method for enhanced cyber-threat detection. For this work, we developed an AI-SIEM system based on a combination of event profiling for data preprocessing and different artificial neural network methods, including FCNN, CNN, and LSTM. The system focuses on discriminating between true positive and false positive alerts, thus helping security analysts to rapidly respond to cyber threats. All experiments in this study are performed by authors using two benchmark datasets (NSLKDD and CICIDS2017) and two datasets collected in the real world. To evaluate the performance comparison with existing methods, we conducted experiments using the five conventional machine-learning methods (SVM, k-NN, RF, NB, and DT). Consequently, the experimental results of this study ensure that our proposed methods are capable of being employed as learning-based models for network intrusion-detection, and show that although it is employed in the real world, the performance outperforms the conventional machine-learning methods.

TABLE OF CONTENTS

CAPTHERS NO	TITLE	PAGE NO
1	INTRODUCTION	1
1.1	Outline of Project	1
1.2	Domain Introduction	2
1.3	Benefits of Deep Neural Network	3
2	LITERATURE SURVEY	4
2.1	Enhanced Network Anomaly Detection	4
2.2	Network Intrusion Detection	5
2.3	Learning hierarchical spatial-temporal	6
3	AIM AND SCOPE OF THE PROJECT	7
3.1	Aim	7
3.2	Scope of the project	7
3.3	Objective	7
3.4	System requirement	8
3.5	Hardware requirements	8
3.5.1	Software requirements	9
3.6	Software features	9
3.6.1	Python	9
3.6.2	Kara's	10
3.6.3	IDE Setup	10
3.7	DJANGO	11

3.8	Algorithm	12
4	METHODOLOGY	13
4.1	Existing System	13
4.2	PROPOSED SYSTEM	13
4.3	SYSTEM ARCHITECTURE	14
4.4	MODULES	14
4.5	Module Description	14
4.5.1	UPLOAD DATASETS	15
4.5.2	TF-IDF ALGORITHM	16
4.5.3	GENRATE EVENT VECTOR	17
4.5.4	NEURAL NETWORKS PROFRING	17
4.5.5	SVM ALGORITHM	18
4.5.6	KNN ALGORITHM	19
4.5.7	RANDOM FOREST ALGORITHM	20
4.5.8	NAÏVE BAYES ALGORITHM	21
4.5.9	DECISION THREE ALGORITHM	21
4.6	SYSTEM STUDY	23
4.6.1	ECONOMICAL FEASIBILITY	23
4.6.2	TECHNICAL FEASIBILITY	23
4.6.3	SOCIAL FEASIBILITY	24
4.7	INPUT AND OUTPUT DESIGN	24
4.7.1	INPUT DESIGN	24
4.7.2	OUTPUT DESIGN	25

5	RESULTS AND DISCUSSIONS	26
5.1	DATASETS	26
5.2	DATA VISUALIZATION	27
5.3	Experimental Results	28
6	CONCLUSION AND FUTURE WORK	29
6.1	SUMMARY	29
6.2	CONCLUSION	30
6.3	FUTURE WORK	31
	REFERENCES	32
	APPENDIX	35
	A. Source code	35
	B. Screenshots	40
	C. Publication	45

LIST OF FIGURES

FIGURE No.	FIGURE NAME
3.7	DJANGO framework
4.3	Architectural Diagram
4.5.1	UPLOAD DATASETS
4.5.2	TF-IDF ALGORITHM
4.5.3	GENRATE EVENT VECTOR
4.5.4	NEURAL NETWORKS PROFRING
4.5.5	SVM ALGORITHM
4.5.6	KNN ALGORITHM
4.5.7	RANDOM FOREST ALGORITHM
4.5.8	NAÏVE BAYES ALGORITHM

LIST OF ABBRETIATIONS

NO.	
1.	SIEM
2.	NSLKDD
3.	DNN
4.	CNN
5.	IDE
6.	TF-IDF
7.	LSTM
8.	SVM
9.	SKNN

CHAPTER 1

INTRODUCTION

1.1 OUTLINE OF THE PROJECT

With the emergence of artificial intelligence (AI) techniques, learning-based approaches for detecting cyberattacks, have become further improved, and they have achieved significant results in many studies. However, owing to constantly evolving cyberattacks, it is still highly challenging to protect IT systems against threats and malicious behaviors in networks. Because of various network intrusions and malicious activities, effective defenses and security considerations were given high priority for finding reliable solutions [1], [2], [3], [4].

Traditionally, there are two primary systems for detecting cyber-threats and network intrusions. An intrusion prevention system (IPS) is installed in the enterprise network, and can examine the network protocols and flows with signature-based methods primarily. It generates appropriate intrusion alerts, called the security events, and reports the generating alerts to another system, such as SIEM. The security information and event management (SIEM) has been focusing on collecting and managing the alerts of IPSs. The SIEM is the most common and dependable solution among various security operations solutions to analyze the collected security events and logs [5]. Moreover, security analysts make an effort to investigate suspicious alerts by policies and threshold, and to discover malicious behavior by analyzing correlations among events, using knowledge related to attacks.

A learning-based method geared toward determining whether an attack occurred in a large amount of data can be useful to analysts who need to instantly analyze numerous events. According to [10], information security solutions generally fall into two categories: analyst-driven and machine learning-driven solutions. Analyst-driven solutions rely on rules determined by security experts called analysts. Meanwhile, machine learning-driven solutions used to detect rare or anomalous patterns can improve detection of new cyber threats [10]. Nevertheless, while learning-based approaches are useful in detecting cyberattacks in systems and

networks, we observed that existing learning-based approaches have four main limitations.

First, learning-based detection methods require labeled data, which enable the training of the model and evaluation of generated learning models. Furthermore, it is not straightforward to obtain such labeled data at a scale that allow accurate training of a model. Despite the need for labeled data, many commercial SIEM solutions do not maintain labeled data that can be applied to supervised learning models [10].

Second, most of the learning features that are theoretically used in each study are not generalized features in the real world, because they are not contained in common network security systems [3]. Hence, it makes difficult to utilize to practical cases. Recent efforts on intrusion detection research have considered an automation approach with deep learning technologies, and performance has been evaluated using well known datasets like NSLKDD [11], CICIDS2017 [12], and Kyoto-Honeypot [13]. However, many previous studies used benchmark dataset, which, though accurate, are not generalizable to the real world because of the insufficient features. To overcome these limitations, an employed learning model requires to evaluate with datasets that are collected in the real world.

1.2 DOMAIN INTRODUCTION

The Deep Neural Network (DNN) is a neural network with a certain level of complexity, a neural network with more than two layers. Deep neural networks use sophisticated mathematical modelling to process data in complex ways.

A neural network, in general, is a technology built to simulate the activity of the human brain – specifically, pattern recognition and the passage of input through various layers of simulated neural connections.

Many experts define deep neural networks as networks that have an input layer, an output layer and at least one hidden layer in between. Each layer performs specific types of sorting and ordering in a process that some refer to as “feature hierarchy.” One of the keys uses of these sophisticated neural networks is dealing with unlabelled or unstructured data. The phrase “deep learning” is also used to describe these deep neural networks, as deep learning represents a specific form of machine

learning where technologies using aspects of artificial intelligence seek to classify and order information in ways that go beyond simple input/output protocols.

1.3 BENEFITS OF DEEP NEURAL NETWORK

Neural networks use randomness by design to ensure they effectively learn the function being approximated for the problem. Randomness is used because this class of machine learning algorithm performs better with it than without.

The most common form of randomness used in neural networks is the random initialization of the network weights. Although randomness can be used in other areas, here is just a short list:

- Randomness in Initialization, such as weights.
- Randomness in Regularization, such as dropout.
- Randomness in Layers, such as word embedding.

CHAPTER 2

LITERATURE SURVEY

2.1 Enhanced Network Anomaly Detection Based on Deep Neural Networks

Due to the monumental growth of Internet applications in the last decade, the need for security of information network has increased manifolds. As a primary defense of network infrastructure, an intrusion detection system is expected to adapt to dynamically changing threat landscape. Many supervised and unsupervised techniques have been devised by researchers from the discipline of machine learning and data mining to achieve reliable detection of anomalies. Deep learning is an area of machine learning which applies neuron-like structure for learning tasks. Deep learning has profoundly changed the way we approach learning tasks by delivering monumental progress in different disciplines like speech processing, computer vision, and natural language processing to name a few. It is only relevant that this new technology must be investigated for information security applications. The aim of this paper is to investigate the suitability of deep learning approaches for anomaly-based intrusion detection system. For this research, we developed anomaly detection models based on different deep neural network structures, including convolutional neural networks, autoencoders, and recurrent neural networks. These deep models were trained on NSLKDD training data set and evaluated on both test data sets provided by NSLKDD, namely NSLKDD Test+ and NSLKDDTest21. All experiments in this paper are performed by authors on a GPU-based test bed. Conventional machine learning-based intrusion detection models were implemented using well-known classification techniques, including extreme learning machine, nearest neighbor, decision-tree, random-forest, support vector machine, naive-bays, and quadratic discriminant analysis. Both deep and conventional machine learning models were evaluated using well-known classification metrics, including receiver operating characteristics, area under curve, precision-recall curve, mean average precision and accuracy of classification.

