A

Major Project

On

AUTO ENCODER SYSTEM FOR INTRUSION DETECTION

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

K.THARAKA RAM (197R1A05L5)

K.THARUN KUMAR(197R1A05M2)

Under the Guidance of

DR.SOMA SHEKAR

(Associate Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.

2019-2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATION

This is to certify that the Project entitled "**AUTOENCODER SYSTEM FOR INTRUSION DETECTION"** being submitted by **K.THARAKARAM (197R1A05L5) K.THARUN KUMAR (197R1A05M2)** in partial fulfillment of the requirements for award of degree B.TECH for the Computer science and engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bondafide work carried out by them under our guidance and supervision during the year 2022-23.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

DR.SOMA SHEKAR
(Associate Professor)
INTERNAL GUIDE

DR.A. RAJI REDDY
DIRECTOR

Dr. K. Srujan Raju
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on _____

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide J. Narasimha Rao, Associate Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **Dr. Punyaban Patel, Ms. Shilpa, Dr. T. Subha Mlastan Rao & J. Narasimharao** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju**, Head, Department of Computer Science and Engineering, **Dr. Ashuthosh Saxena,** Dean R&D, and **Dr. D T V Dharmajee Rao,** Dean Academics for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy**, Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion ofthis project.

K.Tharaka ram (197R1A05L5)

K.Tharun kumar (197R1A05M2)

ABSTRACT

Web applications are popular targets for cyber-attacks because they are network-accessible and often contain vulnerabilities. In this project first, we evaluate the feasibility of an unsupervised/semi-supervised approach for web attack detection based on the Robust Software Modeling Tool (RSMT), which autonomically monitors and characterizes the runtime behaviour of web applications. Second, we describe how RSMT trains a stacked denoising auto encoder to encode and reconstruct the call graph. Third, we analyse the results of empirically testing RSMT on both synthetic datasets and production applications with intentional vulnerabilities.

LIST OF FIGURES/TABLES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1.INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

Attacks could be recognized as the attempts to bypass security policies of the system, which gives attackers easier access to obtain or modify information, even destroying the system. With technologies developing on wireless communication systems, serious threats to network security, especially security of wireless communication systems, have been proposed by more frequent network attack activities, due to openness characteristics of wireless channels. Since we are now in machine learning and big data epoch [9], cybersecurity in wireless communication systems is important for users to protect network, computer, and data from attacks.There exist variant kinds of attacks for cyber systems, such as flooding, distributed denial of service, abnormal packet attack, and spoofing.To deal with such attack threads to cybersecurity, researchers have proposed many solutions.

1.2 PROJECT PURPOSE

An Infringement deection sysems,detects the cyber attacks while monitoing web applications. Exising implemantation of Machine learning based intrusion detection system counts on labeled training data to learn what should be considered normal and abnormal behaviour which can be hard and expensive in large scale production web applications.Various supervised and unsupervised machine learning techniques can be used or integrated with other algorithms in ADNIDS to enhance intrusion detection performance and increase the classification rate of machine learning algorithms, such as decision tree, random forest, self-organization maps (SOM), and support vector machine (SVM), which have been utilized to detect and classify intrusions.

1.3 PROJECT FEATURES

We provide a detailed description of the dataset used, along with the data pre-processing procedure adopted here. The detailed structure of the model is also presented. Finally, a performance analysis of the model is provided followed by a comparative study with other classifiers and models found in the literature. The experiments are done on a Windows 10 – 64-bit PC with 12 GB RAM and CPU Intel(R) i5@2.7 GHz.

An Intrusion Detection System (IDS) dynamically monitors the actions of a specific environment, for example, the network traffic, syslog records or system calls of a given operating system, in order to determine if those actions are a legitimate use or a symptom related to a given attack. These systems are usually classified into Network-based Intrusion Detection Systems (NIDS) and Host-based Intrusion Detection Systems (HIDS). An NIDS works on feature vectors that comprise summarized information related to network traffic within a specified time interval while a HIDS is located on a specific host and monitors information related to the system.

2.SYSTEMANALYSIS

2.SYSTEM ANALYSIS

SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analysed . The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “what must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst has a firm understanding of what is to be done.

