A Project Report on

Application of Robust Software Modelling Tool for Web Attacks Detection

A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

Bachelor of Technology

In

Computer Science and Engineering

Submitted by

K. Rishita
(20H51A0535)

L. Shriya
(20H51A05E5)

B. Ganesh
(20H51A05B1)

Under the esteemed guidance of

Mr. A. Vivekanand
(Assistant Professor, Department of CSE)

Department of Computer Science and Engineering

CMR COLLEGE OF ENGINEERING & TECHNOLOGY
(UGC Autonomous)
*Approved by AICTE *Affiliated to JNTUH *NAAC Accredited with A^+ Grade

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

2020- 2024

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Major Project report entitled **"APPLICATION OF ROBUST SOFTWARE MODELLING TOOL FOR WEB ATTACKS DETECTION"** being submitted by Kalluri Rishita (20H51A0535), Lanka Shriya (20H51A05E5), Balla Ganesh (20H51A05B1) in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out his/her under my guidanceand supervision.

The results embodies in this project report have not been submitted to any other University or Institute for the award of any Degree.

Mr. A. Vivekanand Assistant Professor Dept. of CSE	Dr. Siva Skandha Sanagala Associate Professor and HOD Dept. of CSE	EXTERNAL EXAMINER

ACKNOWLEDGEMENT

With great pleasure we want to take this opportunity to express my heartfelt gratitude to allthe people who helped in making this project work a grand success.

We are grateful to **Mr. A. Vivekanand, Assistant Professor** , Department of Computer Science and Engineering for his valuable technical suggestions and guidance during the execution of this projectwork.

We would like to thank **Dr. Siva Skandha Sanagala,** Head of the Department of Computer Science and Engineering, CMR College of Engineering and Technology, who is the major driving forces to complete my project work successfully.

We are very grateful to **Dr. Ghanta Devadasu**, Dean-Academics, CMR College of Engineering and Technology, for his constant support and motivation in carrying out the project work successfully.

We are highly indebted to **Major Dr. V A Narayana,** Principal, CMR College of Engineering and Technology, for giving permission to carry out this project in a successful and fruitful way.

We would like to thank the **Teaching & Non- teaching** staff of Department of Computer Science and Engineering for their co-operation

We express our sincere thanks to **Shri. Ch. Gopal Reddy**, Secretary& Correspondent, CMR Group of Institutions, and **Shri Ch Abhinav Reddy**, CEO, CMR Group of Institutions for their continuous care and support.

Finally, We extend thanks to our parents who stood behind us at different stages of this Project. We sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this project work.

K. Rishita - 20H51A0535
L. Shriya - 20H51A05E5
B. Ganesh - 20H51A05B1

TABLE OF CONTENTS

List of Figures

List of Tables

ABSTRACT

Web applications are popular targets for cyber-attacks because they are network- accessible and often contain vulnerabilities. An intrusion detection system monitors web applications and issues alerts when an attack attempt is detected. Existing implementations of intrusion detection systems usually extract features from network packets or string characteristics of input that are manually selected as relevant to attack analysis. Manually selecting features, however, is time-consuming and requires in-depth security domain knowledge. Moreover, large amounts of labeled legitimate and attack request data are needed by supervised learning algorithms to classify normal and abnormal behaviors, which is often expensive and impractical to obtain for production web applications.

The current inquiry as an examination of web attacks where proliferation of web-based applications has brought about a concurrent rise in cyber threats, particularly the form of web attacks targeting vulnerable systems. Approaches to web attack detection often rely on rule-based or signature-based methods, which struggle to change with the increasing landscape of attacks. In response, this study proposes an innovative approach leveraging DL techniques for web attacks. By harnessing the capability of DL, especially CNN and recurrent neural networks (RNNs), our proposed system learns directly from raw web traffic data, eliminating the need for manual feature engineering. This end-to-end approach not only streamlines the detection process but also enhances the system's ability to generalize across different types of attacks and adapt to new threats. To evaluate the effectiveness of our approach, we conducted extensive experiments on diverse datasets containing both benign and malicious web traffic. Our results demonstrate the superiority of end-to-end deep learning over traditional methods, achieving higher detection accuracy and robustness against adversarial attacks. In conclusion, our study highlights the promise of end-to-end deep learning as a viable approach related to web attacks, offering enhanced detection capabilities in the phase of evolving cyber threats.

CHAPTER 1
INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

Web attack detection refers to the process of identifying and preventing unauthorized and malicious activities aimed at web applications and their users. It involves deploying various security mechanisms, tools, and techniques to recognize patterns or behaviors indicative of an attack. These attacks can be broadly categorized into server-side attacks, where the attacker exploits vulnerabilities in the web server or application, and client- side attacks, where the attacker targets the end-user's browser or device.

Halfond, W. G., Viegas, J., & Orso, A [1], Web applications are vulnerable to cyber attacks, with common attacks including SQL injection cross-site scripting Wassermann,G., & Su, Z. [2], and remote code execution. Despite the development of counter- measures like firewalls and intrusion detection systems Di Pietro, R., & Mancini, L. V [3], web attacks remain a significant threat. Research shows that over half of web applications during a 2015-2016 scan contained significant security vulnerabilities. False positive limitations Pietraszek, T. [9] require manual selection of attack-specific features and high false positive rates, making it essential to reduce these systems. An infrastructure that requires less expertise and labeled training data is needed to address these challenges. Fu, X., Lu, X., Peltsverger, B., Chen, S., Qian, K., & Tao, L [12] struggle due to workforce limitations, classification limitations, and false positive limitations.

.

1.2 PROBLEM STATEMENT

- Our proposed system deals with different types of web attack such as DoS attack, Probe attack, U2R, R2L, normal attack and algorithms like Logistic Regression, Decision tree, SVM, GNB, GRU, LSTM, CNN, RNN are used where Decision Tree, Logistic Regression, SVM, GNB comes under machine learning algorithms and GRU, RNN, CNN, LSTM comes under deep learning algorithm which require large, labeled datasets for supervised learning. Mitigating Economic Impact: Bots can impact businesses and markets

1.3 PROBLEM OBJECTIVE

In this project we are mainly addressing 5 objectives that are:

- DoS attack (Denial of Service) attack is an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with traffic, or sending it information that triggers a crash.
- Probe attacks are aimed at gathering information about the target network from a source that is often external to the network. Denial-of-Service (DoS) attacks results in an interruption of the service by flooding the target system with illegitimate requests.
- U2R attack, short for "User to Root" attack, is a type of security exploit where an attacker starts with access to a regular user account on a system and escalates their privileges to gain root or administrator-level access. This type of attack typically involves exploiting vulnerabilities in the system or applications to execute arbitrary code with elevated privileges, thus gaining control over the entire system. U2R attacks are a significant threat to system security as they can provide attackers with full control over the targeted system, allowing them to install malware, steal sensitive data, or carry out other malicious activities.
- An R2L (Remote to Local) attack is a type of security breach where an attacker tries to gain unauthorized access to a computer system from a remote location. They exploit vulnerabilities in network services or applications to enter the system. Once inside, attackers may aim to steal data, escalate privileges, or install malware. Common R2L attack methods include brute-force attacks to guess passwords or exploiting software vulnerabilities. R2L attacks pose a serious threat to system security and are often used by hackers to compromise sensitive information or disrupt operations.
- A "normal" attack doesn't have a standard definition in cybersecurity terminologies. However, in a general context, it could refer to any unauthorized or malicious activity targeted towards compromising the security of a system, network, or device. This could encompass a wide range of attack vectors, including but not limited to malware infections, phishing attempts, denial-of-service attacks, data breaches, and social engineering tactics. The term "normal" might imply attacks that are common or typical in nature, as opposed to highly sophisticated or specialized attacks.

1.4 PROJECT SCOPE

1.4.1 SCOPE OF PROJECT

This project explores an unsupervised/semi-supervised intrusion detection approach using RSMT, emphasizing efficient detection of web application attacks, including DoS attack, R2L attack, U2R attack, Normal attack and probe attacks. There are several other attacks like Penetration attack, Phishing attack etc. but our project cannot predict such attacks.

1.4.2 LIMITATIONS OF PROJECT

Address limitations regarding the availability, diversity, and quality of the sequential web traffic data. Complex models often require significant computational resources and longer training times. Discuss potential challenges related to computational limitations. Machine learning algorithms can struggle with high dimensionality, requiring extensive preprocessing and feature engineering. They may also overfit the training data, generalize poorly to unseen data, and lack interpretability, making it challenging to understand their decision-making process. Additionally, they can be computationally intensive and require large amounts of labeled data for training, which may not always be readily available. Deep learning models require large amounts of data and computational resources for training, are often treated as black boxes with limited interpretability, and can suffer from overfitting and vulnerability to adversarial attacks. They also struggle with capturing causality, transferring knowledge between domains, and may perpetuate biases present in the training data.

CHAPTER 2
BACKGROUND WORK

CHAPTER 2

BACKGROUND WORK

In this section we have studied various implementations of web attack detections & we summarized our findings that we concluded by researching & referencing various papers. They are as below:

Halfond, W. G., Viegas, J., & Orso, A. (2006, March) [1] SQL injection attacks pose a significant security threat to web applications, allowing attackers to access sensitive information. Current methods either fail to address the full scope of the problem or have limitations. This paper reviews different types of SQL injection attacks, discusses detection and prevention techniques, and discusses their strengths and weaknesses in addressing the entire range of attacks. Future evaluations should focus on assessing techniques' precision and effectiveness in practice, using empirical evaluations to compare their performance against real-world attacks and legitimate inputs. Wassermann, G., & Su, Z. (2008, May) [2] presents a static analysis for identifying cross- site scripting (XSS) vulnerabilities in web applications. It addresses weak or absent input validation, combining work on tainted information flow with string analysis. The approach addresses the difficulty of checking for vulnerabilities statically by formalizing a policy based on the W3C recommendation, Firefox source code, and online tutorials about closed-source browsers. The paper provides effective checking algorithms and an extensive evaluation of known and unknown vulnerabilities in real-world web applications. Di Pietro, R., & Mancini, L.

V. (Eds.) [3] Anomaly-based network intrusion detection systems (NIDSs) are increasingly effective in detecting attacks, as they focus on packet headers and payloads. A comparison between PAYL and POSEI-DON, two payload-based NIDSS, ispresented to support this thesis.

Qie, X., Pang, R., & Peterson, L. (2002) [4] presents a toolkit to enhance code robustness against DoS attacks. It suggests that software development should focus on implementing protection mechanisms into the code itself, rather than reacting to attacks. The toolkit provides an API for programmers to annotate their code, acting as sensors and actuators for resource abuse detection. Ben-Asher, N., & Gonzalez, C. (2015) [5] explores the impact of knowledge in network operations and information security on detecting intrusions in a simple network. A simplified Intrusion Detection System (IDS) was developed to examine how individuals with or without

knowledge detect malicious events. Results showed that more knowledge in cyber security improved the detection of malicious events and reduced false classifications. However, knowledge about a specific network was needed for accurate detection decisions. Expertise and practical knowledge are crucial in triage analysis, which classifies network events as threats and their connections to overall attack decisions, likely driven by the accumulation of information in cyber security. Japkowicz, N., & Stephen, S. (2002) [6] explores the class imbalance problemin machine learning, focusing on understanding its nature, comparing various re-sampling methods, and examining its impact on other classification systems like Neural Networks and Support Vector Machines. It also explores the relationship between concept complexity, training set size, and class imbalance level. Liu, G., Yi, Z., & Yang, S. (2007) [7] Existing intrusion detection models mainly detect misuse or anomaly attacks. A hierarchical model using principal component analysis (PCA) neural networks is proposed, achieving satisfactory performance in classifying network connections based on 1998 DARPA evaluation data sets.

Xu, X., & Wang, X. (2005, July) [8] proposes a novel adaptive intrusion detection method using principal component analysis (PCA) and support vector machines (SVMs). PCA reduces network data patterns dimension, while SVMs construct classification models. The method has good classification performance without parameter tuning, and is superior to SVMs without PCA in training and detection speed. Experimental results show its effectiveness. Pietraszek, T.(2004) [9] Intrusion Detection Systems (IDSs) are used to detect security violations, but they often trigger false positives, making it difficult for analysts to identify true positives. This paper introduces ALAC, an Adaptive Learner for Alert Classification, which helps reduce false positives in intrusion detection by classifying alerts into true positives and false positives. ALAC can also process autonomously high-confidence alerts, reducing the analyst's workload. The prototype implementation and machine learning technique are described. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017) [10] A deep convolutional neural network was trained to classify 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest, achieving top-1 and top-5 error rates of 37.5% and 17.0%, respectively. The network, with 60 million parameters and 650,000 neurons, used non-saturating neurons and efficient GPU implementation. The model also won the ILSVRC-2012 competition with a top-5 error rate of 15.3%.

Below table represents few other references regarding our study on end-end deep learning on

web attacks, where we have studied different domines related to edge devices, cyber attack detection techniques, tools, techniques, and methodology of developing robust software, Long short-term memory, End-to-end deep learning of optimization heuristics and many more.

Table 2.1: Comparison matrix table for various research papers studied

Reference	Author	Title	Year of Publishing	Results
[14]	Tian, Z., Luo, C., Qiu, J., Du, X., & Guizani, M	A distributed deep learningsystem for web attack detection on edge devices	2019	Accuracy- 99.410% TPR - 98.91% Detection rate - 99.55%
[15]	Dutta, V., Choraś, M., Pawlicki, M., &Kozik, R	A deep learning ensemble fornetwork anomaly and cyber- attack detection	2020	The proposed framework,combining DNN, LSTM, and a meta-classifier, outperformed existing methods in detecting anomalies on three diverse datasets, eliminating the need for recent network traffic datasets.
[16]	Jayaswal, B. K., & Patton, P. C.	Design for trustworthy software: Tools, techniques,and methodology of developing robust software.	2006	

[17]	Hochreiter , S., & Schmidhu ber, J.	Long short-term memory	1997	LSTM outperforms real-time recurrent learning, back propagation, and Elman nets in experiments with artificial data, solving complex, long-time-lag tasks that previous algorithms have struggled with.
[18]	Cummins, C., Petoumenos, P., Wang, Z., & Leather,H.	End-to-end deep learning ofoptimization heuristics	2017	The network improves model accuracy by 14% and 12% without human intervention, compared to hand-picked features.
[19]	Kendall, K. K. R.	A database of computer attacks for the evaluation ofintrusion detection systems	1999	The 1998 DARPA intrusion detection evaluation established thefirst standard corpus for evaluating computer intrusion detection systems, analyzing over 300 attacks from 32 types and 7 scenarios.
[20]	Ng, A.	Sparse autoencoder	2011	The notes discuss feedforward neural networks, backpropagation algorithm for supervisedlearning, autoencoder construction, and sparse autoencoder development, utilizing notation and symbols forclarity.