Experimental results of deep IDS models showed promising results for real-world application in anomaly detection systems.

2.2 Network Intrusion Detection Based on Directed Acyclic Graph and Belief Rule Base

Intrusion detection is very important for network situation awareness. While a few methods have been proposed to detect network intrusion, they cannot directly and effectively utilize semi-quantitative information consisting of expert knowledge and quantitative data. Hence, this paper proposes a new detection model based on a directed acyclic graph (DAG) and a belief rule base (BRB). In the proposed model, called DAG-BRB, the DAG is employed to construct a multi-layered BRB model that can avoid explosion of combinations of rule number because of a large number of types of intrusion. To obtain the optimal parameters of the DAG-BRB model, an improved constraint covariance matrix adaption evolution strategy (CMA-ES) is developed that can effectively solve the constraint problem in the BRB. A case study was used to test the efficiency of the proposed DAG-BRB. The results showed that compared with other detection models, the DAG-BRB model has a higher detection rate and can be used in real networks.

2.3 HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks

The development of an anomaly-based intrusion detection system (IDS) is a primary research direction in the field of intrusion detection. An IDS learns normal and anomalous behavior by analyzing network traffic and can detect unknown and new attacks. However, the performance of an IDS is highly dependent on feature design and designing a feature set that can accurately characterize network traffic is still an ongoing research issue. Anomaly-based IDSs also have the problem of a high false alarm rate (FAR), which seriously restricts their practical applications. In this paper, we propose a novel IDS called the hierarchical spatial-temporal features-based intrusion detection system (HAST-IDS), which first learns the low-level spatial features of network traffic using deep convolutional neural networks (CNNs) and then learns high-level temporal features using long short-term memory networks. The entire process of feature learning is completed by the deep neural networks

automatically; no feature engineering techniques are required. The automatically learned traffic features effectively reduce the FAR. The standard DARPA1998 and ISCX2012 data sets are used to evaluate the performance of the proposed system. The experimental results show that the HAST-IDS outperforms other published approaches in terms of accuracy, detection rate, and FAR, which successfully demonstrates its effectiveness in both feature learning and FAR reduction.

CHAPTER 3

AIM AND SCOPE OF THE PROJECT

3.1 AIM

The aim of the project is Threat detection and response is the most important aspect of cybersecurity for IT organizations that depend on cloud infrastructure. Threat detection, therefore, describes the ability of IT organizations to quickly and accurately identify threats to the network or to applications or other assets within the network.

3.2 SCOPE OF THE PROJECT

A learning-based method geared toward determining whether an attack occurred in a large amount of data can be useful to analysts who need to instantly analyze numerous events. According to [10], information security solutions generally fall into two categories: analyst-driven and machine learning-driven solutions.

3.3 OBJECTIVES

Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant Thus the objective of input design is to create an input layout that is easy to follow

3.4 SYSTEM REQUIREMENTS

For developing the application, the following are the Software Requirements:

1. Python
2. Django

Operating Systems supported

1. Windows 7
2. Windows 8
3. Windows 10

Technologies and Languages used to Develop

1. Python

Debugger and Emulator

- Any Browser

3.5 Hardware Requirements

For developing application, the following are the Hardware Requirements:

- Processor: Pentium IV or higher
- RAM: 256 MB
- Space on Hard Disk: minimum 512MB

3.6 SOFTWARE FEATURES

3.6.1 Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that code readability using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

3.6.2 Kera

Kera is Open-source Neural Network library written in Python that runs on top of Theano or Tensor flow. It is designed to be modular, fast and easy to use. It was developed by François Chollet, a Google engineer. Kera doesn't handle low-level computation. Instead, it uses another library to do it, called the "Backend.

So Kera is high-level API wrapper for the low-level API, capable of running on top of TensorFlow, CNTK, or Theano.

Kara's high-Level API handles the way we make models, defining layers, or set up multiple input-output models. In this level, Kera also compiles our model with loss and optimizer functions, training process with fit function. Kera doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine.

3.6.3 IDE Setup

The Machine Learning Concepts can be well implemented through PYTHON We have a numerous Python Tools but for the DNN implementation Anaconda can meet our needs. Spyder in Anaconda is chosen as the IDE setup.

The latest Anaconda Spyder has Python 3.6 version. Python 3.6 is unstable to hold the KERAS back end Apart from the inability to build over Python 3.6 KERAS requires two additional library packages THEANO and TENSORFLOW. Tensor flow can be implemented over Python3.5 only.

A cloning environment is built using the Anaconda Prompt the Python 3.5 is cloned using the command.

Condi create -n py35 python=3.5 anaconda

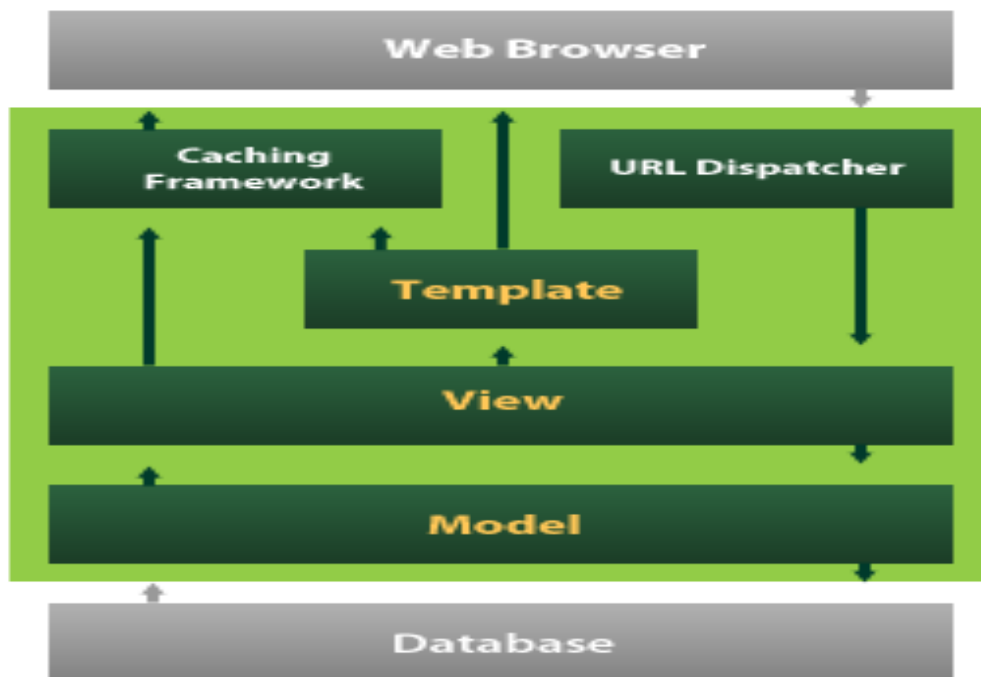
After activating the cloned environment, the Spyder is installed using the command So the Python3.5 has its own Spyder version which does not interfere with the base version of Anaconda.

3.7 DJANGO

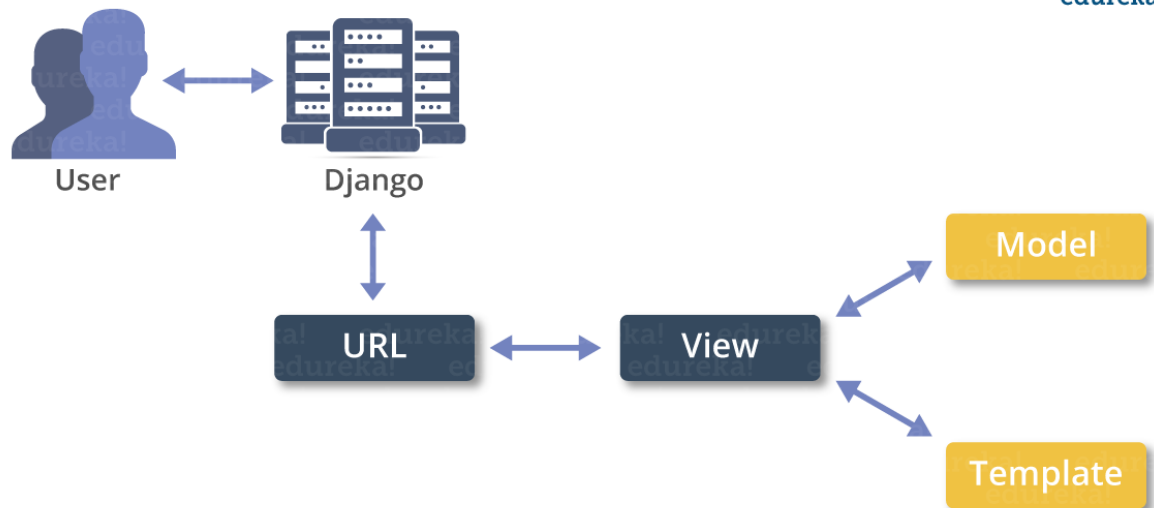
Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes

care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.



Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.



3.8 ALGORITHM

In this paper author is describing concept to detect threats using AI-SIEM (Artificial Intelligence-Security Information and Event Management) technique which is a combination of deep learning algorithms such as FCNN, CNN (Convolution Neural Networks) and LSTM (long short-term memory) and this technique works based on events profiling such as attack signatures. Author evaluating propose work performance with conventional algorithms such as SVM, Decision Tree, Random Forest, KNN and Naïve Bayes. Here I am implementing CNN and LSTM algorithms.