2.1 PROBLEM DEFINITION

In web different types of attacks are there such as database attack, dynamic code execution or XSS attack and all existing techniques like Intrusion detection system (require all possible attack signature to detect attacks and not possible for experience person also to identify all those signature), machine learning algorithms such as SVM require pre label dataset to predict attacks and this algorithm may not work if attack not available in machine learning train model.

0

2.2 EXISTING SYSTEM

Detection of attacks on web applications is monitored by an intrusion detection system, which outputs alerts when an attempt is detected. Traditionally, intrusion detection systems extract features out of network packets or from string characteristics of input that can be used to analyze attacks. In addition, hand-selecting features requires in-depth knowledge of the security domain and can be time consuming.

2.2.1DRABACKS OF EXISTING SYSTEM

In-depth domain knowledge of web security is needed for web developers and network operators to deploy these systems.An experienced security expert is often needed to determine what features are relevant to extract from network packages, binaries, or other inputs for intrusion detection systems.Due to the large demand and relatively low barrier to entry into the software profession, however, many developers lack the necessary knowledge of secure coding practices.

2.3 PROPOSED SYSTEM

In this project first, we evaluate the feasibility of an unsupervised/semi-supervised approach for web attack detection based on the Robust Software Modeling Tool (RSMT), which autonomically monitors and characterizes the runtime behavior of web applications. Second, we describe how RSMT trains a stacked denoising auto encoder to encode and reconstruct the call graph. Third, we analyze the results of empirically testing RSMT on both synthetic datasets and production applications with intentional vulnerabilities.

2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

In this project, instead of just describing a single target object, this model detects multiple target objects and generating grammatically correct caption.
Attacks can have significantly different characteristics. Monitoring can have a significant performance cos.
Collecting labeled attack training data.
The structure and functionality of the robust software modeling tool (RSMT).

2.4 FEASIBILITY STUDY

The feasibility of the project is analysed in this phase and a 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. Three key considerations involved in the feasibility analysis:

- EconomicFeasibility
- Technical Feasibility
- Social Feasibility

2.4.1 ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on a project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require. The following are some of the important financial questions asked during preliminary investigation:

The costs conduct a full system investigation.

- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.
- Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication that the system is economically possible for development

2.4.2 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. The developed system must have a modest requirement,as only minimal or null changes are required for implementing this system.

2.4.3 BEHAVIORAL FEASIBILITY

This includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioural aspects are considered carefully and conclude that the project is behaviourally feasible.

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements:

Processor	:Intel Core i5
RAM	: 4GB
Hard Disk	:Minimum 500 GB

2.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system.

The following are some software requirements:

Operating System	:from Windows 8
Programming Language	:Python 3.7
Browser	:Google Chrome

3.ARCHITECTURE

3.ARCHITECTURE

3.1 PROJECT ARCHITECTURE

This project architecture shows the procedure followed for classification, starting from input to final output.