The above table, combining DNN, LSTM, and a meta-classifier, outperformed existing methods in detecting anomalies on three datasets, eliminating the need for recent network traffic datasets. It improved model accuracy by 14% and 12% without human intervention. The 1998 DARPA intrusion detection evaluation established the first standard corpus. Existing implementations of intrusion detection systems usually extract features from network packets or stringcharacteristics of input that are manually selected as relevant to attack analysis. Manually selecting features, however, is time-consuming and requires in-depth security domain knowledge. Moreover, large amounts of labeled legitimate and attack request data are needed by supervised learning algorithms to classify normal and abnormal behaviors, which is often expensive and impractical to obtain for production web applications. While cross validation is widely used in traditional machine learning, it is often not used for evaluating deep learning models because of the great computational cost.

Zargar, S. [21] In this paper they have discusses about the ceaseless flourishing of the financial advertise, MasterCard volume has until the end of time been impacting these a long time. The blackmail organizations are moreover rising rapidly. Beneath this circumstance, blackmail disclosure has turned into an progressively more critical issue. Be that as it may, the degree of the distortion is completely much lower than the virtuoso trade, so the unevenness dataset makes this issue altogether more testing. In this paper we mainly prompt how to adjust to the Visa distortion distinguishing proof issue by utilizing supporting procedures and moreover gave a commitment of the brief examination between these making a difference methods.

Liu, J., Kantarci, B., & Adams, C. [22] In this paper they have discussed about wide selection of Web of Things (IoT) gadgets and applications experiences security vulnerabilities as barricades. The heterogeneous nature of IoT frameworks avoids common benchmarks, such as the NSL-KDD dataset, from being utilized to test and confirm the execution of distinctive Organize Interruption Location Frameworks (NIDS). In arrange to bridge this crevice, in this paper, we look at particular assaults within the NSL-KDD dataset that can affect sensor hubs and systems in IoT settings. Moreover, in arrange to identify the presented assaults, we consider eleven machine learning calculations and report the comes about. Through numerical investigation, we appear that tree-based strategies and gathering strategies outflank the rest of

the examined machine learning strategies. Among the supervised algorithms, XGBoost positions the primary with 97curacy, 90.5% Matthews relationship coefficient (MCC), and 99.6% Region Beneath the Bend (AUC) execution. In addition, a striking inquire about findingof this ponder is that the Expectation-Maximization (EM) calculation, which is an unsupervisedstrategy, too performs reasonably well within the location of the assaults within the NSL-KDDdataset and outflanks the exactness of the Naïve Bayes classifier by 22.0%.

Bisong, E. [23] This paper deals about the Education organized to create the information of machine learning, profound learning, information science, and cloud computing effortlessly open Prepares you with aptitudes to construct and send large - scale learning models on Google Cloud Stage Covers the programming abilities fundamental for machine learning and profound learning modeling utilizing the Python stack Incorporates bundles such as Numpy, Pandas, Matplotlib, Scikit -learn, Tensorflow, and Keras.

CHAPTER 3
PROPOSED SYSTEM

CHAPTER 3

PROPOSED SYSTEM

3.1 ALGORITHMS USED FOR PROPOSED MODEL

GRU(Gated Recurrent Unit): GRUs have fewer parameters compared to LSTMs, making them computationally more efficient. They are designed to capture long-term dependencies in sequential data while addressing the vanishing gradient problem.

Applications

• GRUs are commonly used in natural language processing (NLP) tasks such as machine translation, sentiment analysis, and text generation.

• They are also used in time series analysis for tasks like stock price prediction and weather forecasting.

LSTM (Long Short-Term Memory): LSTMs have memory cells and gating mechanisms (input gate, forget gate, output gate) that allow them to learn long-term dependencies in sequential data. They are robust against the vanishing gradient problem and can handle sequences of varying lengths.

Applications

• LSTMs are widely used in NLP for tasks like language modeling, named entity recognition, and speech recognition.

• They are also used in time series forecasting, anomaly detection, and generative modeling.

RNN (Recurrent Neural Network): RNNs maintain a hidden state that captures information from previous time steps, making them suitable for sequential data processing. They can model temporal dependencies and are capable of handling variable-length sequences.

Applications

• RNNs are used in sequence prediction tasks such as next word prediction in text, stock market prediction, and music generation.

• They are also applied in robotics for trajectory planning, in healthcare for disease prediction, and in video analysis for action recognition.

CNN (Convolutional Neural Network): CNNs use convolutional layers to extract spatial features from input data, making them effective for image analysis tasks. They often include pooling layers for down-sampling and feature reduction.

Applications

• CNNs are widely used in computer vision tasks such as object detection, image classification, and facial recognition.

• They are also applied in medical imaging for disease diagnosis, in autonomous vehicles for object detection, and in satellite imagery analysis.

Logistic Regression: Logistic Regression is a simple linear model that estimates probabilities for binary classification tasks. It uses a logistic function (sigmoid) to map input features to probabilities.

Applications

• Logistic Regression is used in various domains such as healthcare for disease diagnosis, marketing for customer segmentation, and finance for credit risk analysis.

SVM (Support Vector Machine): SVMs find the optimal hyperplane that separates different classes in the feature space, maximizing the margin between classes. They can handle both linearly separable and non-linearly separable data using kernel functions.

Applications

• SVMs are used in classification tasks like image classification, text categorization, and bioinformatics.

• They are also applied in regression tasks, anomaly detection, and feature selection.

GNB (Gaussian Naive Bayes): GNB assumes that features are independent and follow a Gaussian distribution, making it computationally efficient. Despite its simplicity, GNB can perform well in practice, especially with high-dimensional data.

Applications

• GNB is used in text classification, spam detection, sentiment analysis, and document categorization.

• It is also applied in healthcare for disease diagnosis, in finance for credit scoring, and in recommendation system.

Decision Tree: Decision Trees recursively split the input space based on feature values, creating a tree-like structure for decision making. They are easy to interpret and can capture non-linear relationships in the data.

Applications

• Decision Trees are used in classification tasks like customer churn prediction, fraud detection, and pattern recognition.

• They are also applied in regression tasks, feature selection, and ensemble learning techniques like Random Forests.

Classification of attacks

We address a assortment of assaults in our organize security endeavors. These incorporate Dissent of Benefit (DoS) assaults like apache2, back, arrive, neptune, mailbomb, unit, processtable, smurf, tear, udpstorm, and worm. Moreover, we center on Test assaults such as ipsweep, mscan, nmap, portsweep, holy person, and satan. Moreover, we consider Unauthorized Get to to Nearby Superuser (U2R) assaults like buffer_overflow, loadmodule, perl, ps, rootkit, sqlattack, and xterm. Finally, we address Unauthorized Get to from a Farther Machine (R2L) assaults like ftp_write, guess_passwd, http_tunnel, imap, multihop, named, phf, sendmail, snmpgetattack, snmpguess, spy, warezclient, warezmaster, xclock, and xsnoop. These categories offer assistance us classify and get it the nature of arrange interruptions, permitting us to create compelling defense instruments.

Dataset

The NSL-KDD dataset serves as an upgraded form of the KDD'99 dataset, advertising a profitable asset for analysts within the field of interruption location frameworks (IDS) and organize security. Its primary reason is to supply an compelling benchmark for comparing different interruption discovery strategies. Categorized into four primary classes—Denial of Benefit (DoS), Test, Unauthorized Get to from a Farther Machine (R2L), and Unauthorized Get to to Nearby Superuser Benefits (U2R)—the dataset includes a assorted extend of cyber assaults experienced in arrange situations. Analysts utilize the NSL-KDD dataset to create and assess interruption discovery methods, pointing to upgrade the discovery and moderation of security

dangers inside computer systems.

Data Analysis

Exploratory Information Investigation (EDA) is an fundamental step in understanding and analyzing a dataset comprehensively. It includes a few key assignments pointed at picking up bits of knowledge into the data's structure and characteristics. At first, labeling the column names is pivotal, because it allots significant identifiers to each include, encouraging less demanding elucidation and investigation. Taking after this, checking for invalid values guarantees that there are no lost sections within the dataset, which might something else skew investigation comes about or impede show execution. Along these lines, information visualization strategies such as making plots and charts are utilized to outwardly speak to the conveyance of information and investigate connections between diverse highlights. These visualizations help in recognizing designs, patterns, and anomalies within the dataset, subsequently illuminating ensuing steps within the information investigation prepare.

Feature Selection

Deciding the foremost relevant highlights may be a essential angle of information investigation, supporting in recognizing those traits that contribute most to anticipating the target variable or course name. In this setting, the recorded highlights are positioned based on their relationship with the target course. Highlights like 'dst_host_srv_count', 'logged_in', and 'dst_host_diff_srv_rate' display generally solid relationships with the target lesson, showing their potential centrality in recognizing between distinctive classes of organize activity. On the other hand, highlights such as 'num_shells' and 'urgent' appear weaker relationships with the target course, recommending they may have less prescient control or significance in this setting. Understanding the quality of these relationships guides the selection of highlights for building prescient models, guaranteeing that as it were the foremost enlightening traits are utilized, subsequently improving demonstrate exactness and productivity. Also, the nonattendance of relationship for 'num_outbound_cmds' with the target course highlights its negligible impact in separating between diverse classes of organize activity.

Algorithms

In our extend, we utilize a differing extend of calculations to address different angles of our issue. These calculations incorporate Choice Trees, Calculated Relapse, Bolster Vector Machines (SVM), Gaussian Naïve Bayes, as well as profound learning models such as Gated Repetitive Units (GRU), Long Short-Term Memory (LSTM) systems, Repetitive Neural Systems (RNN), and Convolutional Neural Systems (CNN). Each calculation offers interesting qualities and capabilities suited to diverse sorts of information and assignments inside our venture. By leveraging this combination of conventional machine learning and profound learning strategies, we point to viably address the complexities and challenges show in our issue space, eventually moving forward the precision and strength of our arrangements.

Implementation of block diagram

The flowchart starts with the introductory setup, where an application is opened and essential bundles are imported to encourage the advancement handle. Taking after this, the dataset is investigated and experiences information preprocessing, which regularly includes errands like cleaning the information, taking care of lost values, and changing the information into a appropriate organize for investigation. Another, the flowchart delineates a few key steps within the include designing handle, counting include era, include choice, and name encoding. These steps offer assistance in planning the information for encourage investigation and modeling. The flowchart at that point moves to the preparing and testing stages of the venture. Within the preparing stage, different machine learning calculations are connected, counting KFold cross-validation, calculated relapse, back vector machines (SVM), credulous Bayes, irregular timberlands, stacking classifiers, and voting classifiers. Also, profound learning procedures such as long short-term memory (LSTM), convolutional neural networks (CNN), gated repetitive units (GRU), and repetitive neural systems (RNN) are utilized amid the testing stage to assess the execution of the models. At long last, the flowchart concludes with the execution of user registration and login functionalities, allowing clients to supply input and get the ultimate yield or result from the online location. This organized approach laid out within the flowchart guarantees a precise and organized advancement handle, driving to the creation of a useful and user-friendly online stage.

3.2 DESIGNING

3.2.1 UML DIAGRAM

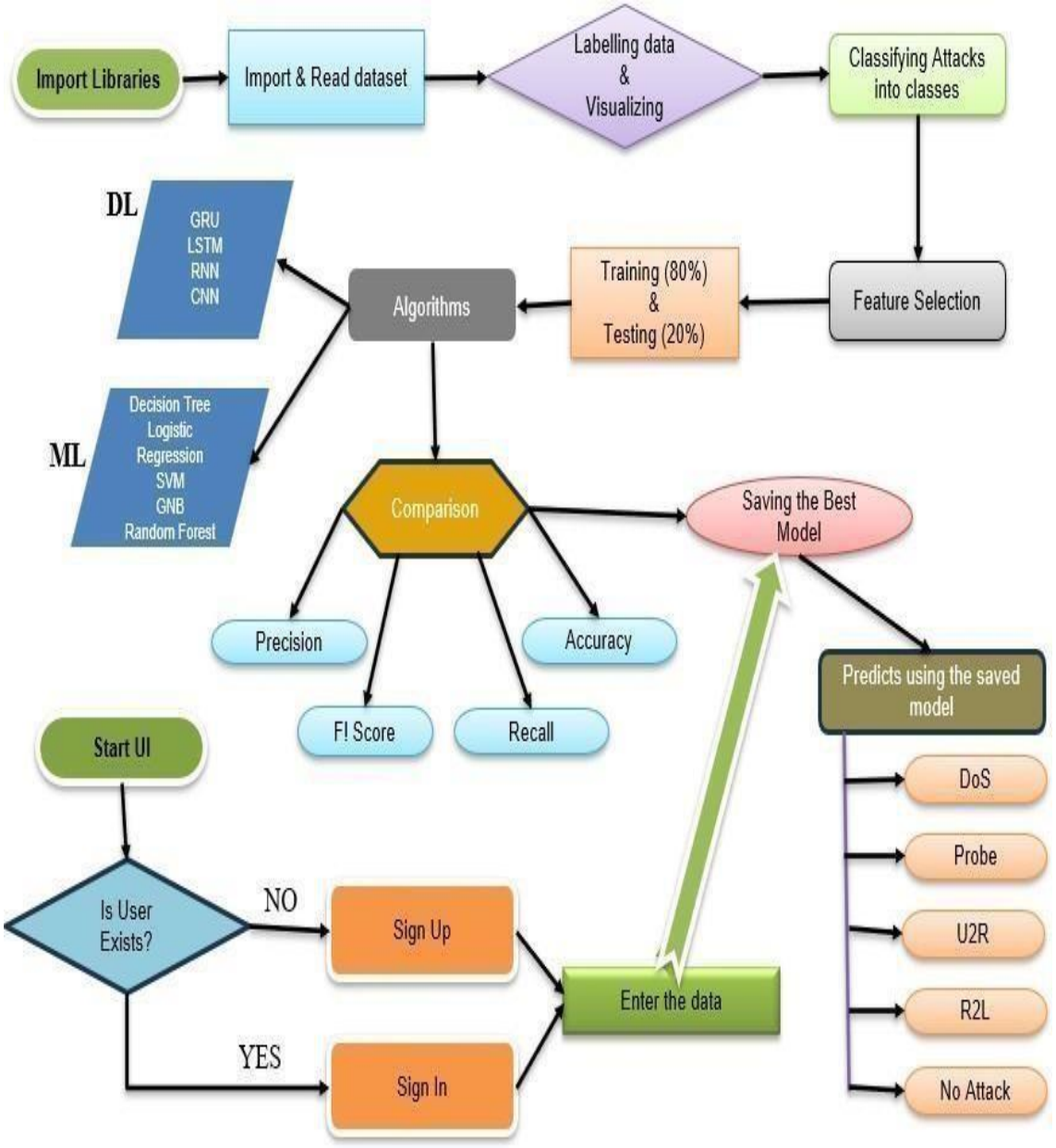


Figure 3.2.1 UML Diagram of propose system

3.3 STEPWISE IMPLEMENTATION OF CODE

Importing libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import cross_val_score
from sklearn import metrics
from sklearn.model_selection import train_test_split
import pickle
import time
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
import tensorflow as tf
from sklearn.preprocessing import LabelBinarizer
plt.style.use('fivethirtyeight')
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
from tensorflow.keras.layers import Dense, LSTM, GRU,
Conv1D
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
import warnings
warnings.filterwarnings('ignore')
```

Importing dataset