- Data Parsing
- TF-IDF
- Event Profiling Stage
- Deep Learning Neural Network Model

CHAPTER 4

METHODOLOGY

4.1 Existing System

As there is no staff accessible in automated cafés, it is hard for the eatery the board to appraise how the idea and the food is capable by the clients. Existing Rating frameworks, like Google and Trip Advisor, just in part tackle this issue, as they just cover a piece of the client's conclusions. These rating frameworks are just utilized by a subset of the clients who rate the café on free evaluating stages on their own drive. This applies basically to clients who experience their visit as certain or negative.

4.2 PROPOSED SYSTEM

In order to solve the above problem, all customers must be motivated to give a rating. This paper introduces an approach for a restaurant rating system that asks every customer for a rating after their visit to increase the number of ratings as much as possible. This system can be used unmanned restaurants; the scoring system is based on facial expression detection using pretrained convolutional neural network (CNN) models. It allows the customer to rate the food by taking or capturing a picture of his face that reflects the corresponding feelings. Compared to text-based rating system, there is much less information and no individual experience reports collected. However, this simple fast and playful rating system should give a wider range of opinions about the experiences of the customers with the restaurant concept.

4.3 SYSTEM ARCHITECTURE

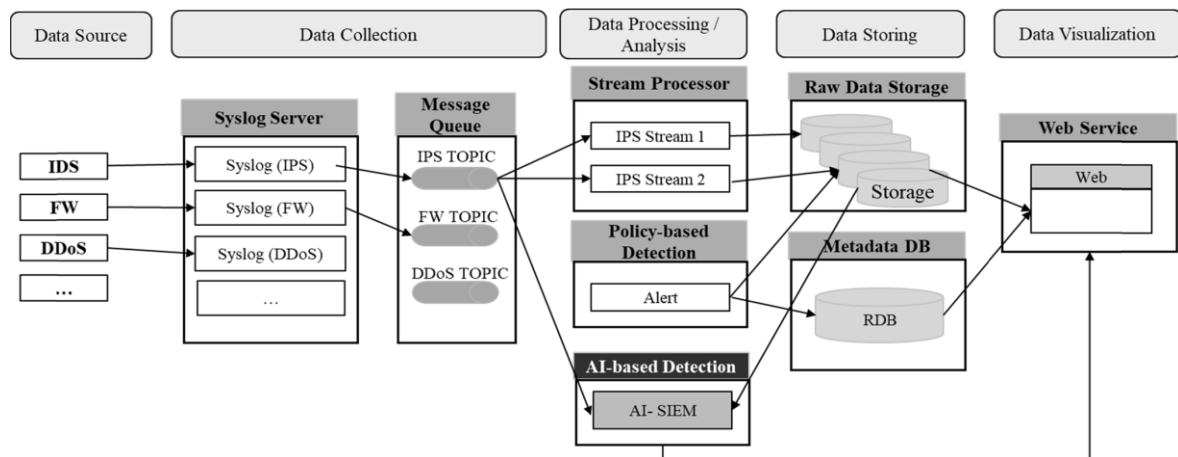


Fig 4.1 Architectural design

4.4 MODULES

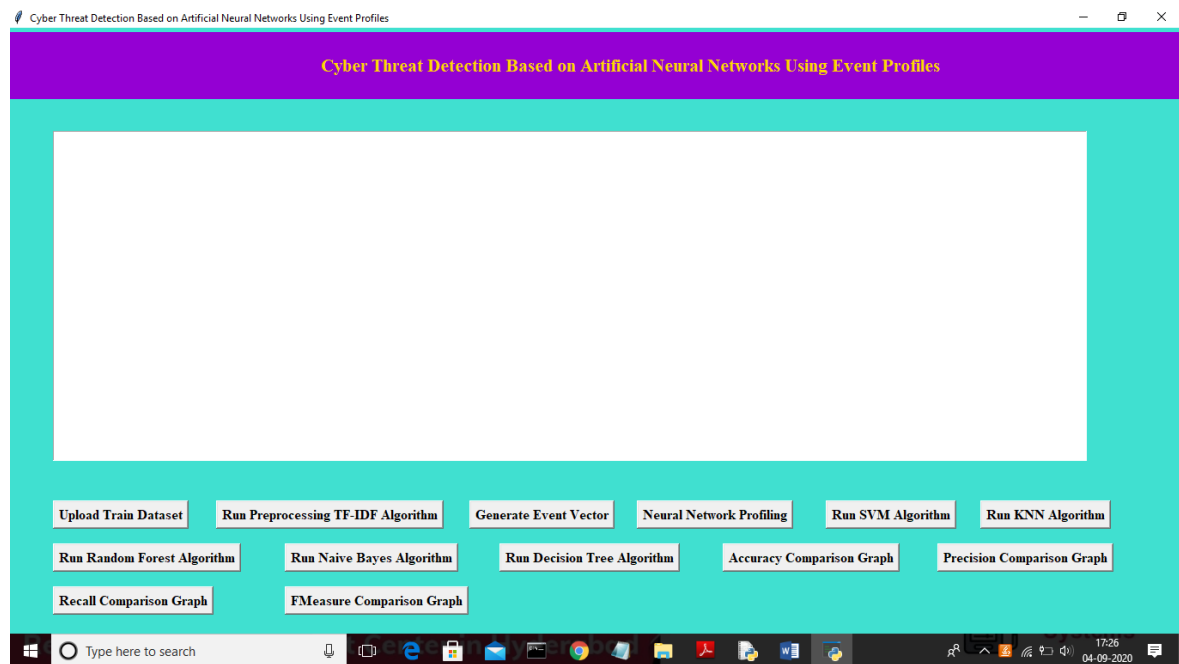
Propose algorithms consists of following module are.

1. Data Parsing: This module takes input dataset and parse that dataset to create a raw data event model
2. TF-IDF: using this module we will convert raw data into event vector which will contains normal and attack signatures
3. Event Profiling Stage: Processed data will be splitted into train and test model based on profiling events.
4. Deep Learning Neural Network Model: This module runs CNN and LSTM algorithms on train and test data and then generate a training model. Generated trained model will be applied on test data to calculate prediction score, Recall, Precision and FMEA sure. Algorithm will learn perfectly will yield better accuracy result and that model will be selected to deploy on real system for attack detection.

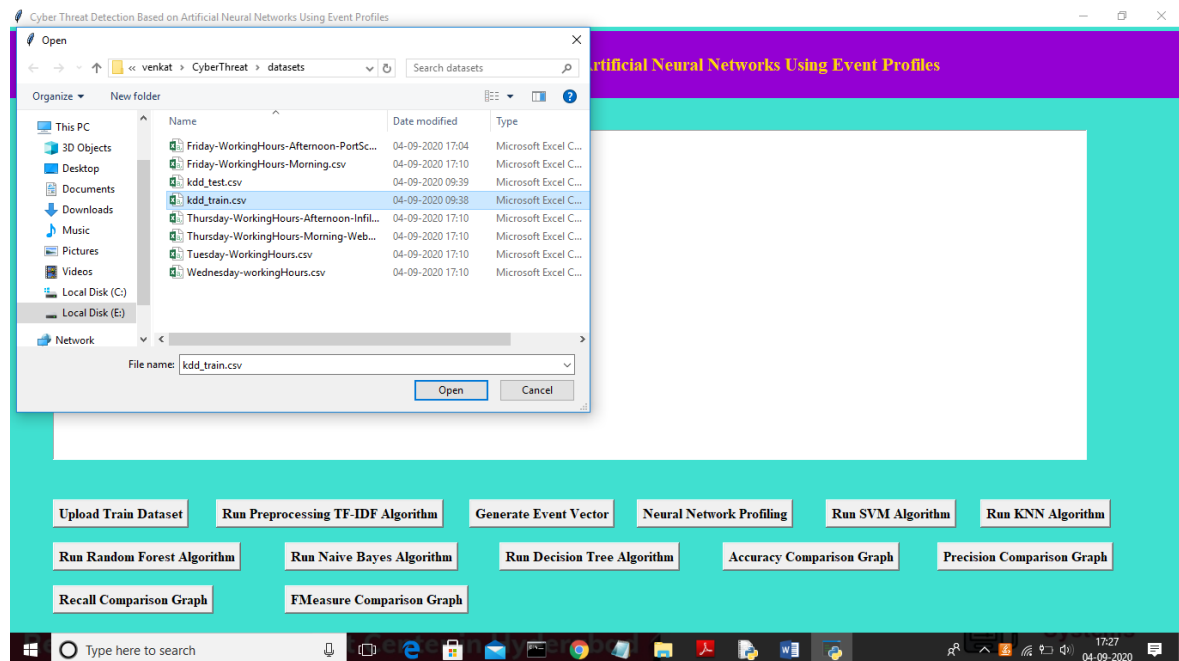
4.5 Module Description

To build the complete prediction model, it is essential to carry out step-by-step execution of all required modules. To run project double, click on 'run.bat' file to get below screen.