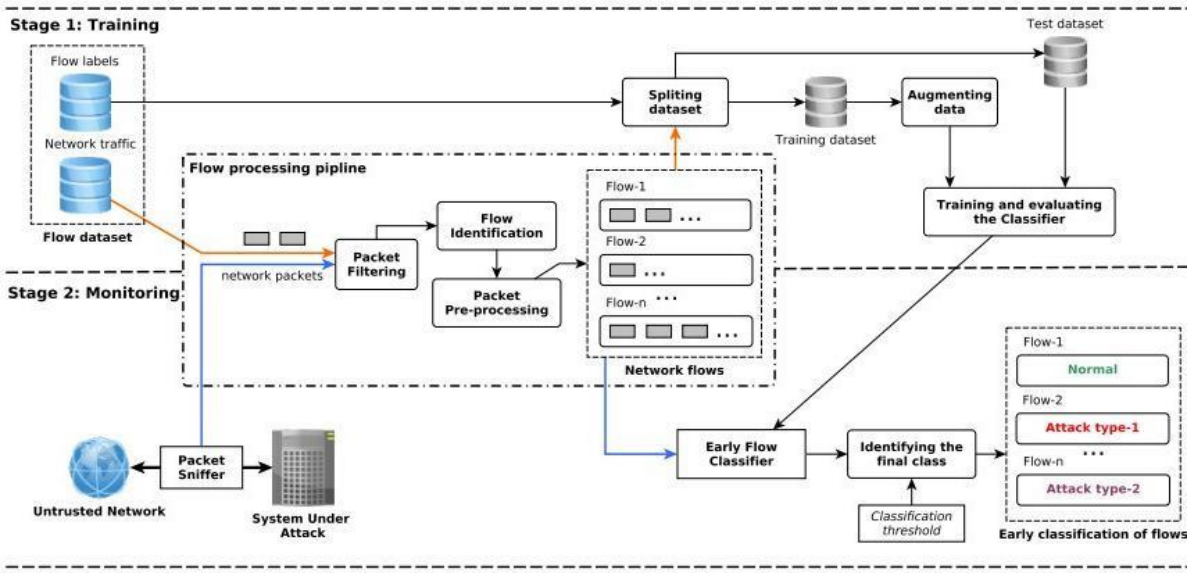


Figure 3.1: Project Architecture Autoencoder system for intrusion detection.

3.2 DESCRIPTION

Shows the high-level workflow of RSMT's web attack monitoring and detection system. This system is driven by one or more environmental stimuli (a), which are actions transcending process boundaries that can be broadly categorized as either manual (e.g., human interaction-driven) or automated (e.g., test suites and fuzzers) inputs. The manifestation of one or more stimuli results in the execution of various application behaviors. RSMT attaches an agent and embeds lightweight shims into an application (b). These shims do not affect the functionality of the software, but instead serve as probes that allow efficient examination of the inner workings of software applications. The events tracked by RSMT are typically control flow-oriented, though dataflow-based analysis is also possible.

3.3 USE CASE DIAGRAM

In the use case diagram, we have basically one actor who is the user in the trained model. A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