```
df_train_full = pd.read_csv("KDDTrain+.txt")
```

Labelling the dataset

```
NSL_KDD_columns = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
    'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
    'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell',
    'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
```

```
        'num_access_files', 'num_outbound_cmds', 'is_host_login',
        'is_guest_login', 'count', 'srv_count', 'serror_rate',
        'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_srv_rate',
        'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
        'dst_host_srv_count', 'dst_host_same_srv_rate',
        'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
        'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
        'dst_host_srv_serror_rate', 'dst_host_rerror_rate',
        'dst_host_srv_rerror_rate', 'class', 'classnum']
df_train_full.columns = NSL_KDD_columns
df_train_full.info()
df_train_full.nunique()
df_train_full.isnull().sum()
NSL_KDD_columns_categorical = ['protocol_type', 'service', 'flag', 'class']
NSL_KDD_columns_int = ['duration',  'src_bytes',
        'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
        'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell',
        'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
        'num_access_files', 'num_outbound_cmds', 'is_host_login',
        'is_guest_login', 'count', 'srv_count', 'dst_host_count',
        'dst_host_srv_count', 'classnum']
         'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
         'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
         'dst_host_srv_serror_rate', 'dst_host_rerror_rate',
         'dst_host_srv_rerror_rate']
```

Data visualization

```
# categorical feature 'protocol_type'
temp = df_train_full['protocol_type'].value_counts()
a = temp.index
b = temp.values
zipped = list(zip(a, b))
d = pd.DataFrame(zipped, columns=['protocol_type', 'occurances'])
sns.barplot(data=d, x='protocol_type', y='occurances')
# categorical feature 'service'
temp = df_train_full['service'].value_counts()
a = temp.index
b = temp.values
zipped = list(zip(a, b))
d = pd.DataFrame(zipped, columns=['service', 'occurances'])
fig, ax = plt.subplots(figsize=(15, 6))
sns.barplot(data=d, x='service', y='occurances')
```

```
fig.tight_layout()
plt.xticks(rotation=90)
plt.show()
#ax.set_xlabel()
#ax.set_ylabel()
# categorical feature 'flag'
temp = df_train_full['flag'].value_counts()
a = temp.index
b = temp.values
zipped = list(zip(a, b))
d = pd.DataFrame(zipped, columns=['flag', 'occurances'])
fig, ax = plt.subplots()
sns.barplot(data=d, x='flag', y='occurances')
fig.tight_layout()
plt.xticks(rotation=90)
plt.show()
# categorical feature 'class'
temp = df_train_full['class'].value_counts()
a = temp.index
b = temp.values
zipped = list(zip(a, b))
d = pd.DataFrame(zipped, columns=['class', 'occurances'])
fig, ax = plt.subplots()
sns.barplot(data=d, x='class', y='occurances')
fig.tight_layout()
plt.xticks(rotation=90)
plt.show()
print(df_train_full[NSL_KDD_columns_int].nunique())
# int type features - candidate for OneHotEncoding
list1 = ['land', 'wrong_fragment', 'urgent', 'num_failed_logins',
        'logged_in', 'root_shell', 'su_attempted', 'num_shells',
        'num_outbound_cmds', 'is_host_login', 'is_guest_login']
# int type features having highly diverse values
list2 = ['duration', 'src_bytes','dst_bytes', 'hot',
        'num_compromised', 'num_root', 'num_file_creations',
        'num_access_files', 'count', 'srv_count', 'dst_host_count',
        'dst_host_srv_count', 'classnum']
# Those int type features which are candidate for OneHotEncoding
for i in list1:
    print('feature name:',i,'\n', df_train_full[i].value_counts(ascending=False), '\n\n\n')
# highly skewed values - candidate for LogTransformation
# except 'dst_host_srv_count' all features are skewed. This feature is bi-modal.
temp = df_train_full[list2]
```

```
fig, ax = plt.subplots(13, 1, figsize=(4, 60))
fig.tight_layout()
for i, j in enumerate (list2):
   sns.kdeplot(ax=ax[i], data=temp, x=j)
plt.show()
# multi-modal values (dominantly bi-mondal)
fig, ax = plt.subplots(15, 1, figsize=(4, 50))
fig.tight_layout()
for i, j in enumerate (NSL_KDD_columns_float):
   sns.kdeplot(ax=ax[i], data=df_train_full, x=j)
plt.show()
```

Attack Classes

```
DoS_attacks
['apache2','back','land','neptune','mailbomb','pod','processtable','smurf',Probe_attacks =
['ipsweep','mscan','nmap','portsweep','saint','satan']
U2R = ['buffer_overflow','loadmodule','perl','ps','rootkit','sqlattack','xterm']
R2L
['ftp_write','guess_passwd','http_tunnel','imap','multihop','named',
s','spy','warezclient','warezmaster','xclock','xsnoop']
#attack_labels = ['Normal','DoS','Probe','U2R','R2L']
df_train_full['class'].replace(['apache2','back','land','neptune','mailbomb','pod',
                        'processtable','smurf','teardrop','udpstorm','worm'],
                        'DoS_attack', inplace=True)
df_train_full['class'].replace(['ipsweep','mscan','nmap','portsweep','saint','satan'],
                        'Probe_attacks', inplace=True)
df_train_full['class'].replace(['buffer_overflow','loadmodule','perl','ps','rootkit',
                        'sqlattack','xterm'], 'U2R', inplace=True)
df_train_full['class'].replace(['ftp_write','guess_passwd','http_tunnel','imap',
                        'multihop','named','phf','sendmail','snmpgetattack',
                        'snmpguess','spy','warezclient','warezmaster','xclock',
                        'xsnoop'], 'R2L', inplace=True)
df_train_full['class'].replace('normal', 'Normal', inplace=True)
df_train_full['class'].value_counts()
# 'class' feature
temp = df_train_full['class'].value_counts()
a = temp.index
b = temp.values
zipped = list(zip(a, b))
d = pd.DataFrame(zipped, columns=['class', 'occurances'])
fig.tight_layout()
fig, ax = plt.subplots()
```

```
plt.pie(data=d, x=b)
labels = ['Normal','DoS','Probe','U2R','R2L']
plt.legend(labels, bbox_to_anchor=(1, 0, 0.5, 1))
plt.show()
#obtain the correlations of each features in dataset
corr_matrix = df_train_full.corr()
top_corr_features = corr_matrix.index
plt.figure(figsize=(20,20))
#plot heat map
h_map=sns.heatmap(df_train_full[top_corr_features].corr(), cmap="Blues")
```

Model Building

```
df_train_full.info()
df_train_full['protocol_type'].unique()
df_train_full['service'].unique()
df_train_full['flag'].unique()
df_train_full['class'].unique()
protocol_type_mapping = {'udp': 0, 'tcp': 1, 'icmp': 2}
df_train_full['protocol_type'] = df_train_full['protocol_type'].map(protocol_type_mapping)
service_mapping df_train_full.info()
df_train_full['protocol_type'].unique()
df_train_full['service'].unique()
df_train_full['flag'].unique()
df_train_full['class'].unique()
protocol_type_mapping = {'udp': 0, 'tcp': 1, 'icmp': 2}
df_train_full['protocol_type'] = df_train_full['protocol_type'].map(protocol_type_mapping)
service_mapping
    'other': 0, 'private': 1, 'http': 2, 'remote_job': 3, 'ftp_data': 4,
    'name': 5, 'netbios_ns': 6, 'eco_i': 7, 'mtp': 8, 'telnet': 9, 'finger': 10,
    'domain_u': 11, 'supdup': 12, 'uucp_path': 13, 'Z39_50': 14, 'smtp': 15,
    'csnet_ns': 16, 'uucp': 17, 'netbios_dgm': 18, 'urp_i': 19, 'auth': 20,
    'domain': 21, 'ftp': 22, 'bgp': 23, 'ldap': 24, 'ecr_i': 25, 'gopher': 26,
    'vmnet': 27, 'systat': 28, 'http_443': 29, 'efs': 30, 'whois': 31, 'imap4': 32,
    'iso_tsap': 33, 'echo': 34, 'klogin': 35, 'link': 36, 'sunrpc': 37, 'login': 38,
    'kshell': 39, 'sql_net': 40, 'time': 41, 'hostnames': 42, 'exec': 43, 'ntp_u': 44,
    'discard': 45, 'nntp': 46, 'courier': 47, 'ctf': 48, 'ssh': 49, 'daytime': 50,
    'shell': 51, 'netstat': 52, 'pop_3': 53, 'nnsp': 54, 'IRC': 55, 'pop_2': 56,
    'printer': 57, 'tim_i': 58, 'pm_dump': 59, 'red_i': 60, 'netbios_ssn': 61,

    'rje': 62, 'X11': 63, 'urh_i': 64, 'http_8001': 65, 'aol': 66, 'http_2784': 67,
    'tftp_u': 68, 'harvest': 69
}
```

```
df_train_full['service'] = df_train_full['service'].map(service_mapping)
flag_mapping = {
    'SF': 0, 'S0': 1, 'REJ': 2, 'RSTR': 3, 'SH': 4, 'RSTO': 5, 'S1': 6, 'RSTOS0': 7,
    'S3': 8, 'S2': 9, 'OTH': 10
}
df_train_full['flag'] = df_train_full['flag'].map(flag_mapping)
class_mapping = {
    'Normal': 0,
    'DoS_attack': 1,
    'R2L': 2,
    'Probe_attacks': 3,
    'U2R': 4
}
df_train_full['class'] = df_train_full['class'].map(class_mapping)
df_train_full.info()
```

Feature Selection

```
df_train_full = df_train_full.drop(['classnum'], axis =1)
correlation = df_train_full.corr()['class'].abs().sort_values(ascending=False)
print(correlation)
X = df_train_full[
    ['dst_host_srv_count', 'logged_in', 'dst_host_diff_srv_rate', 'dst_host_same_srv_rate',
     'dst_host_same_src_port_rate', 'flag', 'same_srv_rate', 'dst_host_srv_rerror_rate',
     'srv_rerror_rate', 'rerror_rate', 'diff_srv_rate', 'dst_host_srv_diff_host_rate',
     'dst_host_rerror_rate', 'count', 'protocol_type', 'dst_host_srv_serror_rate',
     'serror_rate', 'dst_host_serror_rate', 'srv_serror_rate']
]
y = df_train_full['class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
def evaluate(model, X_train, X_test, y_train, y_test):

    y_test_pred = model.predict(X_test)
    y_train_pred = model.predict(X_train)
    print("TRAINIG RESULTS: \n==============================")
    clf_report = pd.DataFrame(classification_report(y_train, y_train_pred, output_dict=True))
    print(f"CONFUSION MATRIX:\n{confusion_matrix(y_train, y_train_pred)}")
    print(f"ACCURACY SCORE:\n{accuracy_score(y_train, y_train_pred):.4f}")
    print(f"CLASSIFICATION REPORT:\n{clf_report}")
    print("TESTING RESULTS: \n==============================")
    clf_report = pd.DataFrame(classification_report(y_test, y_test_pred, output_dict=True))
    print(f"CONFUSION MATRIX:\n{confusion_matrix(y_test, y_test_pred)}")
    print(f"ACCURACY SCORE:\n{accuracy_score(y_test, y_test_pred):.4f}")
```

```
    print(f"CLASSIFICATION REPORT:\n{clf_report}")
```

Decision Tree

```
from sklearn import tree
dt = tree.DecisionTreeClassifier(
    splitter='best',
    min_samples_split=2,
    criterion='entropy',
    max_depth=5,
    random_state=500
)
start_time = time.time()
dt.fit(X_train, y_train)
end_time = time.time()
dt_time = end_time - start_time
evaluate(dt, X_train, X_test, y_train, y_test)
y_pred = dt.predict(X_test)
a = accuracy_score(y_test, y_pred) * 100
p = precision_score(y_test,y_pred,average='macro') * 100
r = recall_score(y_test,y_pred,average='macro') * 100
f = f1_score(y_test,y_pred,average='macro') * 100
conf_matrix = confusion_matrix(y_test, y_pred)

TN = conf_matrix[0, 0]
FP = conf_matrix[0, 1]
FN = conf_matrix[1, 0]
TP = conf_matrix[1, 1]
sensitivity = TP / (TP + FN)* 100
specificity = TN / (TN + FP)* 100
scores = cross_val_score(dt, X_train, y_train, cv=10)
print("Average accuracy:", scores.mean())
  plt.xlim([-0.1,1.2])
  plt.ylim([-0.1,1.2])
  plt.ylabel('True Positive Rate')
  plt.xlabel('False Positive Rate')
  plt.show()
def plotGraph():
  global graph
  height = graph
  bars = ('ML Bot Accuracy','ML URL Accuracy','VGG19 BOT & URL Accuracy')
```

```
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars)
    plt.xlabel("Algorithm Names")
    plt.ylabel("Accuracy")
    plt.title("Accuracy Comparison Graph")
    plt.show()
font = ('times', 16, 'bold')
title = Label(main, text='Detecting Malicious Twitter Bots Using Machine Learning')
title.config(bg='goldenrod2', fg='black')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)
font1 = ('times', 12, 'bold')
text=Text(main,height=20,width=150)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=50,y=120)
text.config(font=font1)
font1 = ('times', 13, 'bold')
uploadButton = Button(main, text="Upload Tweets Dataset", command=uploadDataset,
bg='#ffb3fe')
        tweets = details[i,14]
        if 'http' in str(tweets):
            urls.append(1)
        else:
            urls.append(0)
    train_attr = dataset[
        ['followers_count', 'friends_count', 'listedcount', 'favourites_count', 'statuses_count',
'verified']]
    train_attr["URLS"] = urls #adding URLS to training dataset
    text.insert(END,str(train_attr))
    train_label = dataset[['bot']]
```