4.5.1 UPLOAD DATASETS



In above screen click on 'Upload Train Dataset' button and upload dataset

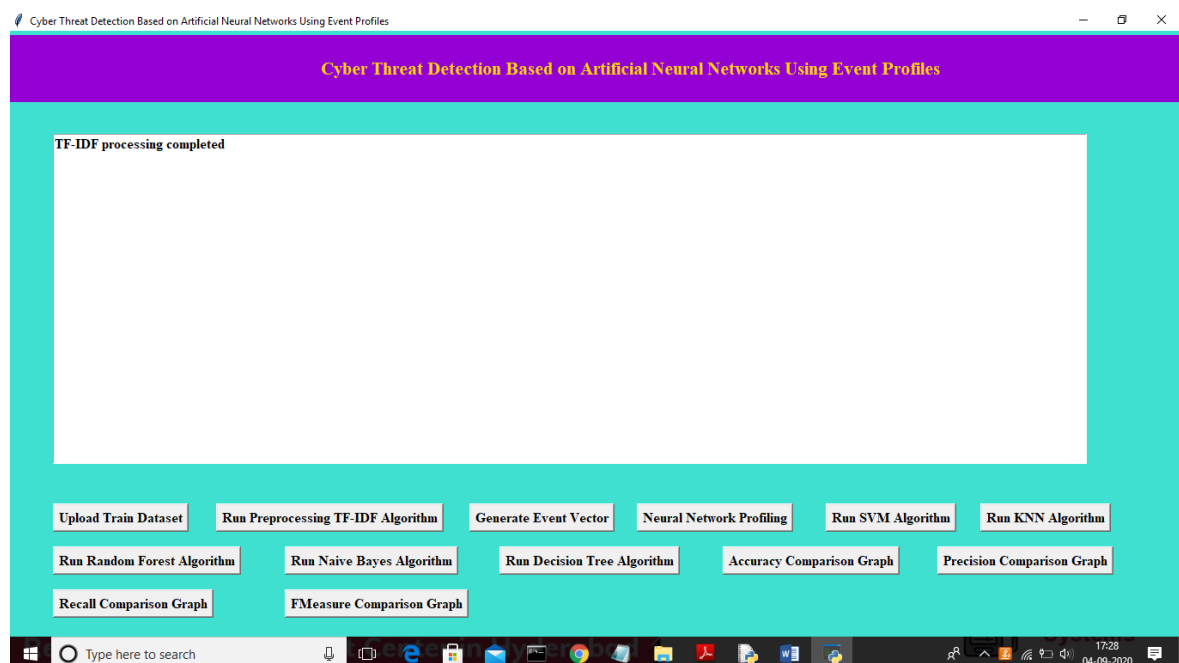


In above screen uploading 'kdd_train.csv' dataset and after upload will get below screen.

4.5.2 TF-IDF ALGORITHM

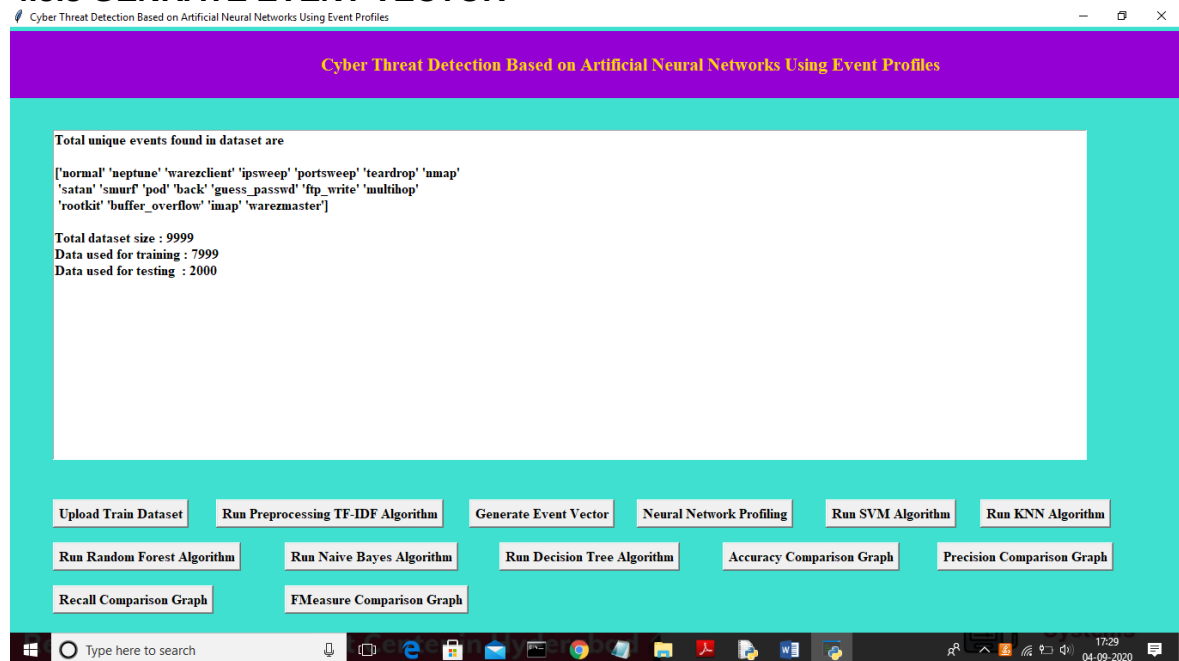


In above screen we can see dataset contains 9999 records and now click on 'Run Preprocessing TF-IDF Algorithm' button to convert raw dataset into TF-IDF values.



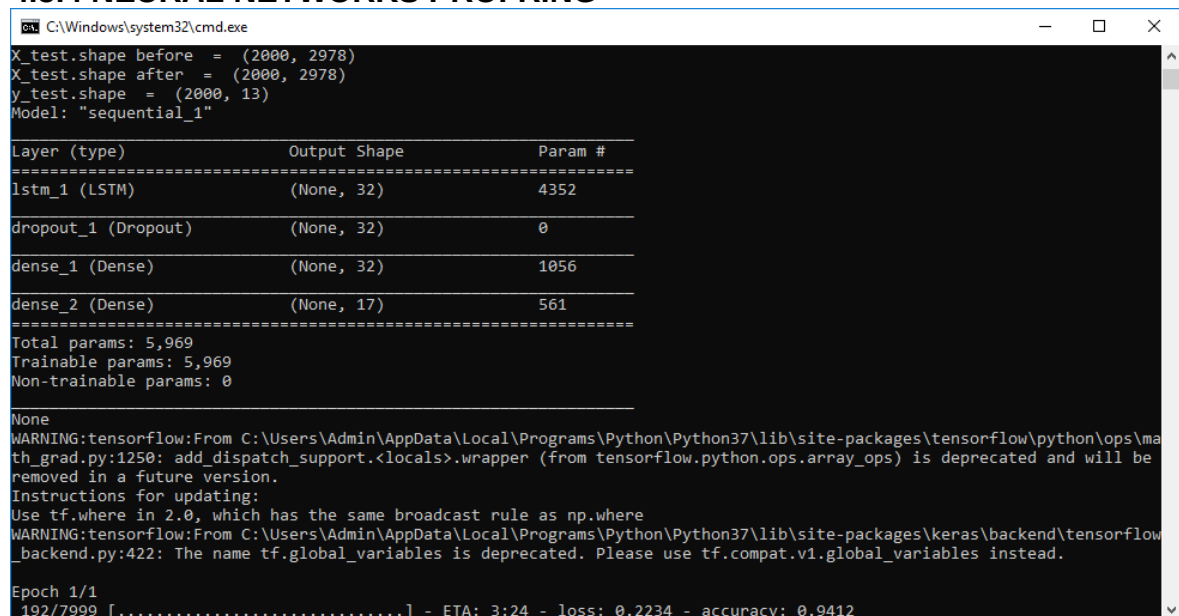
In above screen TF-IDF processing completed and now click on 'Generate Event Vector' button to create vector from TF-IDF with different events

4.5.3 GENRATE EVENT VECTOR



In above screen we can see totally different unique events names and in below we can see dataset total size and application using 80% dataset (7999 records) for training and using 20% dataset (2000 records) for testing. Now dataset train and test events model ready and now click on 'Neural Network Profiling' button to create LSTM and CNN model.

4.5.4 NEURAL NETWORKS PROFRING



In above screen LSTM model is generated and its epoch running also started and its starting accuracy is 0.94. Running for entire dataset may take time so wait till LSTM and CNN training process completed. Here dataset contains 7999 records and LSTM will iterate all records to filter and build model.

```

C:\Windows\system32\cmd.exe
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: UndefinedMetricWarning: 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

```

In above selected text we can see LSTM complete all iterations and in below lines we can see CNN model also starts execution.

4.5.5 SVM ALGORITHM

```

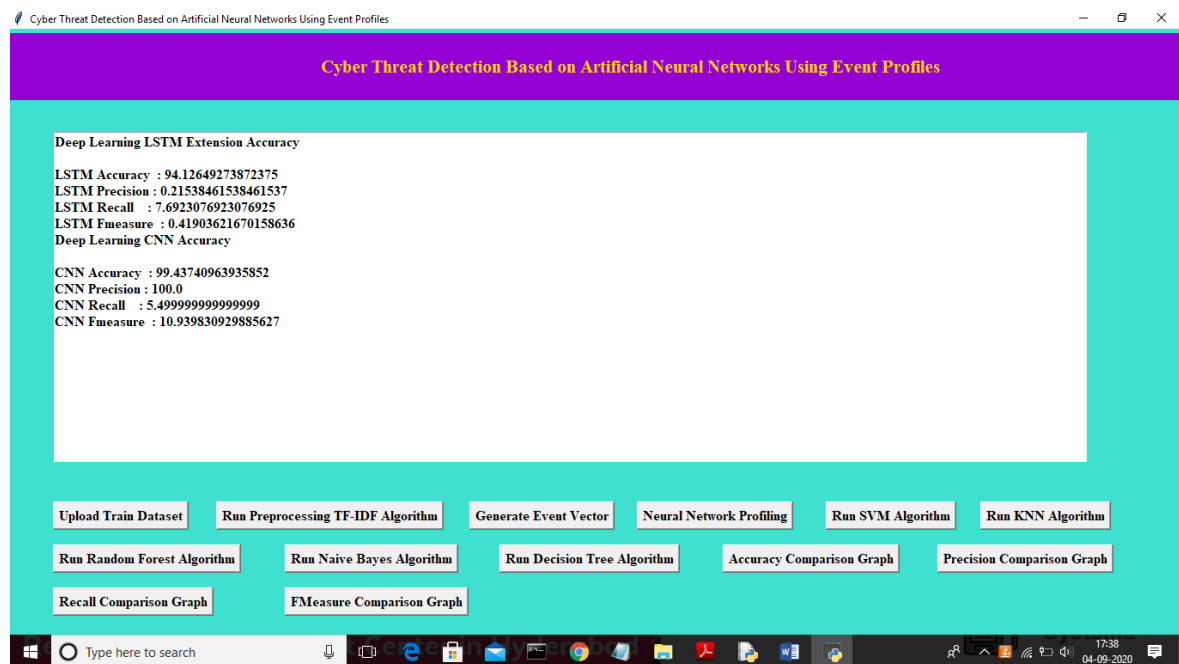
C:\Windows\system32\cmd.exe
activation_3 (Activation)    (None, 17)                0
=====
Total params: 1,796,625
Trainable params: 1,796,625
Non-trainable params: 0

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

```

In above screen CNN also starts first iteration with accuracy as 0.72 and after completing all iterations 10 we got filtered improved accuracy as 0.99 and multiply by 100 will give us 99% accuracy. So, CNN is giving better accuracy compare to LSTM and now see below GUI screen with all details.

4.5.6 KNN ALGORITHM



The screenshot shows a web application titled "Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles". The interface displays performance metrics for two models:

Deep Learning LSTM Extension Accuracy

- LSTM Accuracy : 94.12649273872375
- LSTM Precision : 0.21538461538461537
- LSTM Recall : 7.6923076923076925
- LSTM Fmeasure : 0.41903621670158636

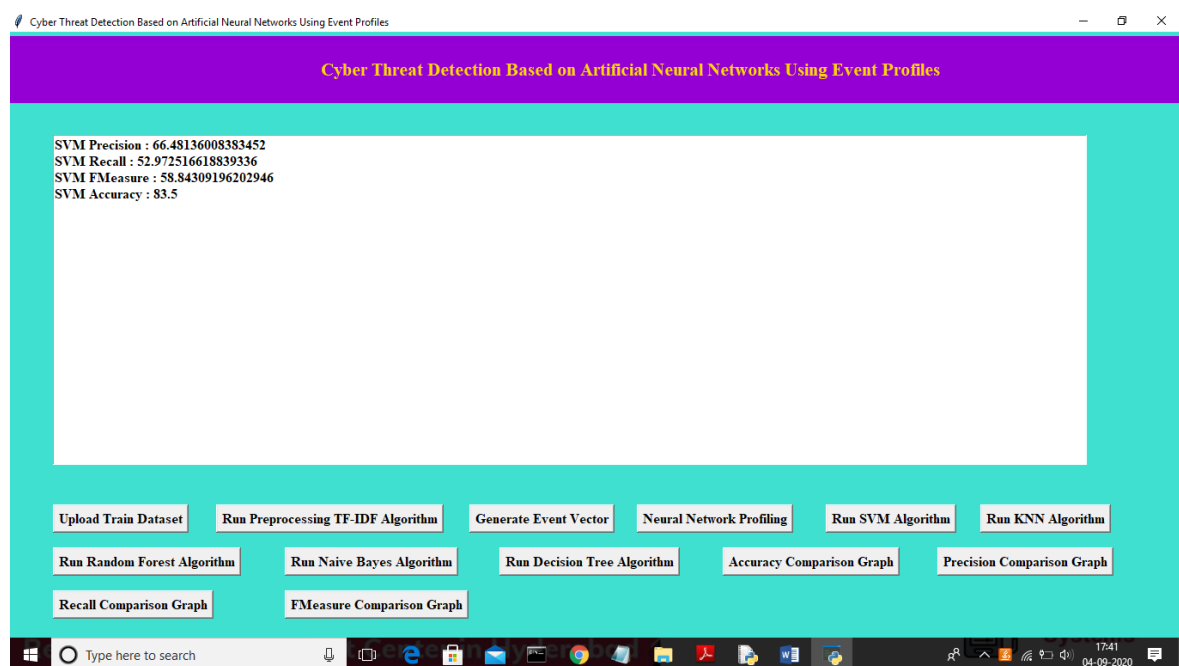
Deep Learning CNN Accuracy

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

Below the metrics, there are several buttons for running different algorithms and generating comparison graphs:

- 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

In above screen we can see both algorithms accuracy, precision, recall and FMEASURE values. Now click on 'Run SVM Algorithm' button to run existing SVM algorithm.



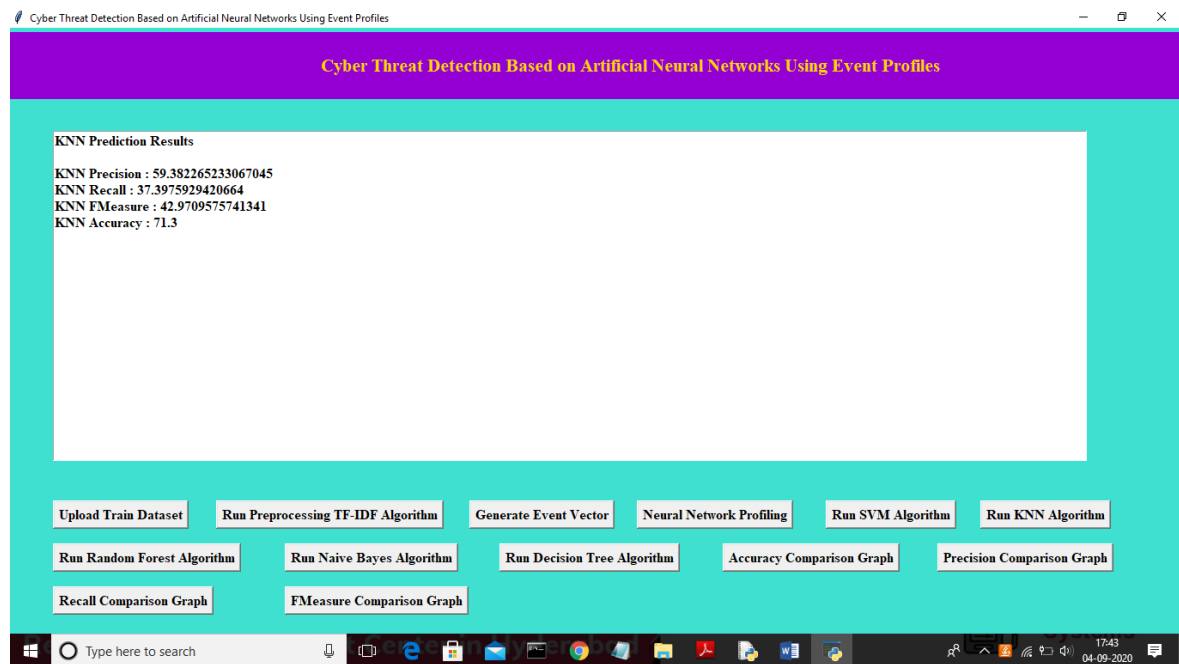
The screenshot shows the same web application after clicking the "Run SVM Algorithm" button. The interface displays the following SVM metrics:

- SVM Precision : 66.4813600833452
- SVM Recall : 52.972516618839336
- SVM FMeasure : 58.84309196202946
- SVM Accuracy : 83.5