```
    X1 = train_attr
    Y1 = np.asarray(train_label)
    X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, Y1, test_size=0.2)
    logreg = LogisticRegression().fit(X_train1, y_train1) #logistic regression object
    actual = y_test1
    pred = logreg.predict(X_test1)
    accuracy = accuracy_score(actual, pred) * 100
    graph.append(accuracy)
    precision = precision_score(actual, pred) * 100
    recall = recall_score(actual, pred) * 100
    f1 = f1_score(actual, pred)
    auc = roc_auc_score(actual, pred)
    text.insert(END,'\nLogistic Regression Accuracy : '+str(accuracy)+"\n")
    text.insert(END,'Logistic Regression Precision : '+str(precision)+"\n")
    text.insert(END,'Logistic Regression Recall is : '+str(recall)+"\n")
    text.insert(END,'Logistic Regression Area Under Curve is : '+str(auc))
    fpr, tpr, thresholds = metrics.roc_curve(actual, pred)
    auc = metrics.auc(fpr, tpr)
    plt.title('ROC')
    plt.plot(fpr, tpr, 'b',
        vgg_model.add(Convolution2D(32, (1, 1), activation = 'relu'))
        vgg_model.add(MaxPooling2D(pool_size = (1, 1)))
        vgg_model.add(Flatten())
        vgg_model.add(Dense(units = 256, activation = 'relu'))
        vgg_model.add(Dense(units = y_train.shape[1], activation = 'softmax'))
        vgg_model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics =
['accuracy'])
        model_check_point = ModelCheckpoint(filepath='model/vgg19_weights.hdf5', verbose = 1,
save_best_only = True)
        hist = vgg_model.fit(X_train, y_train, batch_size=16, epochs=40, shuffle=True,
callbacks=[model_check_point], verbose=1, validation_data=(X_test, y_test))
    predict = vgg_model.predict(X_test) #now perform prediction on test data
    predict = np.argmax(predict, axis=1)
```

```
y_test1 = np.argmax(y_test, axis=1)
accuracy = accuracy_score(y_test1, predict) * 100
graph.append(accuracy)
precision = precision_score(y_test1, predict) * 100
recall = recall_score(y_test1, predict) * 100
f1 = f1_score(y_test1, predict)
auc = roc_auc_score(y_test1, predict)
text.insert(END,'\nVGG19 Accuracy : '+str(accuracy)+"\n")
text.insert(END,'VGG19 Precision : '+str(precision)+"\n")
text.insert(END,'VGG19 Recall is : '+str(recall)+"\n")
text.insert(END,'VGG19 Area Under Curve is : '+str(auc))
fpr, tpr, thresholds = metrics.roc_curve(y_test1,  predict)
auc = metrics.auc(fpr, tpr)
plt.title('ROC')
plt.plot(fpr, tpr, 'b',
label='AUC = %0.2f'% auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
text.insert(END,"Possible BOT users\n\n")
users = []
for i in range(len(details)):
    screen = details[i,0]
    status = details[i,1]
    name = details[i,2]
    followers = int(details[i,3])
    friends =  int(details[i,4])
    listed = int(details[i,5])
    favourite = int(details[i,6])
    status_count = int(details[i,7])
    verified = details[i,8]
    if not verified: #check user not verified
        bow = defaultdict(int) #bag of words
        data = str(screen)+" "+str(name)+" "+str(status)#checking screen name, tweets and
```

```
        data = data.lower().strip("\n").strip()
        data = re.findall(r'\w+', data)
        for j in range(len(data)):
            bow[data[j]] += 1 #adding each word frequency to bag of words
        frequency = getFrequency(bow) #getting frequency of BOTS words
        if frequency > 0 and listed < 16000 and followers < 200: #if condition true then its bots
            users.append(screen)
    text.insert(END,str(users)+"\n")
    train_attr = dataset[
        ['followers_count', 'friends_count', 'listedcount', 'favourites_count', 'statuses_count',
'verified']]
    train_label = dataset[['bot']]
    X = train_attr
    Y = train_label.as_matrix()
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
    print(X.shape)
```

App.py

```
from flask import Flask,request, url_for, redirect, render_template
import pandas as pd
import numpy as np
import pickle
import sqlite3
import keras
app = Flask(_name_)
import tensorflow as tf
loaded_model = tf.keras.models.load_model('models/lstm.h5')
@app.route('/')
def hello_world():
    return render_template("home.html")
@app.route('/logon')
def logon():
        return render_template('signup.html')
@app.route('/login')
def login():for i in range(1,20):
        inp.append(float(request.form[str(i)]))
```