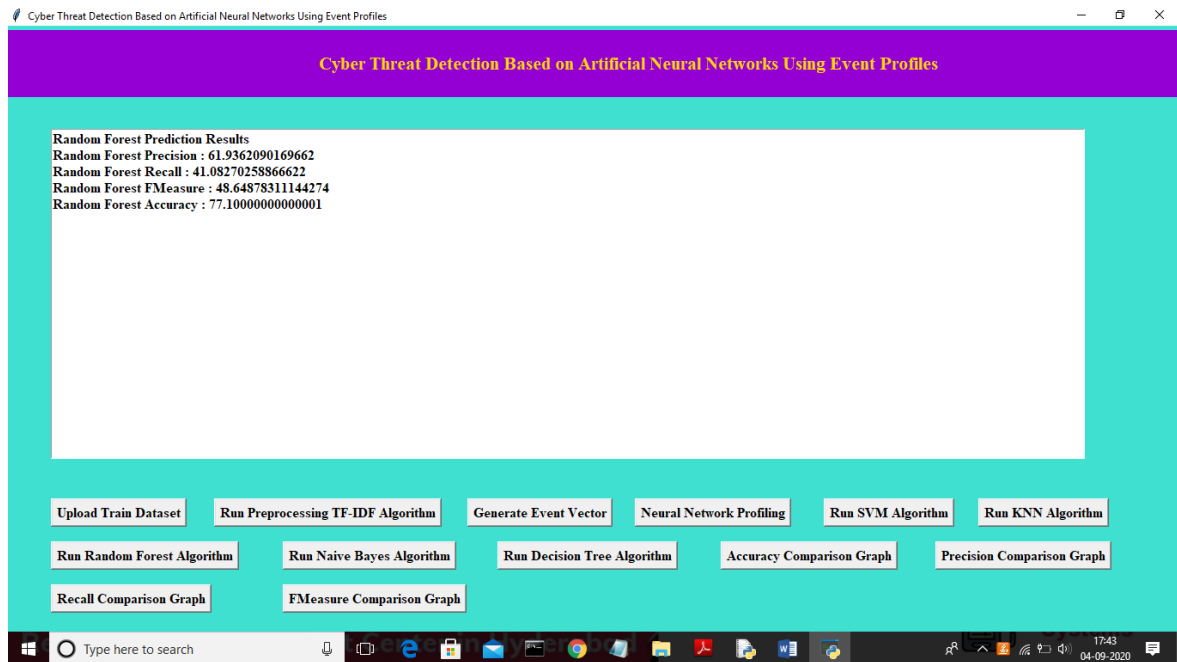
The buttons for running algorithms and generating comparison graphs remain visible at the bottom of the interface.

In above screen we can see SVM algorithm output values and now click on 'Run KNN Algorithm' to run KNN algorithm.

4.5.7 RANDOM FOREST ALGORITHM



In above screen we can see KNN algorithm output values and now click on 'Run Random Forest Algorithm' to run Random Forest algorithm.



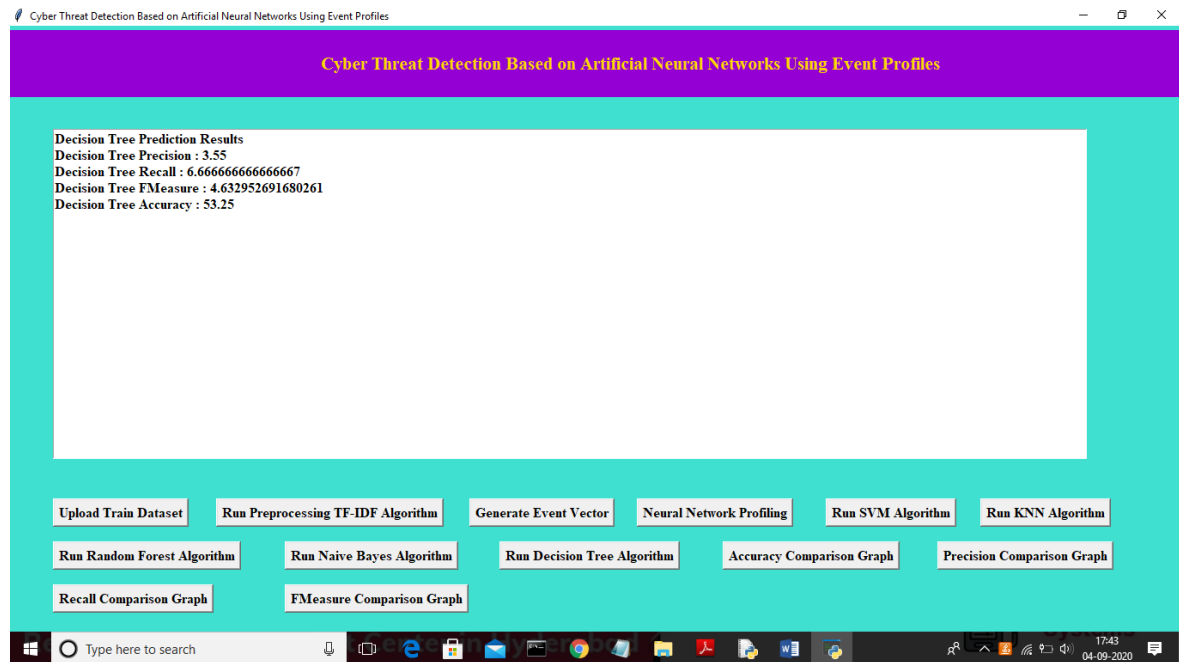
In above screen we can see Random Forest algorithm output values and now click on 'Run Naïve Bayes Algorithm' to run Naïve Bayes algorithm.

4.5.8 NAÏVE BAYES ALGORITHM

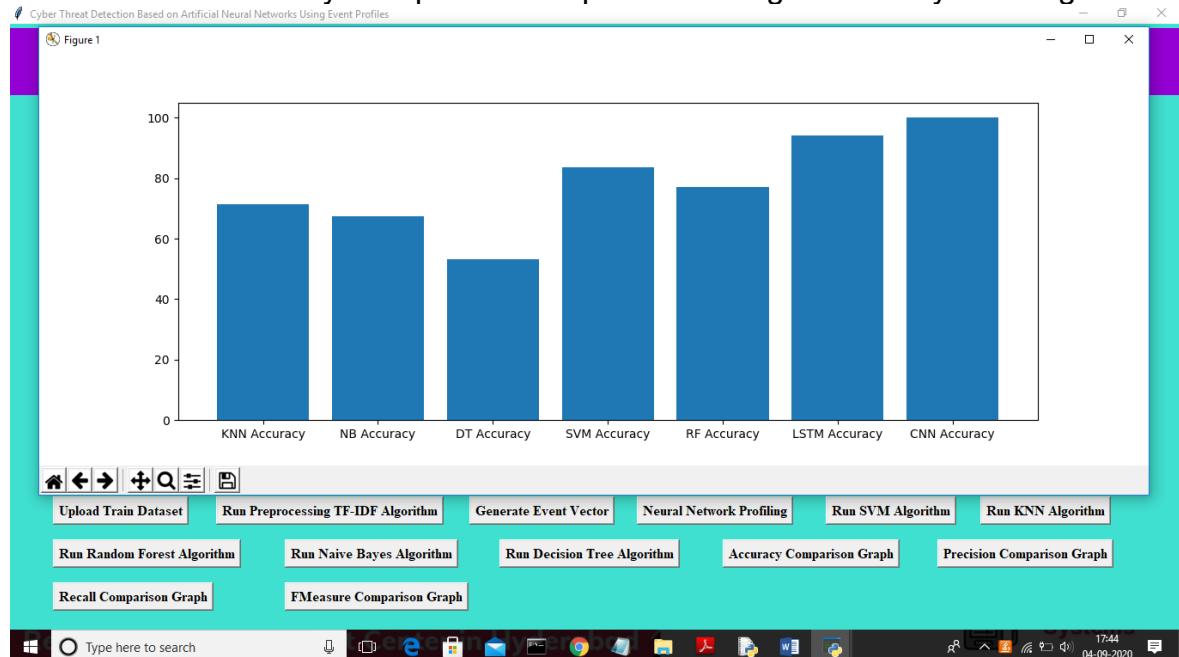


In above screen we can see Naïve Bayes algorithm output values and now click on 'Run Decision Tree Algorithm' to run Decision Tree Algorithm

4.5.9 DECISION TREE ALGORITHM

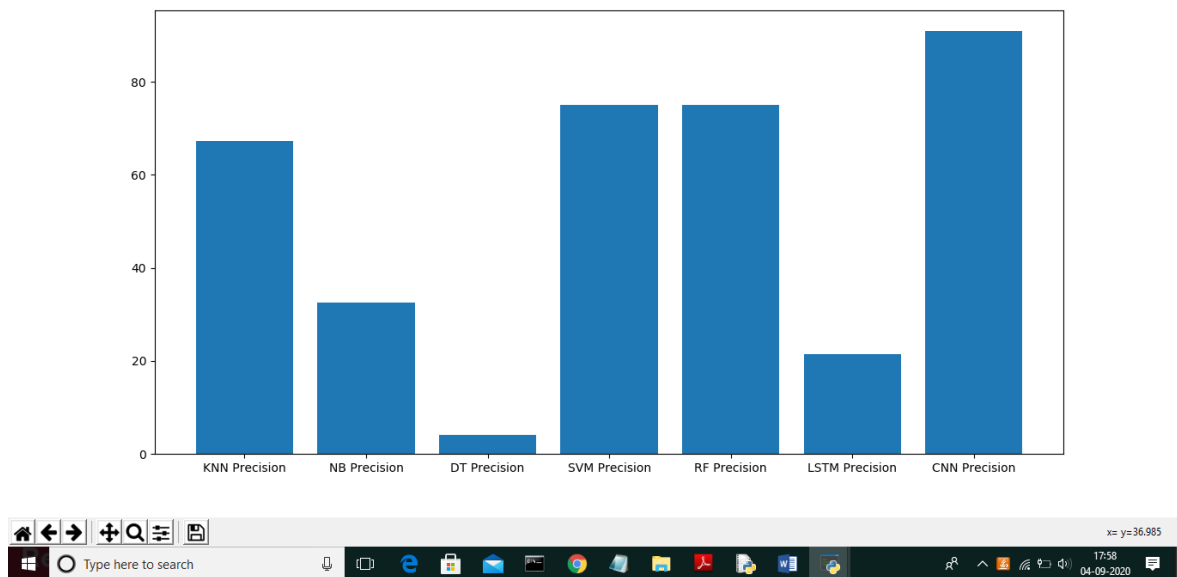


Now click on 'Accuracy Comparison Graph' button to get accuracy of all algorithms



In above graph x-axis represents algorithm name and y-axis represents accuracy of those algorithms and from above graph we can conclude that LSTM and CNN perform well. Now click on 'Precision Comparison Graph' to get below graph

Figure 1



In above graph CNN is performing well and now click on 'Recall Comparison Graph'

4.6 SYSTEM STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

- ◆ **ECONOMICAL FEASIBILITY**
- ◆ **TECHNICAL FEASIBILITY**
- ◆ **SOCIAL FEASIBILITY**

4.6.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

4.6.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

4.6.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

4.7 INPUT AND OUTPUT DESIGN

4.7.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for

processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy.

4.7.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 DATASETS

duration	protocol	service	flag	src_bytes	dst_bytes	land	wrong_fra	urgent	hot	num_fails	logged_in	num_com	root_shell	su_attempt	num_root	num_file	num_shel	num_acce	num_outt	is_ho
5	tcp	smtp	SF	2429	475	0	0	0	0	0	1	0	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
0	udp	domain_u	SF	45	80	0	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	0
14462	tcp	other	RSTR	1	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
0	tcp	http	SF	294	5499	0	0	0	0	0	1	0	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
0	icmp	ecr_i	SF	1032	0	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
0	tcp	mtpt	REJ	0	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
0	tcp	private	S0	0	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
0	tcp	http_443	S0	0	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
0	tcp	private	S0	0	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
0	tcp	smtp	SF	1201	330	0	0	0	0	0	1	0	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
0	udp	domain_u	SF	46	133	0	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	0

Our dataset has been collected from two large enterprise sys-terms, namedESX-1andESX-2. The security raw events were collected over 5 months for ESX-1, over 30 days for ESX-2, respectively, in which the detecting threat information was separately recorded by the SOC security analysts whenever network intrusion occurred. The list of threat detection information contains threat occurrence time, related attacks, category of attack, respond contents, attack IP address, and victim network information.

In our datasets, we investigated 798 detecting cyber threats in ESX-1, which are dispersed across the entire collection period. Looking at the type of occurred attacks in recorded cyber threats, there are 240 scanning, 547 system hack-king, and 11 worm attacks. Similarly, in ESX-2 there are941 scanning, 3,077 system hacking, and 51 worm attacks. This categorizing of attack type was manually performed by SOC analysts. By category, the system hacking attack includes a cross site script, DDoS, brute force attack, and injection attack. A trojan and backdoor attack belongs scanning attack. Overall, the number of attacks were found 4,079 cyber-threats.

5.2 DATA VISUALIZATION

The t-SNE is not only commonly utilized for vector data visualization but also considered as embedding tools to visualize high-dimensional data. The t-SNE is able to visual-zed high-dimensional data into two-dimensional maps by learning two-dimensional embedding vectors that preserves neighbour structures among high-dimensional data. The N data rows in dataset are randomly selected, which are visualized by performing analysis in t-SNE represent the maps that are visualized by t-SNE for CICIDS 2017 and ESX-2, respectively. The t-SNE plots in the figure show that the normal and attack data points located nearby in the same space, which makes it very hard to classify them into either normal or attack. Although the t-SNE plots of normal and attack data are clustered, it clearly finds out that those are not linearly separated. In general, it is known that deep learning is then effective at dealing with high-dimensional data with non-linearity [50], which is one of the reasons we employ deep learning approaches to detect cyber threats.

5.3 EXPERIMENTAL RESULTS

Based on the results of this experiment, we are able to arrive at two meaningful conclusions. First, our mech-amiss are capable of being employed as learning-based models for network intrusion detection. When the performance evaluations were conducted using two well-known benchmark datasets such as NSLKDD and CICIDS2017, the result proved as capable as the conventional machine-learning models. This means that our proposed methods, employed in the AI-SIEM system, have applicability for learning-based network intrusion detection. Second, when the conventional learning-based methods, which accomplish good result by bench mark dataset, are employed in the real world, the performance of overall accuracy is not as reliable as those of benchmark datasets. Never the less, the accuracy performance of our three EP-ANN models were not significantly degraded, despite the large amount of data and a lack of benchmark dataset features, such as seen in the result for ESX-2. By contrast, the accuracy of conventional methods had degraded from approximately 0.90 to 0.85.

```
IPython console
Console 2/A
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 neural networks generally, an epoch is a single pass through a full dataset. It is the iterations constituting one forward pass and one backward pass. A confusion matrix is a technique for summarizing the performance of a classification algorithm. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. Classification accuracy is the ratio of correct cyber threats. Computers do not generally store arbitrarily large numbers. Instead, each number stored by a computer is allotted a fixed amount of space. Therefore, when the number of time units that have elapsed since a system's epoch exceeds the largest number that can fit in the space allotted to the time representation, the time representation overflows, and problems can occur. While a system's behavior after overflow occurs is not necessarily predictable, in most systems the number representing the time will reset to zero, and the computer system will think that the current time is the epoch time again.

CHAPTER 6

Conclusion and Future Work

6.1 SUMMARY

With Intrusion Detection Systems and trained network security auditors, organizations have a reliable means to prioritize and isolate the most critical threats in real time. They are tasked with identifying the significant elements of the attack and translating them into IDS signatures threat detection and response is the most important aspect of cybersecurity for IT organizations that depend on cloud infrastructure. Threat detection, therefore, describes the ability of IT organizations to quickly and accurately identify threats to the network or to applications or other assets within the network.

The proposed procedure changes large number of gathered security occasions over to singular occasion profiles and utilize a profound learning-based discovery strategy for upgraded digital danger identification. For this work, we built up an AI-SIEM framework dependent on a blend of occasion profiling for information preprocessing and distinctive counterfeit neural organization techniques, including FCNN, CNN, and LSTM. The framework centers around separating between obvious positive and bogus positive cautions, consequently causing security examiners to quickly react to digital dangers.

6.2 CONCLUSION:

In this paper, we have proposed the AI-SIEM system using event profiles and artificial neural networks. The novelty of our work lies in condensing very large-scale data into event profiles and using the deep learning-based detection methods for enhanced cyber-threat detection ability. The AI-SIEM system enables the security analysts to deal with significant security alerts promptly and efficiently by comparing long term security data. By reducing false positive alerts, it can also help the security analysts to rapidly respond to cyber threats dispersed across a large number of security events.

For the evaluation of performance, we performed a performance comparison using two benchmark datasets (NSLKDD, CICIDS2017) and two datasets collected in the real world. First, based on the comparison experiment with other methods, using widely known benchmark datasets, we showed that our mechanisms can be applied as one of the learning-based models for network intrusion detection. Second, through the evaluation using two real datasets, we presented promising results that our technology also outperformed conventional machine learning methods in terms of accurate classifications.

6.3 FUTURE WORK

In the future, to address the evolving problem of cyber-attacks, we will focus on enhancing earlier threat predictions through the multiple deep learning approach to discovering the long-term patterns in history data. In addition, to improve the precision of labeled dataset for supervised-learning and construct good learning datasets, many SOC analysts will make efforts directly to record labels of raw security events one by one over several months. For testing, we constructed the purpose-built test bed where for conducting performance evaluations. This test bed consists of the big data platform and the AI-SIEM system. Moreover, in the SOC, we also had collected real-world IPS data over several months.

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 spatial-temporal 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 signature-based intrusion detection systems: A survey," *Compute. Common.*, vol. 49, pp. 1-17, 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 Al-Namrata 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. 171-182.
- [28] E. Raff, J. Sylvester, and C. Nicholas, "Learning the PE header, malware detection with minimal domain knowledge," In Proc. 10th CalWORKs Artif. 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. 50850-50859, 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.

APPENDIX

A. SCOUCE CODE

```
from tinkers import message box  
from tinkers import *  
from tinkers import simple dialog  
import tinkers  
from tinkers import file dialog  
import matplotlib. Pylon as Plt  
import NumPy as np  
from interfile dialog import ask open file name  
import OS  
import pandas as pd  
from skarn import preprocessing  
from Sklearn. feature extraction. Text  
import Count Vectorizer, Tf-idf Vectorizer  
from sklearn import Svm  
from sklearn.metrics import accuracy score  
from sklearn.model selection import train test slit  
from keras.models import Sequential
```

```
from keras.layers import Flatten
from keras.layers import Dense, Activation, Dropout
from sklearn. pre-processing import One Hot Encoder
import skleras. Layers
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers
import Dense, Activation, Batch Normalization, Dropout
```