```
row_df_reshaped = row_df.reshape(1, 1, 19)
    predictions = loaded_model.predict(row_df_reshaped)
    predicted_labels = np.argmax(predictions, axis=1)
    prediction = predicted_labels[0]
    if prediction == 0:
        return render_template('after.html',pred=f'Normal')
    elif prediction == 1:
        return render_template('after.html',pred=f'DoS Attack')
    elif prediction == 2:
        return render_template('after.html',pred=f'R2L Attack')
    elif prediction == 3:
        return render_template('after.html',pred=f'Probe Attack')
    elif prediction == 4:
        return render_template('after.html',pred=f'U2R Attack')
@app.route('/index')
```

CHAPTER 4
RESULTS AND DISCUSSION

CHAPTER 4

RESULTS AND DISCUSSION

4.1 PERFORMANCE VALIDATION OF MACHINE LEARNING ALGORITHMS

Precision Definition: It is the measure of ratio of correctly predicted positive observations to the total predicted positives

Table 4.1.1: Precision

Algorithms	Normal	DoS Attack	R2L Attack	Probe Attack	U2R Attack
Decision Tree	0.967720	0.972861	0.744722	0.826865	0.0
Logistic Regression	0.888980	0.944034	0.0	0.857955	0.0
SVM	0.169973	0.716787	0.057082	0.169648	0.0
GNB	0.946300	0.974133	0.043536	0.341962	0.002609

Recall Definition: It is the measure of ratio of correctly predicted positive observations to all actual positives.

Table 4.1.2: Recall

Algorithms	Normal	DoS Attack	R2L Attack	Probe Attack	U2R Attack
Decision Tree	0.954000	0.968885	0.487437	0.936335	0.0
Logistic Regression	0.980076	0.939841	0.0	0.439190	0.0
SVM	0.087330	0.097022	0.440955	0.611871	0.0
GNB	0.678349	0.746969	0.204774	0.576322	0.974359

F1 score Definition: It is the harmonic mean of precision and recall. It provides a balanced measure between precision and recall.

Table 4.1.3: F1 score

Algorithms	Normal	DoS Attack	R2L Attack	Probe Attack	U2R Attack
Decision Tree	0.960811	0.970869	0.589218	0.878202	0.0
Logistic Regression	0.932308	0.941933	0.0	0.580976	0.0
SVM	0.115380	0.170910	0.101080	0.265644	0.0
GNB	0.790228	0.845560	0.071806	0.429236	0.005204

Support Definition: It indicated the distribution of instances across different classes.

Table 4.1.4: Support

Algorithms	Normal	DoS Attack	R2L Attack	Probe Attack	U2R Attack
Decision Tree	53956.000000	36703.000000	796.000000	9283.000000	39.0
Logistic Regression	53956.000000	36703.000000	796.0	9283.000000	39.0
SVM	53956.000000	36703.000000	796.000000	9283.000000	39.0
GNB	53956.000000	36703.000000	796.000000	9283.000000	39.000000

4.2 PERFORMANCE VALIDATION OF DEEP LEARNING ALGORITHMS

- Average accuracy is calculated as the mean of the accuracy of each fold.

Table 4.2.1: Average Accuracy across k-Folds (k=10)

Algorithm	Average Accuracy across k-folds (K=10)
GRU	0.9850327432155609
LSTM	0.9848382592201232
RNN	0.9822266280651093
CNN	0.9752728700637817

4.3 COMPARISION OF ALGORITHMS

Table 4.3.1 interprets the data of the comparison of the accuracy, precision, f1-score, recall, sensitivity and specificity of Decision Tree, Logistic Regression, SVM, Gaussian Naïve Bayes, GRU, LSTM, RNN and CNN algorithms. We can have a clear view that in terms of accuracy, f1-score and specificity LSTM has the high performance and in precision RNN, in recall and sensitivity GRU.

Table 4.3.1: Comparison Of Algorithms

Algorithm	Accuracy	Precision	F1-Score	Recall	Sensitivity	Specificity
Decision Tree	95.336376	70.186724	67.715002	66.592677	97.558317	98.265495
Logistic Regression	90.208375	53.396384	48.559363	46.693533	93.774488	99.010873
SVM	14.054376	21.989918	12.874339	24.240124	15.304756	75.351641
Gaussian Naïve Bayes	68.779520	46.208900	42.779500	62.191868	94.777563	98.068995
GRU	98.654495	72.935717	74.018918	75.251725	99.337605	99.660557
LSTM	98.765628	74.384986	74.783166	75.194744	99.207469	99.856809
RNN	98.543362	76.799996	72.906212	70.491976	99.197049	99.706723
CNN	97.983727	75.900331	71.622521	68.963413	99.043270	99.586435

4.4 GRAPHICAL REPRESENTATION

Figure 4.4.1 shows the graphical representation of the accuracy, precision, f1-score, recall, sensitivity and specificity of the algorithms.

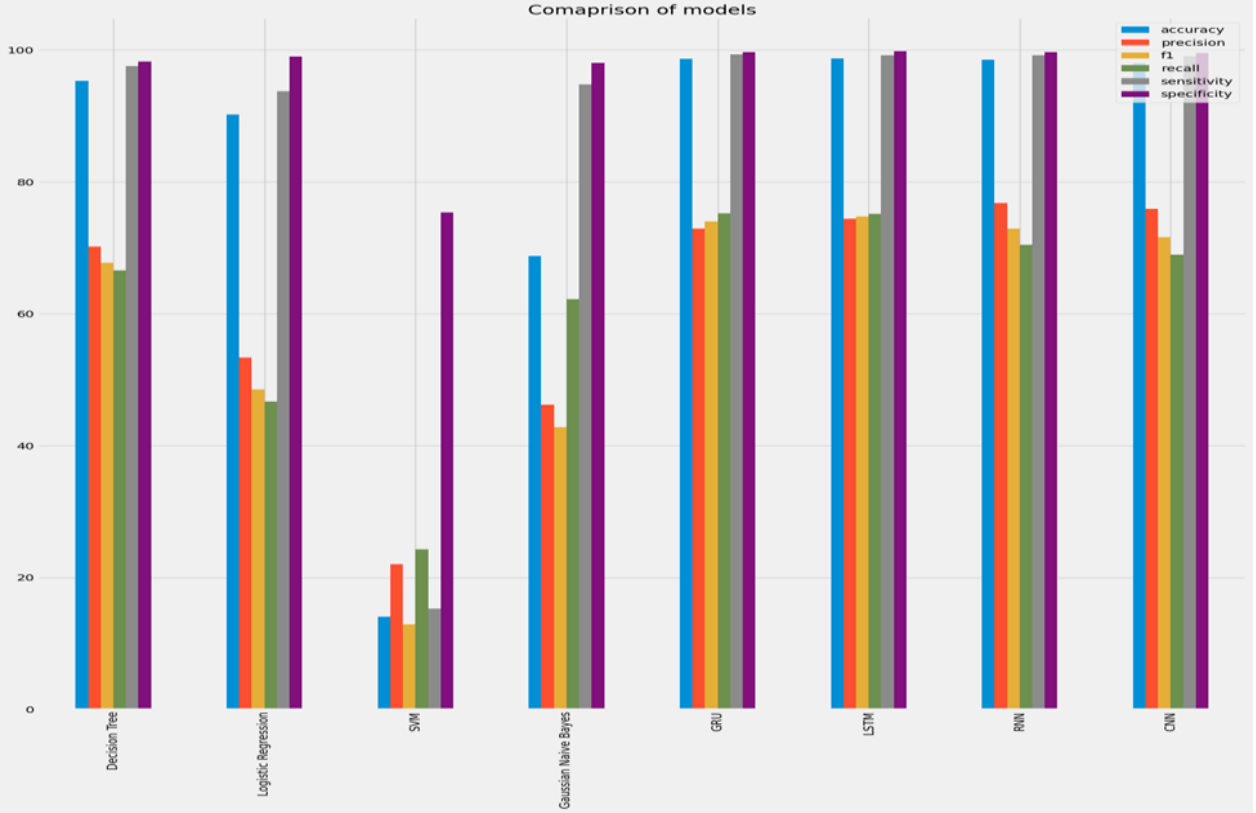


Figure 4.4.1: Graphical representation of Comparison of models

Figure 4.4.2 shows the graphical representation of the comparison of cross validation accuracy of algorithms.

Cross Validation Accuracy: It is a robust technique used as a performance metric to compare the efficiency of different models.

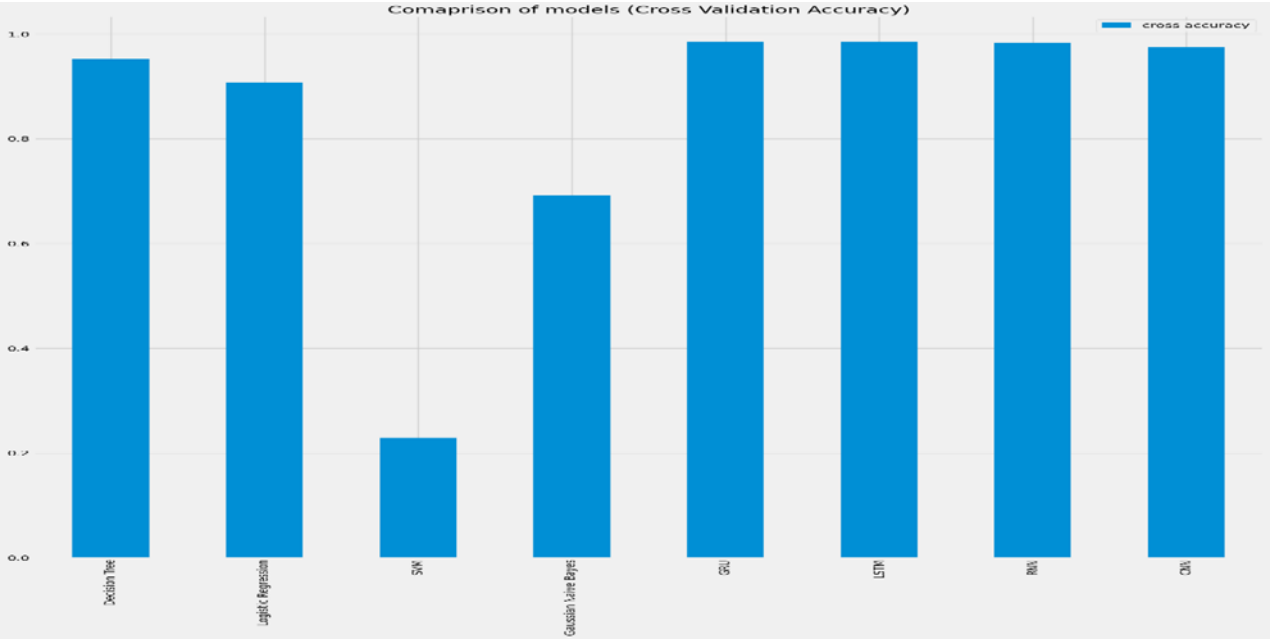


Figure 4.4.2: Graphical representation of Cross Validation Accuracy

Figure 4.4.3 shows the execution time of different models to train and test the dataset. We can generalize that the Deep Learning models takes longer time to train than the ML algorithms. LSTM and GAN take the maximum time to train the model.

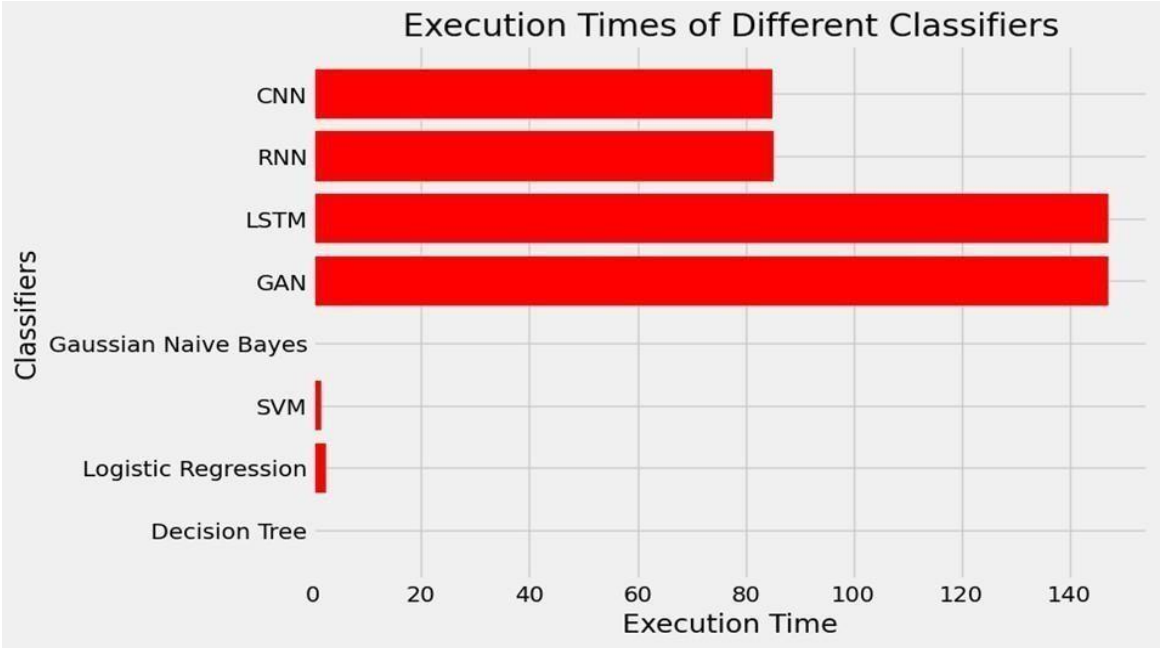


Figure 4.4.3: Graphical Representation of Execution Times of algorithms

4.5 USER INTERFACE

Figure 4.5.1 shows the first step of user interface where user will come across home page which contains login and sign up features and user need to select signup or login to move forward.

Figure 4.5.1: Home page

Figure 4.5.2 shows the second step of user interface where user will come across signing up and logging in. If user is new to website then they have to signup using signup feature it will ask user to set password and save that password. If user had already registered to website then user need to enter their credentials that is they need to enter their login id and password in order to enter to next page and here there is an additional for setting password if user had forgotten their password, in that case they need to click forgot password and can set their new password.

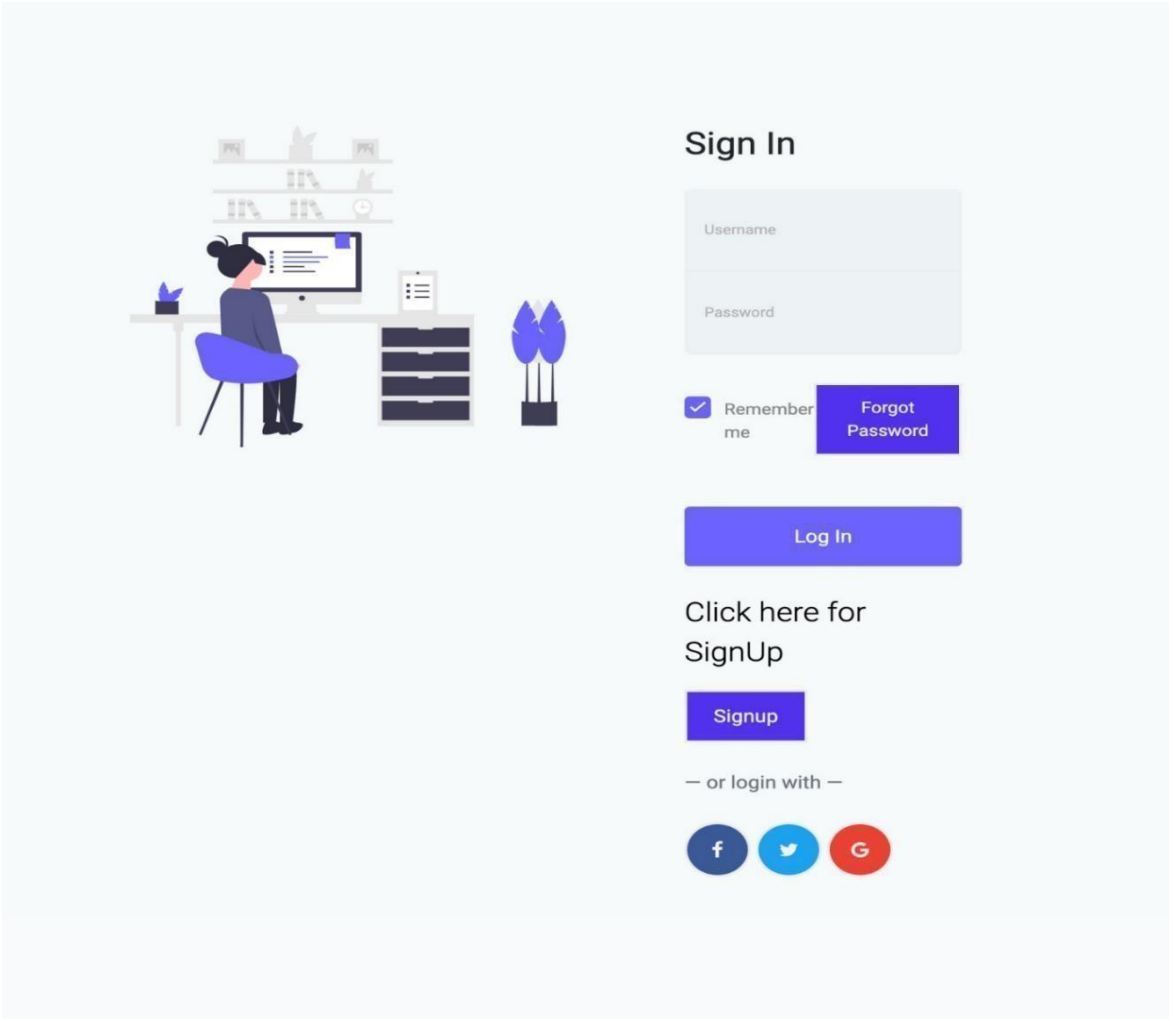


Figure 4.5.2: Signup/Login page

Figure 4.5.3 shows the third step of user interface where user need to enter 19 features where these features will act as inputs for detecting what type of attack is been taken place.

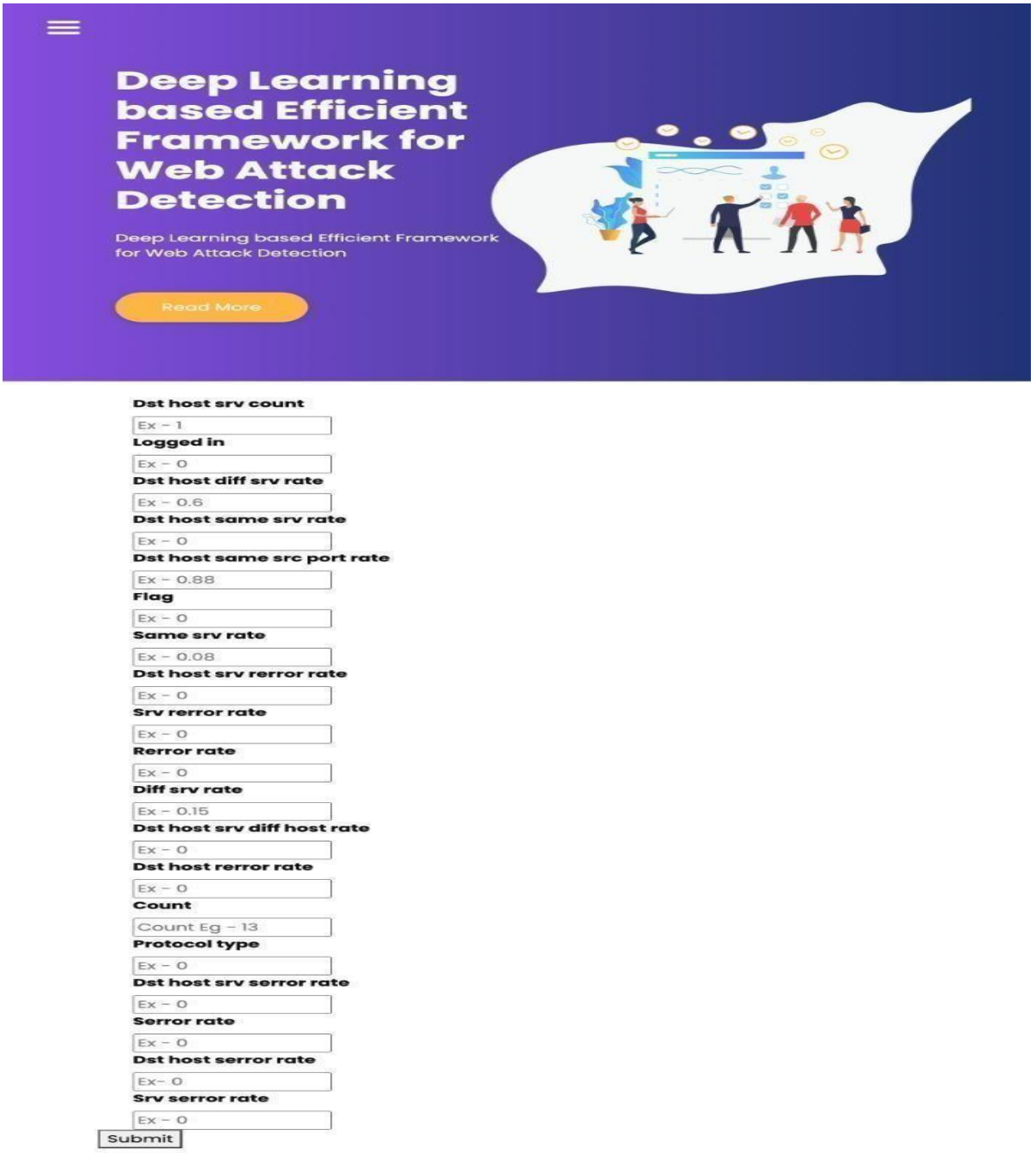


Figure 4.5.3: Inputting features

Figure 4.5.4 will be the final step of user interface where user will get their desired output where this result in between 5 attacks i.e normal, R2L, U2R, DoS, Probe attack.

Figure 4.5.4: Final Output

CHAPTER 5
CONCLUSION

5. CONCLUSION

The ponder presents a novel approach for arrange assault discovery, leveraging profound learning methods such as LSTM (Long Short-Term Memory) nearby conventional machine learning models. Through tests conducted on the NSL-KDD dataset, both LSTM and different machine learning calculations were utilized for execution assessment. The discoveries of this investigate not as it were illustrate the adequacy of the LSTM demonstrate but too highlight its predominance over existing state-of-the-art approaches, counting machine learning calculations. This approval through execution comparison underscores the potential of LSTM as a strong arrangement for organize assault discovery in real-world scenarios. In future endeavors, the center will be on optimizing the computational productivity of the LSTM demonstrate. This involves refining its design to diminish computational costs without compromising discovery precision. Furthermore, the proposed demonstrate will experience encourage preparing on assorted sorts of assaults to guarantee its adequacy in tending to modern and advancing dangers. In outline, the consider presents a promising progression in arrange assault location, advertising a profound learning approach with LSTM that outperforms conventional machine learning strategies in terms of execution. The commitment to future enhancements, counting computational optimization and improved versatility to rising dangers, implies a proactive position in progressing cybersecurity measures for arrange defense.

REFERENCES

REFERENCES

[1] Halfond, W. G., Viegas, J., & Orso, A. (2006, March). A classification of SQL-injection attacks and countermeasures. In Proceedings of the IEEE international symposium on secure software engineering (Vol. 1, pp. 13-15). IEEE.

[2] Wassermann, G., & Su, Z. (2008, May). Static detection of cross-site scripting vulnerabilities. In Proceedings of the 30th international conference on Software engineering (pp. 171-180).

[3] Di Pietro, R., & Mancini, L. V. (Eds.). (2008). Intrusion detection systems (Vol. 38). Springer Science & Business Media.

[4] Qie, X., Pang, R., & Peterson, L. (2002). Defensive programming: Using an annotation toolkit to build DoS-resistant software. ACM SIGOPS Operating Systems Review, 36(SI), 45-60.

[5] Ben-Asher, N., & Gonzalez, C. (2015). Effects of cyber security knowledge on attack detection. Computers in Human Behavior, 48, 51-61.

[6] Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. Intelligent data analysis, 6(5), 429-449.

[7] Liu, G., Yi, Z., & Yang, S. (2007). A hierarchical intrusion detection model based on the PCA neural networks. Neurocomputing, 70(7-9), 1561-1568.

[8] Xu, X., & Wang, X. (2005, July). An adaptive network intrusion detection method based on PCA and support vector machines. In International conference on advanceddata mining and applications (pp. 696-703). Berlin, Heidelberg: Springer Berlin Heidelberg.

[9] Pietraszek, T. (2004). Using adaptive alert classification to reduce false positives in intrusion detection. In Recent Advances in Intrusion Detection: 7th International Symposium, RAID 2004, Sophia Antipolis, France, September 15-17, 2004. Proceedings 7 (pp. 102-124).

[10] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84-90.

[11] Sun, F., Zhang, P., White, J., Schmidt, D., Staples, J., & Krause, L. (2017, June). A feasibility study of autonomically detecting in-process cyber-attacks. In 2017 3rd IEEE International Conference on Cybernetics (CYBCONF) (pp. 1-8). IEEE.

[12] Fu, X., Lu, X., Peltsverger, B., Chen, S., Qian, K., & Tao, L. (2007, July). A static analysis

framework for detecting SQL injection vulnerabilities. In 31st annual international computer software and applications conference (COMPSAC 2007) (Vol. 1, pp. 87-96). IEEE.

[13] Pan, Y., Sun, F., Teng, Z., White, J., Schmidt, D. C., Staples, J., & Krause, L. (2019). Detecting web attacks with end-to-end deep learning. Journal of Internet Services and Applications, 10(1), 1-22.

[14] Tian, Z., Luo, C., Qiu, J., Du, X., & Guizani, M. (2019). A distributed deep learning system for web attack detection on edge devices. IEEE Transactions on Industrial Informatics, 16(3), 1963-1971.

[15] Dutta, V., Choraś, M., Pawlicki, M., & Kozik, R. (2020). A deep learning ensemble for network anomaly and cyber-attack detection. Sensors, 20(16), 4583.

[16] Jayaswal, B. K., & Patton, P. C. (2006). Design for trustworthy software: Tools, techniques, and methodology of developing robust software. Pearson Education.

[17] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neuralcomputation, 9(8), 1735-1780.

[18] Cummins, C., Petoumenos, P., Wang, Z., & Leather, H. (2017, September). End-to-end deep learning of optimization heuristics. In 2017 26th International Conference on Parallel Architectures and Compilation Techniques (PACT) (pp. 219-232). IEEE.

[19] Kendall, K. K. R. (1999). A database of computer attacks for the evaluation of intrusion detection systems (Doctoral dissertation, Massachusetts Institute of Technology).

[20] Ng, A. (2011). Sparse autoencoder. CS294A Lecture notes, 72(2011), 1-19.

[21] Zargar, S. (2021). Introduction to sequence learning models: RNN, LSTM, GRU. Department of Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, North Carolina, 27606.

[22] Liu, J., Kantarci, B., & Adams, C. (2020, July). Machine learning-driven intrusion detection for Contiki-NG-based IoT networks exposed to NSL-KDD dataset. In Proceedings of the 2nd ACM workshop on wireless security and machine learning (pp. 25-30).

[23] Bisong, E. (2019). Building machine learning and deep learning models on Google cloud Platform(pp. 59-64). Berkeley, CA: Apress.

GITHUB LINK:

https://github.com/RISHITA-45/Application-of-Robust-Software-Modelling-Tool-for-Web-Attacks-Detection

iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** III **Month of publication:** March 2024

DOI: https://doi.org/10.22214/ijraset.2024.59078

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue III Mar 2024- Available at www.ijraset.com

Application of Robust Software Modelling Toolfor Web Attacks Detection

A. Mounika Rajeswari[1], Kalluri Rishita[2], Lanka Shriya[3], Balla Ganesh[4]
[1]Assistance Professor, [2, 3, 4]UG Student, Department of Computer Science & Engineering, CMR College of Engineering & Technology, Hyderabad, India

Abstract: ***The current inquiry as an examination of web attacks where proliferation of web-based applications has brought about a concurrent rise in cyber threats, particularly the form of web attacks targeting vulnerable systems. Approaches to web attack detection often rely on rule-based or signature-based methods, which struggle to change with the increasing landscape of attacks. In response, this study proposes an innovative approach leveraging DL techniques for web attacks. By harnessing the capability of DL, especially CNN and recurrent neural networks (RNNs), our proposed system learns directly from raw web traffic data, eliminating the need for manual feature engineering. This end-to-end approach not only streamlines the detection process but also enhances the system's ability to generalize across different types of attacks and adapt to new threats. To evaluatethe effectiveness of our approach, we conducted extensive experiments on diverse datasets containing both benign and maliciousweb traffic. Our results demonstrate the superiority of end-to-end deep learning over traditional methods, achieving higher detection accuracy and robustness against adversarial attacks. In conclusion, our study highlights the promise of end-to-end deep learning as a viable approach related to web attacks, offering enhanced detection capabilities in the phase of evolving cyberthreats.***

Keywords: Deep Learning, Web Attack Detection

I. INTRODUCTION

A web assault alludes to any malevolent movement or activity carried out with the deliberate of compromising the security, astuteness, or availabilities of websites, web applications, web servers, or there clients. Theses assaults misuse vulnerabilities or shortcomings in web advances, conventions, or arrangements to attain different evil goals. Web assaults can show in various shapes and can target distinctive layers of the internet stack, counting the application layer, organize layer, and server framework. A few common sorts of web assaults Cross-site Scripting (XSS), Cross-Site Ask Forgerty (CSRF), Denny of Benefit (DoS) Man-in-the- middle (MITM), Phishing attacks, Probe, R2L, U2R.

Halfond, W. G., Viegas, J., & Orso, A. [1], Web applications are medium to cyber-attacks, including common ones like SQL injection Wassermann, G., & Su, Z. [2], and inaccessible code execution. In spite of the advancement of countermeasures like firewalls and interruption location frameworks Raponi, S., Caprolu, M., & Di Pietro, R. [3], web assaults remain a critical danger. Investigation appears that over half of web applications amid a 2015-2016 filter contained noteworthy security vulnerabilities. Wrong positive confinements Pietraszek, T. [4] require manual choice of attack-specific highlights and tall untrue positive rates, making it basic to diminish these frameworks. An foundation that requires less mastery and labeled preparing information is required to address these challenges. Fu, X., Lu, X., Peltsverger, B., Chen, S., Qian, K., & Tao, L. [5] battle due to workforce impediments, classification impediments, and wrong positive restrictions. Workforce impediments include in-depth space information of web security, whereas classification restrictions include huge sums of labeled preparing information and the trouble of getting it for subjective custom applications. Su, T., Sun, H., Zhu, J., Wang, S., & Li, Y. [6] Consideration instrument is utilized to screen the organize stream vector composed of parcel vectors produced by the BLSTM demonstrate, which can get the key highlights for arrange activity classification. Numerous convolutional layers are utilized to handle information tests strategies.

II. LITERATURE SURVEY

Zargar, S. [7] In this paper they have discusses about the ceaseless flourishing of the financial advertise, MasterCard volume has until the end of time been impacting these a long time. The blackmail organizations are moreover rising rapidly. Beneath this circumstance, blackmail disclosure has turned into an progressively more critical issue. Be that as it may, the degree o f the distortion is completely much lower than the virtuoso trade, so the unevenness dataset makes this issue altogether more testing. In this paper we mainly prompthow to adjust to the Visa distortion distinguishing proof issue by utilizing supporting procedures and moreover gave a commitment ofthe brief examination between these making a difference methods.

Liu, J., Kantarci, B., & Adams, C. [8] This paper digs into the security vulnerabilities experienced by a wide cluster of Web of Things (IoT) gadgets and applications. The differing nature of IoT systems postures challenges for utilizing common benchmarks just like the NSL-KDD dataset to assess distinctive Arrange Interruption Discovery Frameworks (NIDS). To address this crevice, the paper analyzes particular assaults inside the NSL-KDD dataset that seem affect sensor hubs and systems in IoT situations. Moreover, it assesses eleven machine learning calculations to distinguish these assaults, displaying the comes about of their examination. The consider uncovers that tree-based strategies and gathering strategies perform way better than other machine learning approaches. Eminently, XG Boost rises as the top-performing administered calculation with 97curacy, a Matthews relationship coefficient (MCC) of90.5%, and an Zone Beneath the Bend (AUC) of 99.6%. Moreover, a critical finding is that the Expectation-Maximization (EM) calculation, an unsupervised strategy, too illustrates solid execution in recognizing assaults inside the NSL-KDD dataset, outperforming the exactness of the Naïve Bayes classifier by 22.0%.