```
from sklearn.metrics import precision score
from sklearn.metrics import recall score
from sklearn.metrics import f1 score
```

```
from sklearn.naive bayes import Bernoulli
from sklearn.neighbors import K Neighbors Classifier
from sklearn.tree import Decision Tree Classifier
from sklearn.ensemble import Random Forest Classifier
```

```
main = tkinter.Tk()
main. Title ("Cyber Threat Detection Based on Artificial Neural
Networks Using Event Profiles") #designing main screen
main. Geometry("1300x1200")
```

```
le = pre-processing. Label Encoder ()
global filename
global feature extraction
```

```

global X, Y
global doc
global label names
global Train, X test, y train, ytest
global lstm acc,cnn acc,svm acc,knn acc,dt acc,random acc,nb
acc
global lstm_precision,cnn_precision,svm_precision,knn_precision
dt_precision,random_precision,nb_precision

global lstm_recall,cnn_recall,svm_recall,knn_recall,dt_acc,do_real
l,nb_recall
global lstm_fm,cnn_fm,svm_fm,knn_fm,dt_fm,random_fm,nb_fm

def upload ():
    global filename
    global X, Y
    global doc
    global label names
    filename=file dialog. Ask open filename(initial dirt="datasets")
    dataset = pd.read_csv(filename)
    label names = dataset.labels.unique()
    dataset['labels'] = le.fit_transform(dataset['labels'])
    cols = dataset. Shape [1]
    cols = cols - 1
    X = dataset. Values [: 0: cols]
    Y = dataset. Values [: cols]
    Y = Y.astype('int')
    doc = []

```

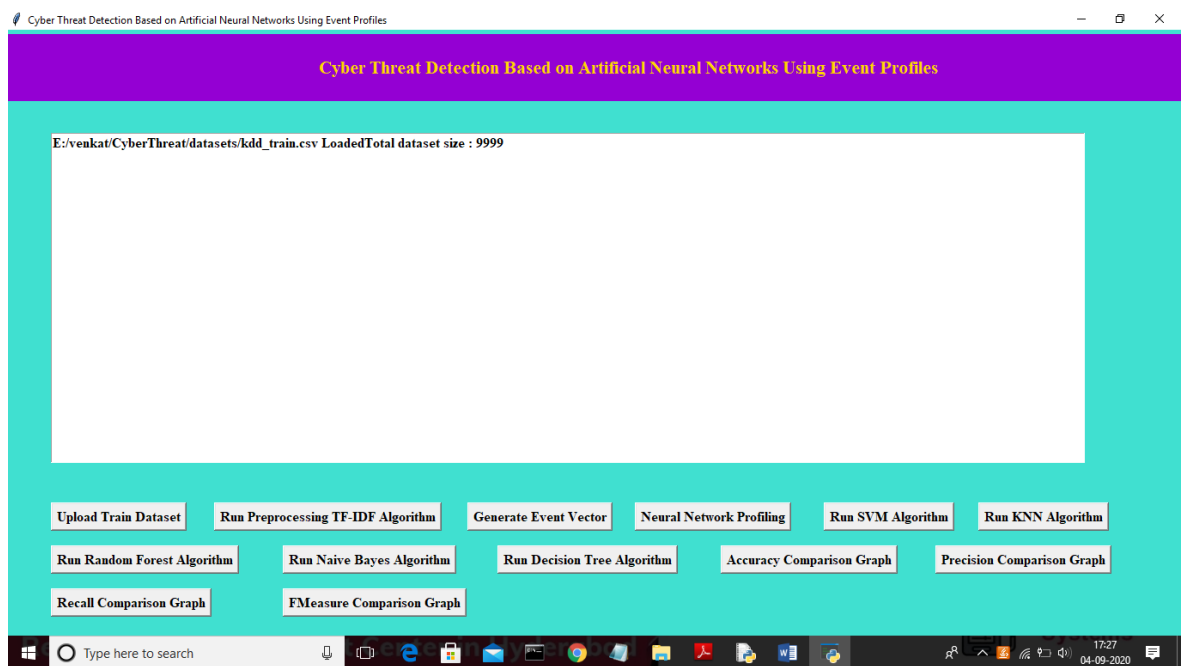
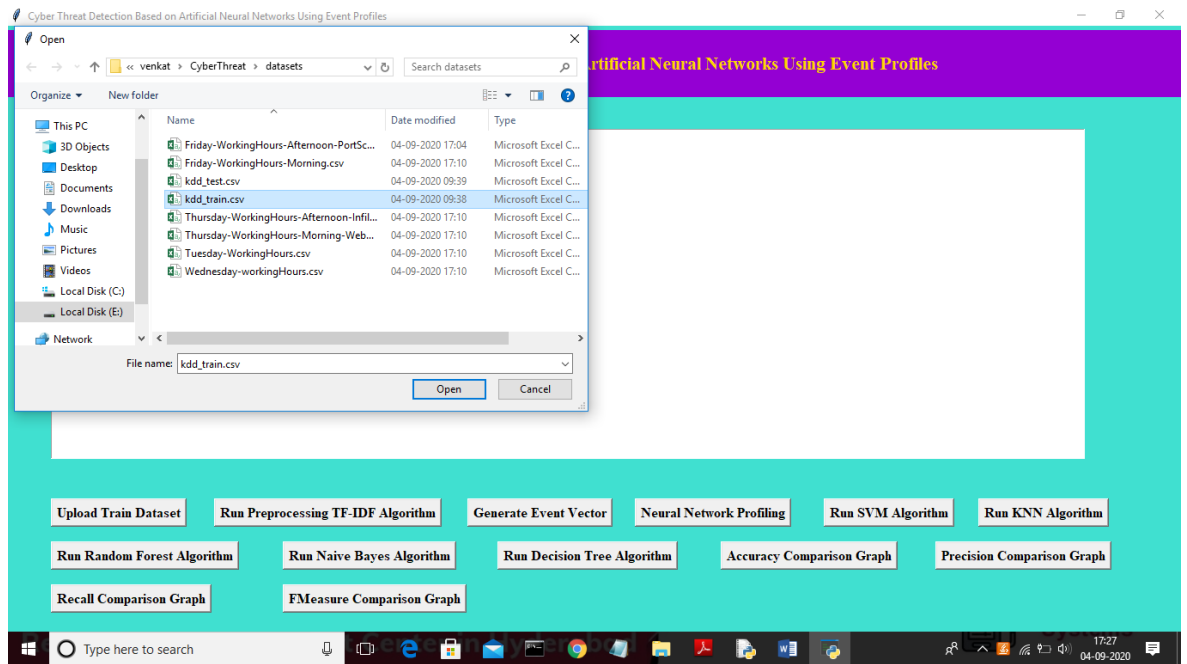
```

for l in range (Len(X)):
    str1 = "for j in range (Len(X[l])):
    str1+=str(X[l,j])+ " " doc.append(str1.strip())

```

B. SCREEN SHORTS

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	duration	protocol	service	flag	src_bytes	dst_bytes	land	wrong_fr	urgent	hot	num_fail	logged_in	num_com	root_shell	su_attempt	num_root	num_file	num_shel	num_acce	num_out	is_hos
2	5	tcp	smtp	SF	2429	475	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	0	udp	domain_u	SF	45	134	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	udp	domain_u	SF	45	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1979	udp	other	SF	145	105	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	14462	tcp	other	RSTR	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	tcp	http	SF	294	5499	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
9	0	tcp	http	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	icmp	ecr_l	SF	1032	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	udp	private	SF	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	tcp	mtp	REJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	tcp	netstat	RSTR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	tcp	http_443	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	tcp	http	SF	291	670	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
18	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	tcp	http	SF	340	851	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
20	0	tcp	smtp	SF	1201	330	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
21	0	tcp	http	SF	192	3717	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
22	0	udp	domain_u	SF	46	133	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	tcp	http	SF	234	13713	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0



```
C:\WINDOWS\system32\cmd.exe
None
Epoch 1/1
1999/7999 [*****] - 348s 43ms/step - loss: 0.0880 - accuracy: 0.9706
-----0 0.97057
C:\Users\goutham\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning: 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) 1528832
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, 34) 17442
activation_3 (Activation) (None, 34) 0
-----
Total params: 1,808,930
Trainable params: 1,808,930
Non-trainable params: 0
None
Train on 6399 samples, validate on 1600 samples
Epoch 1/10
- 4s - loss: 1.7123 - accuracy: 0.6445 - val_loss: 0.9032 - val_accuracy: 0.7725
Epoch 2/10
- 4s - loss: 0.7405 - accuracy: 0.7940 - val_loss: 0.6511 - val_accuracy: 0.8231
Epoch 3/10
- 4s - loss: 0.5053 - accuracy: 0.8553 - val_loss: 0.4540 - val_accuracy: 0.8763
Epoch 4/10
- 4s - loss: 0.3282 - accuracy: 0.9175 - val_loss: 0.3341 - val_accuracy: 0.9169
Epoch 5/10
- 4s - loss: 0.2234 - accuracy: 0.9430 - val_loss: 0.2755 - val_accuracy: 0.9269
```

S

Figure 1