Bisong, E. [9] This paper deals about the Education organized to create the information of machine learning, profound learnin g, information science, and cloud computing effortlessly open Prepares you with aptitudes to construct and send large - scale learning models on Google Cloud Stage Covers the programming abilities fundamental for machine learning and profound learningmodeling utilizing the Python stack Incorporates bundles such as Numpy , Tensorflow, Matplotlib, Keras, Pandas and Scikit-learn.

III. STRUCTURE OF LSTM

The LSTM (Long Short-Term Memory) calculation offers unmistakable focuses of intrigued for recognizing web attacks in an end-to- end way compared to other calculations such as CNN, RNN, and customary machine learning calculations. LSTM basically bargains with Long-Term, Conditions, Continuous Modeling, Memory Cells, End-to-End Learning, Capturing Worldly Conditions, Learning Relevant Data.

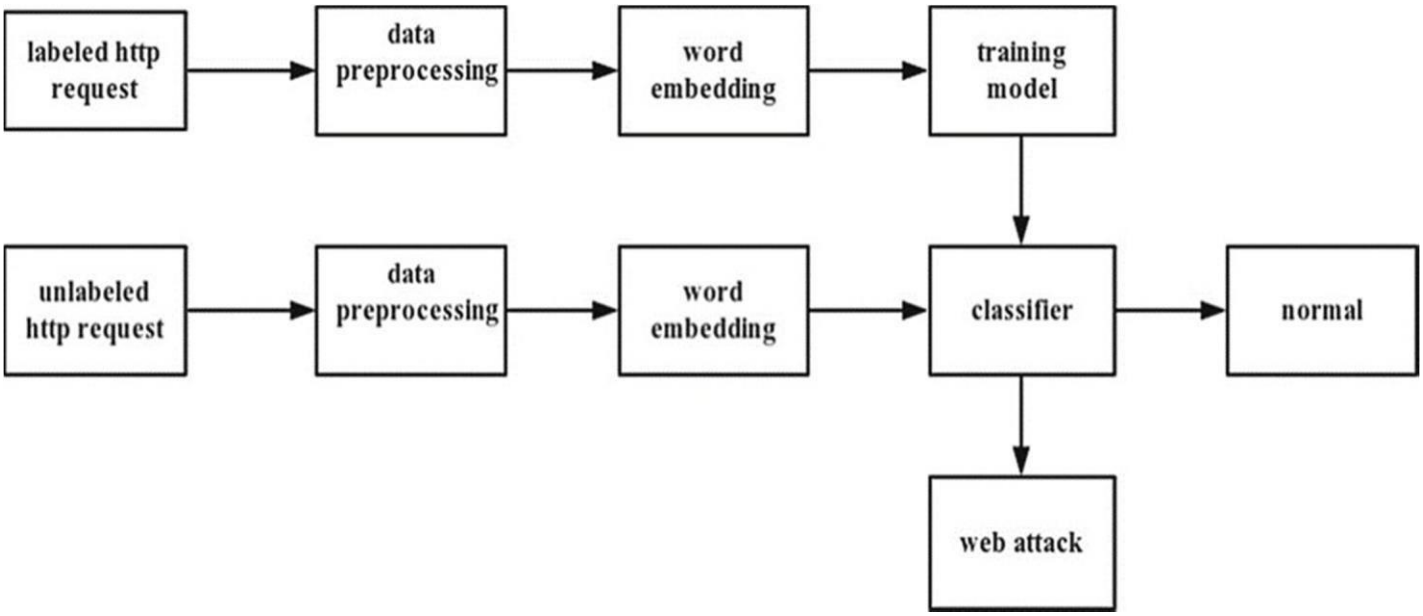


Figure 1**:** Structure of modified LSTM for WebAttacks Source Hao, S.et all [10]

IV. METHODOLOGY

A. Classification of Attacks

We address a assortment of assaults in our organize security endeavors. These incorporate Dissent of Benefit (DoS) assaults like apache2, back, arrive, neptune, mailbomb, unit, processtable, smurf, tear, udpstorm, and worm. Moreover, we center on Test assaults such as ipsweep, mscan, nmap, portsweep, holy person, and satan. Moreover, we consider Unauthorized Get to to Nearby Superuser (U2R) assaults like buffer_overflow, loadmodule, perl, ps, rootkit, sqlattack, and xterm. Finally, we address Unauthorized Get to from a Farther Machine (R2L) assaults like ftp_write, http_tunnel, imap, named, phf, sendmail, snmpgetattack, snmpguess, spy,warezmaster, xsnoop. These categories offer assistance us classify and get it the nature of arrange interruptions, permitting us to create compelling defense instruments.

B. Dataset

The NSL-KDD dataset serves as an upgraded form of the KDD'99 dataset, advertising a profitable asset for analysts within the field of interruption location frameworks (IDS) and organize security. Its primary reason is to supply an compelling benchmark for comparing different interruption discovery strategies. Categorized into four primary classes—Denial of Benefit (DoS), Test, Unauthorized Get to from a Farther Machine (R2L), and Unauthorized Get to to Nearby Superuser Benefits (U2R)—the dataset includes a assorted extendof cyber assaults experienced in arrange situations. Analysts utilize the NSL-KDD dataset to create and assess interruption discovery methods, pointing to upgrade the discovery and moderation of security dangers inside computer systems.

C. Data Analysis

Exploratory Data Examination (EDA) is an principal step in understanding and analyzing a dataset comprehensively. It incorporates a couple of key assignments pointed at picking up bits of information into the data's structure and characteristics. At to begin with, labeling the column names is essential, since it distributes critical identifiers to each incorporate, empowering less requesting explanation and examination. Taking after this, checking for invalid values ensures that there are no misplaced areas inside the dataset, which might something else skew examination comes approximately or obstruct appear execution. Along these lines, data visualization techniques such as making plots and charts are utilized to apparently talk to the transport of data and explore associations between differing highlights. These visualizations offer assistance in recognizing plans, designs, and peculiarities insidethe dataset, in this way enlightening following steps inside the data examination get ready.

D. Feature Selection

Deciding the foremost relevant highlights may be a essential angle of information investigation, supporting in recognizing those traits that contribute most to anticipating the target variable or course name. In this setting, the recorded highlights are positioned based on their relationship with the target course. Highlights like 'dst_host_srv_count', 'logged_in', and 'dst_host_diff_srv_rate' display generally solid relationships with the target lesson, showing their potential centrality in recognizing between distinctive classes of organize activity. On the other hand, highlights such as 'num_shells' and 'urgent' appear weaker relationships with the target course, recommending they may have less prescient control or significance in this setting. Understanding the quality of these relationships guides the selection of highlights for building prescient models, guaranteeing that as it were the foremost enlightening traits are utilized, subsequently improving demonstrate exactness and productivity. Also, the nonattendance of relationship for 'num_outbound_cmds' with the target course highlights its negligible impact in separating between diverse classes of organize activity.

E. Algorithms

In our extent, we utilize a differing extend of calculations to address different angles of our issue. These calculations incorporate Choice Relapse, Bolster Vector Machines (SVM),Calculated Trees, Gaussian Naïve Bayes, as well as profound learning models suchas LSTM ,GRU, CNN, and RNN. Each calculation offers interesting qualities and capabilities suited to diversesorts of information and assignments inside our venture. By leveraging this combination of conventional machine learning and profound learning strategies, we point to viably address the complexities and challenges show in our issue space, eventually moving forward the precision and strength of our arrangements.

F. Implementation Block Diagram

The flowchart starts with the introductory setup, where an application is opened and essential bundles are imported to encourage the advancement handle. Taking after this, the dataset is investigated and experiences information preprocessing, which regularly includes errands like cleaning the information, taking care of lost values, and changing the information into a appropriate organize for investigation. Another, the flowchart delineates a few key steps within the include designing handle, counting include era, include choice, and name encoding. These steps offer assistance in planning the information for encourage investigation and modeling. The flowchart at that point moves to the preparing and testing stages of the venture. Within the preparing stage, different machine learning calculations are connected, counting KFold cross-validation, calculated relapse, back vector machines (SVM), credulous Bayes, irregular timberlands, stacking classifiers, and voting classifiers. Also, profound learning procedures such as CNN, LSTM, gated repetitive units (GRU), and repetitive neural systems (RNN) are utilized amid the testing stage to assess the execution of the models.

At long last, the flowchart concludes with the execution of user registration and login functionalities, allowing clients to supply input and get the ultimate yield or result from the online location. This organized approach laid out within the flowchart guarantees a precise and organized advancement handle, driving to the creation of a useful and user-friendly online stage.

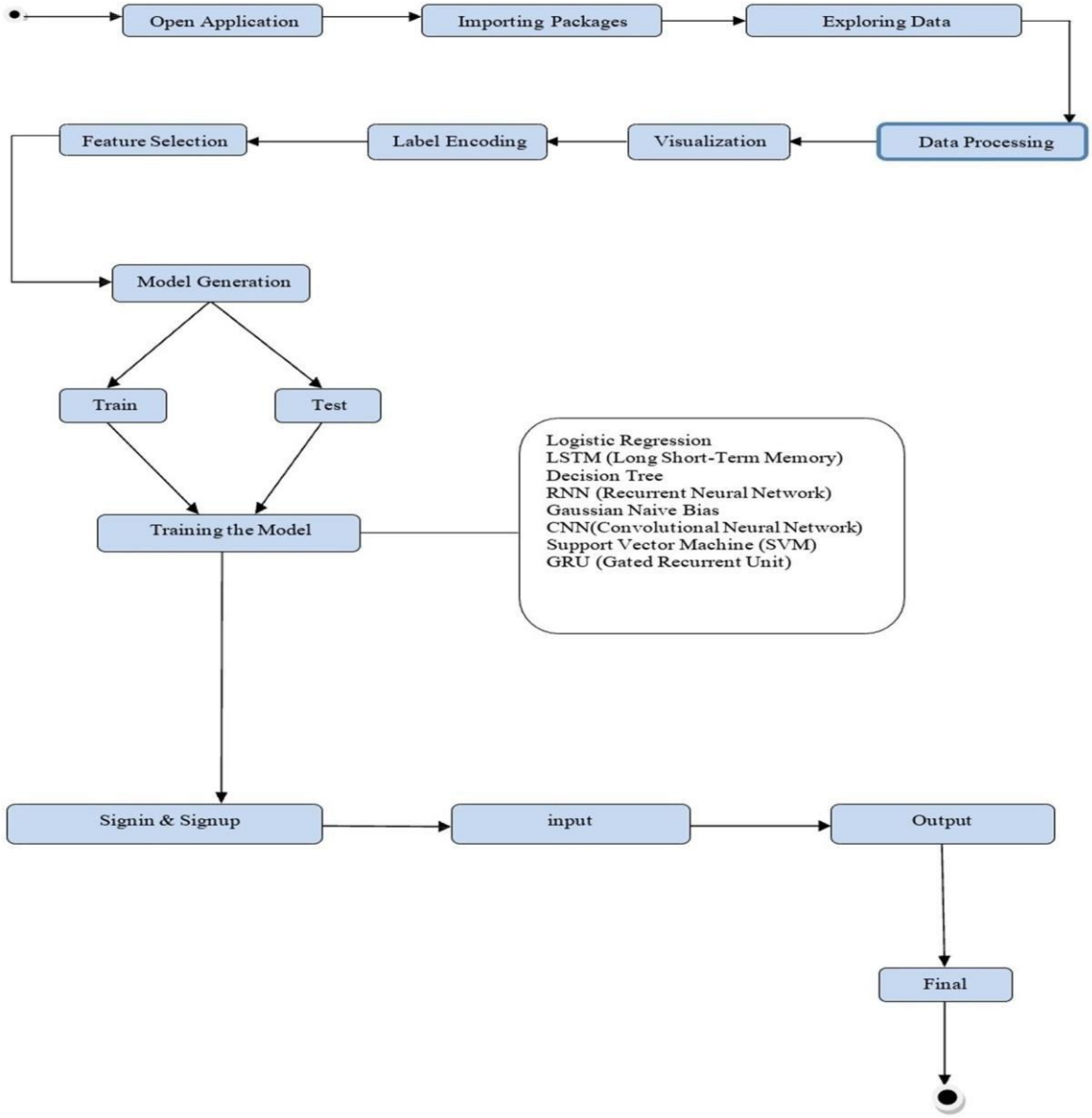


Figure 2**:** Block Diagram of Proposed System's Implementation

V. PERFORMANCE VALIDATION

A. Performance Validation Of Machine Learning Algorithms

Table 1: Precision

Algorithm	Normal-Attack	DoS-Attack	R2L-Attack	Probe-Attack	U2R-Attack
Decision Tree	0.967720	0.972861	0.744722	0.826865	0.0
Logistic Regression	0.888980	0.944034	0.0	0.857955	0.0
SVM	0.169973	0.716787	0.057082	0.169648	0.0
GNB	0.946300	0.974133	0.043536	0.341962	0.002609

Table 2: Recall

Algorithm	Normal-Attack	DoS-Attack	R2L-Attack	Probe-Attack	U2R-Attack
Decision Tree	0.954000	0.968885	0.487437	0.936335	0.0
Logistic Regression	0.980076	0.939841	0.0	0.439190	0.0
SVM	0.087330	0.097022	0.440955	0.611871	0.0
GNB	0.678349	0.746969	0.204774	0.576322	0.974359

Table 3: F1-Score

Algorithm	Normal-Attack	DoS-Attack	R2L-Attack	Probe-Attack	U2R-Attack
Decision Tree	0.960811	0.970869	0.589218	0.878202	0.0
Logistic Regression	0.932308	0.941933	0.0	0.580976	0.0
SVM	0.115380	0.170910	0.101080	0.265644	0.0
GNB	0.790228	0.845560	0.071806	0.429236	0.005204

Table 4: Support

Algorithm	Normal-Attack	DoS-Attack	R2L-Attack	Probe-Attack	U2R-Attack
Decision Tree	53956.000000	36703.000000	796.000000	9283.000000	39.0
Logistic Regression	53956.000000	36703.000000	796.0	9283.000000	39.0
SVM	53956.000000	36703.000000	796.000000	9283.000000	39.0
GNB	53956.000000	36703.000000	796.000000	9283.000000	39.000000

B. Performance Validation Of Deep Learning Algorithms

Table 5: Average Accuracy across k-Folds (k=10)

Algorithm	Average Accuracy across k-folds (K=10)
GRU	0.9850327432155609
LSTM	0.9848382592201232
RNN	0.9822266280651093
CNN	0.9752728700637817

VI. PERFORMANCE VALIDATION

A. Comparison Of Algorithms

Table 6: Comparison of Algorithms

Algorithm	Accuracy	Precision	F1-Score	Recall	Sensitivity	Specificity
Decision Tree	95.336376	70.186724	67.715002	66.592677	97.558317	98.265495
Logistic Regression	90.208375	53.396384	48.559363	46.693533	93.774488	99.010873
SVM	14.054376	21.989918	12.874339	24.240124	15.304756	75.351641
Gaussian Naïve Bayes	68.779520	46.208900	42.779500	62.191868	94.777563	98.068995
GRU	98.654495	72.935717	74.018918	75.251725	99.337605	99.660557
LSTM	98.765628	74.384986	74.783166	75.194744	99.207469	99.856809
RNN	98.543362	76.799996	72.906212	70.491976	99.197049	99.706723
CNN	97.983727	75.900331	71.622521	68.963413	99.043270	99.586435

Table 6 interprets the data of the comparison of the accuracy, precision, f1-score, recall, sensitivity and specificity of Decision Tree, Logistic Regression, SVM, Gaussian Naïve Bayes, GRU, LSTM, RNN and CNN algorithms. We can have a clear view that in terms of accuracy, f1-score and specificity LSTM has the high performance and in precision RNN, in recall and sensitivity GRU.

B. Figures

Figure 3 shows the graphical representation of the accuracy, precision, f1-score, recall, sensitivity and specificity of the algorithms.

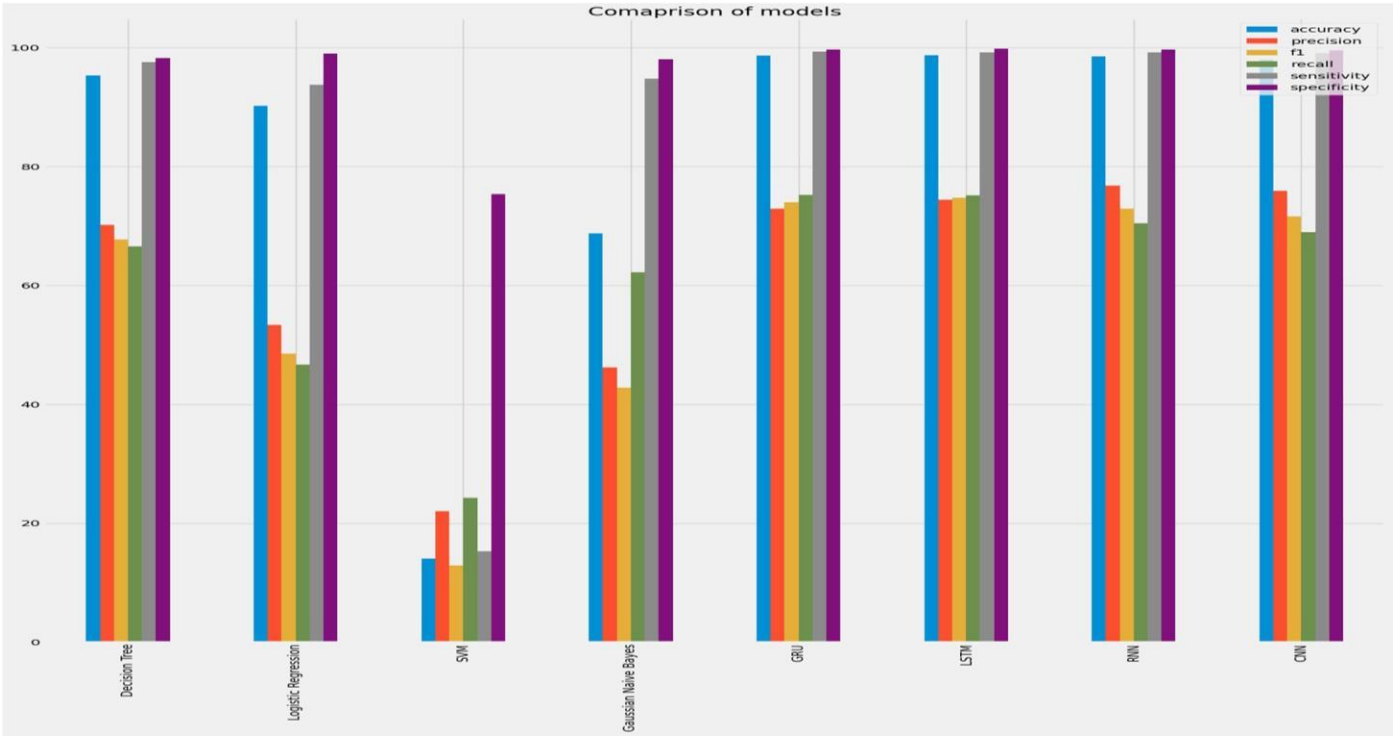


Figure 3: Graphical representation of Comparison of models

Figure 4 shows the graphical representation of the comparison of cross validation accuracy of algorithms.

Cross Validation Accuracy: It is a robust technique used as a performance metric to compare the efficiency of different models.

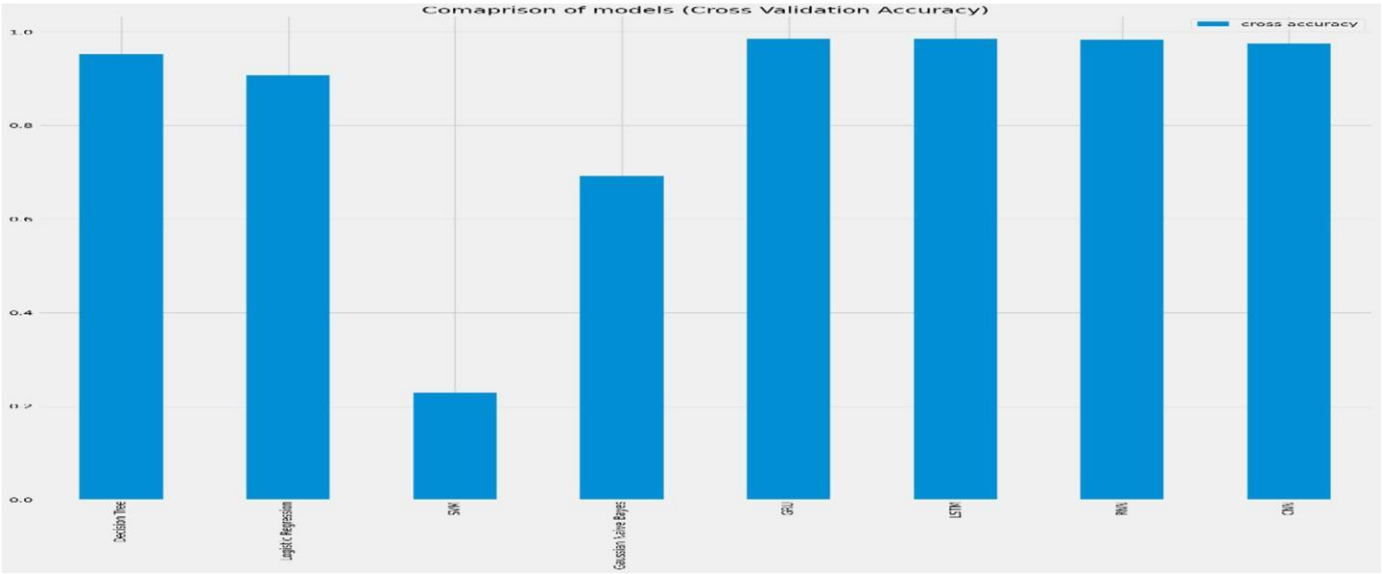


Figure 4: Graphical representation of Cross Validation Accuracy

Figure 5 shows the execution time of different models to train and test the dataset. We can generalize that the Deep Learning models takes longer time to train than the ML algorithms. LSTM and GAN take the maximum time to train the model.

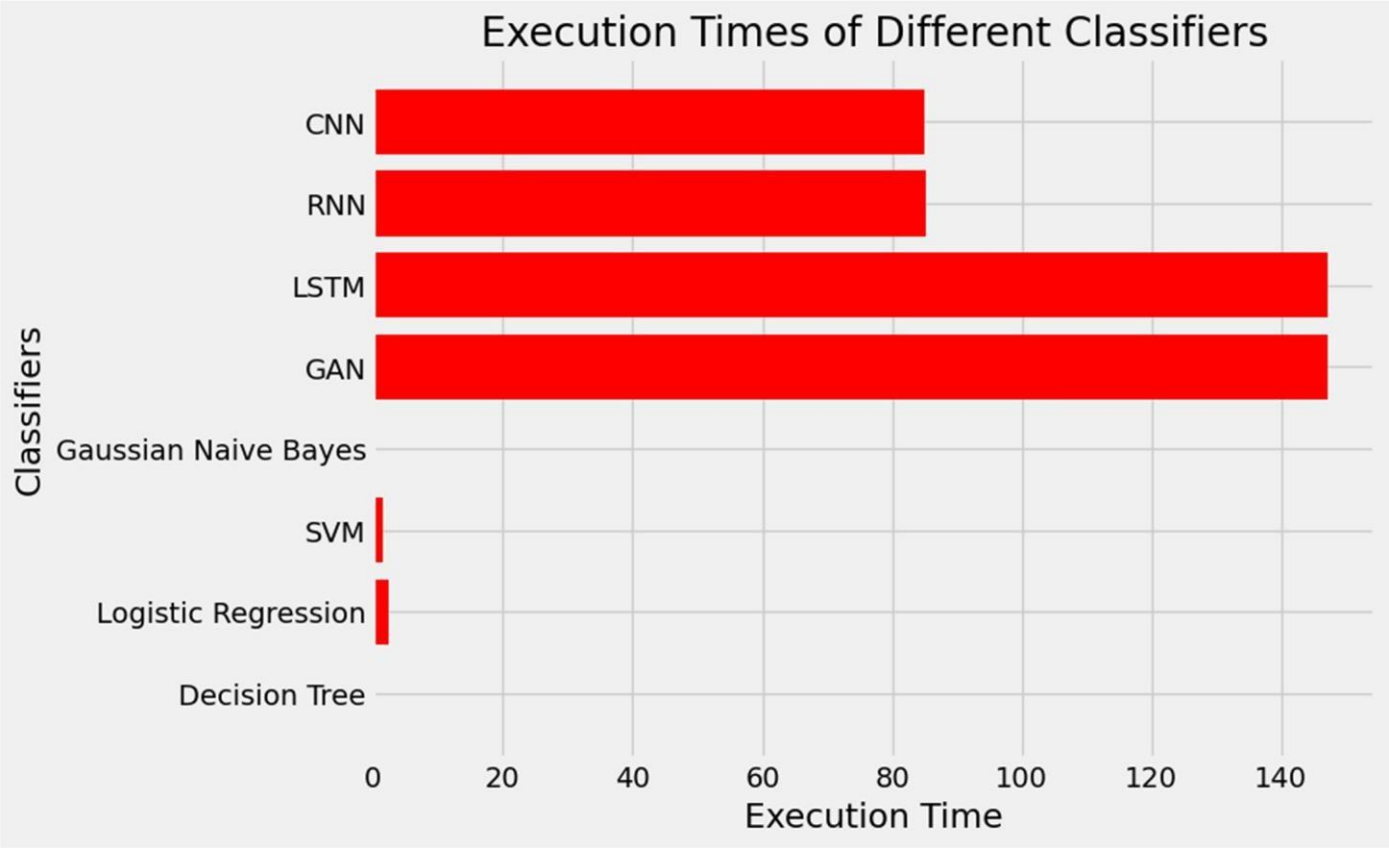


Figure 5: Graphical Representation of Execution Times of algorithms

VII. CONCLUSION

The ponder presents a novel approach for arrange assault discovery, leveraging profound methods like LSTM nearby conventional machine learning models. Through tests conducted on NSL-KDD dataset, both LSTM and different machine learning calculations wereutilized for execution assessment. The discoveries of this investigation not as it were illustrating the adequacy of the LSTM demonstrate but too highlight its predominance over existing state-of-the-art approaches, counting machine learning calculations. This approval through execution comparison underscores the potential of LSTM as a strong arrangement for organize assault discovery in real-world scenarios. In future endeavors, the center will be on optimizing the computational productivity of the LSTM demonstrate. This involves refining its design to diminish computational costs without compromising discovery precision. Furthermore, the proposed demonstrate will experience encourage preparing on assorted sorts of assaults to guarantee its adequacy in tending to modern and advancing dangers. In outline, the consider presents a promising progression in arrange assault location, advertising a profound learning approach with LSTM that outperforms conventional machine learning strategies in terms of execution. The commitment to future enhancements, counting computational optimization and improved versatility to rising dangers, implies a proactive position in progressing cybersecurity measures for arrange defense.

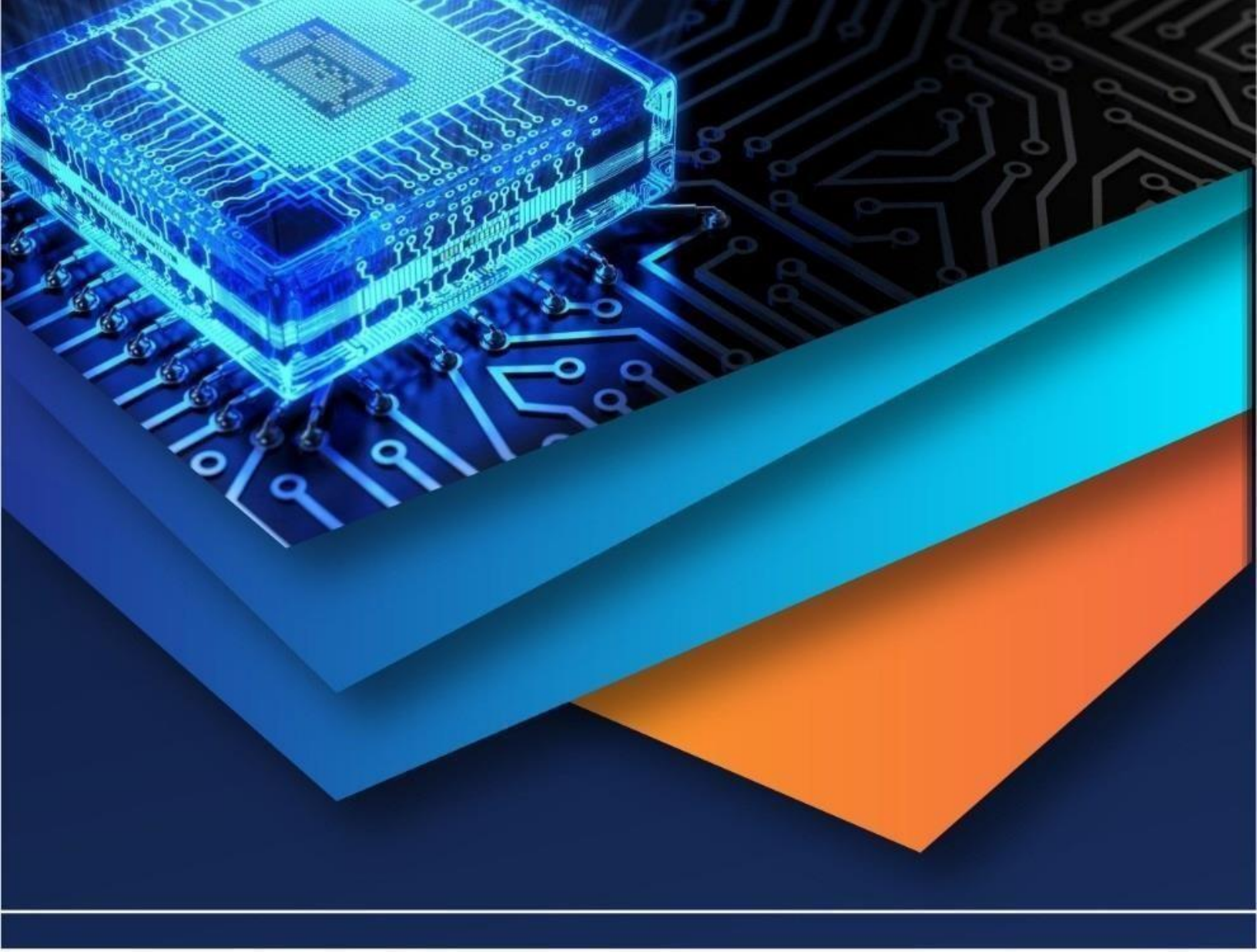
REFERENCES

[1] Halfond, W. G., Viegas, J., & Orso, A. (2006, March). A classification of SQL-injection attacks and countermeasures. In Proceedings of the IEEE international symposium on secure software engineering (Vol. 1, pp. 13-15). IEEE.

[2] Wassermann, G., & Su, Z. (2008, May). Static detection of cross-site scripting vulnerabilities. In Proceedings of the 30th international conference on Software engineering (pp. 171-180).

[3] Raponi, S., Caprolu, M., & Di Pietro, R. (2019). Intrusion detection at the network edge: Solutions, limitations, and future directions. In Edge Computing– EDGE 2019: Third International Conference, Held as Part of the Services Conference Federation, SCF 2019, San Diego, CA, USA, June 25–30, 2019, Proceedings 3 (pp. 59- 75). Springer International Publishing.

[4] Pietraszek, T. (2004). Using adaptive alert classification to reduce false positives in intrusion detection. In Recent Advances in Intrusion Detection: 7th International Symposium, RAID 2004, Sophia Antipolis, France, September 15-17, 2004. Proceedings 7 (pp. 102-124). Springer Berlin Heidelberg.
[5] Fu, X., Lu, X., Peltsverger, B., Chen, S., Qian, K., & Tao, L. (2007, July). A static analysis framework for detecting SQL injection vulnerabilities. In 31st annual international computer software and applications conference (COMPSAC 2007) (Vol. 1, pp. 87-96). IEEE.
[6] Su, T., Sun, H., Zhu, J., Wang, S., & Li, Y. (2020). BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset. IEEE Access, 8, 29575- 29585.
[7] Zargar, S. (2021). Introduction to sequence learning models: RNN, LSTM, GRU. Department of Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, North Carolina, 27606.
[8] Liu, J., Kantarci, B., & Adams, C. (2020, July). Machine learning-driven intrusion detection for Contiki-NG-based IoT networks exposed to NSL-KDD dataset. In Proceedings of the 2nd ACM workshop on wireless security and machine learning (pp. 25-30).
[9] Bisong, E. (2019). Building machine learning and deep learning models on Google cloud platform (pp. 59-64). Berkeley, CA: Apress.
[10] Hao, S., Long, J., & Yang, Y. (2019, April). Bl-ids: Detecting web attacks using bi-lstm model based on deep learning. In International conference on security and privacy in new computing environments (pp. 551-563). Cham: Springer International Publishing.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY


iJRASET
International Journal for Research in Applied Science & Engineering Technology
ISSN No. : 2321-9653
IJRASET is indexed with Crossref for DOI-DOI : 10.22214
Website : www.ijraset.com, E-mail : ijraset@gmail.com
ISRA Journal Impact Factor: 7.429
45.98
INDEX COPERNICUS
THOMSON REUTERS
Researcher ID: N-9681-2016
10.22214/IJRASET
TOGETHER WE REACH THE GOAL
SJIF 7.429
Certificate
It is here by certified that the paper ID : IJRASET59078, entitled
Application of Robust Software Modelling Tool for Web Attacks Detection
by
A. Mounika Rajeswari
after review is found suitable and has been published in
Volume 12, Issue III, March 2024
in
International Journal for Research in Applied Science & Engineering Technology
(International Peer Reviewed and Refereed Journal)
Good luck for your future endeavors
Editor in Chief, iJRASET


iJRASET
International Journal for Research in Applied Science & Engineering Technology
ISSN No. : 2321-9653
IJRASET is indexed with Crossref for DOI-DOI : 10.22214
Website : www.ijraset.com, E-mail : ijraset@gmail.com
ISRA Journal Impact Factor: 7.429
45.98
INDEX COPERNICUS
THOMSON REUTERS
Researcher ID: N-9681-2016
10.22214/IJRASET
doi
cross ref
TOGETHER WE REACH THE GOAL
SJIF 7.429
Certificate
It is here by certified that the paper ID : IJRASET59078, entitled
Application of Robust Software Modelling Tool for Web Attacks Detection
by
Kalluri Rishita
after review is found suitable and has been published in
Volume 12, Issue III, March 2024
in
International Journal for Research in Applied Science & Engineering Technology
(International Peer Reviewed and Refereed Journal)
Good luck for your future endeavors
Editor in Chief, iJRASET


iJRASET
ISSN No. : 2321-9653
International Journal for Research in Applied Science & Engineering Technology
IJRASET is indexed with Crossref for DOI-DOI : 10.22214
Website : www.ijraset.com, E-mail : ijraset@gmail.com
ISRA Journal Impact Factor: 7.429
45.98
INDEX COPERNICUS
THOMSON REUTERS
Researcher ID: N-9681-2016
10.22214/IJRASET
TOGETHER WE REACH THE GOAL
SJIF 7.429
Certificate
It is here by certified that the paper ID : IJRASET59078, entitled
Application of Robust Software Modelling Tool for Web Attacks Detection
by
Lanka Shriya
after review is found suitable and has been published in
Volume 12, Issue III, March 2024
in
International Journal for Research in Applied Science & Engineering Technology
(International Peer Reviewed and Refereed Journal)
Good luck for your future endeavors
Editor in Chief, iJRASET


iJRASET
International Journal for Research in Applied Science & Engineering Technology
IJRASET is indexed with Crossref for DOI-DOI : 10.22214
Website : www.ijraset.com, E-mail : ijraset@gmail.com
ISSN No. : 2321-9653
ISRA Journal Impact Factor: 7.429
45.98
INDEX COPERNICUS
THOMSON REUTERS
Researcher ID: N-9681-2016
10.22214/IJRASET
crossref
TOGETHER WE REACH THE GOAL
SJIF 7.429
Certificate
It is here by certified that the paper ID : IJRASET59078, entitled
Application of Robust Software Modelling Tool for Web Attacks Detection
by
Balla Ganesh
after review is found suitable and has been published in
Volume 12, Issue III, March 2024
in
International Journal for Research in Applied Science & Engineering Technology
(International Peer Reviewed and Refereed Journal)
Good luck for your future endeavors
Editor in Chief, iJRASET