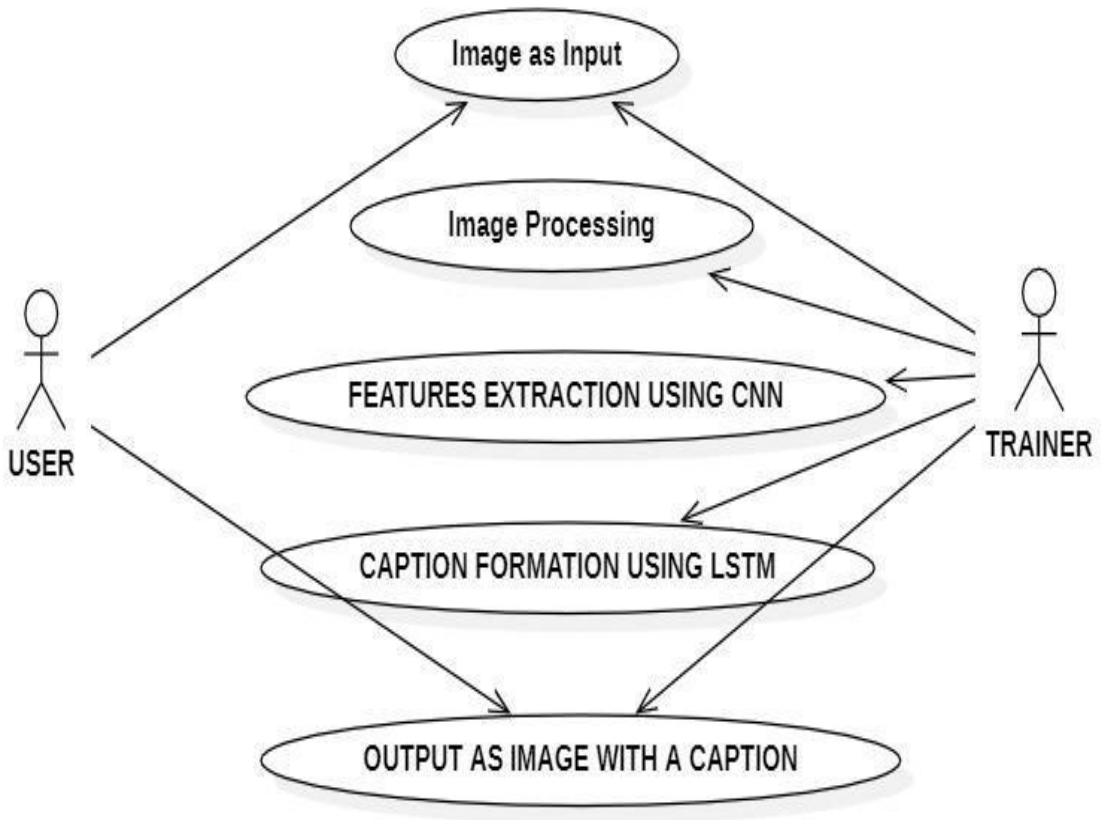


Figure 3.2: Use Case Diagram Autoencoder System for Intrusion Detection

3.4 CLASS DIAGRAM

Class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations(methods),and the relationships among objects.

Figure 3.3: Use Case Diagram Autoencoder System For Intrusion Detection

3.5 SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development.

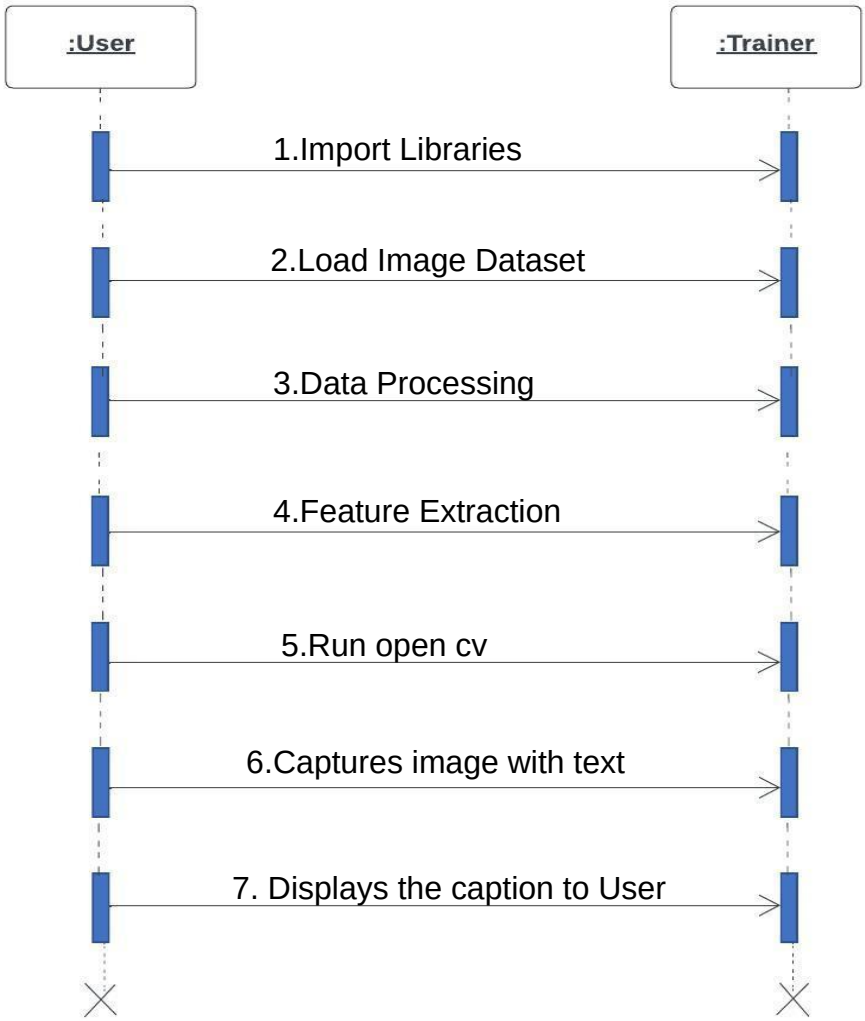


Figure 3.4: Sequence Diagram Autoencoder System For Intrusion Detection

3.6 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of work flows of stepwise activities and actions with support for choice, iteration and concurrency. They can also include elements showing the flow of data between activities through one or more data stores.

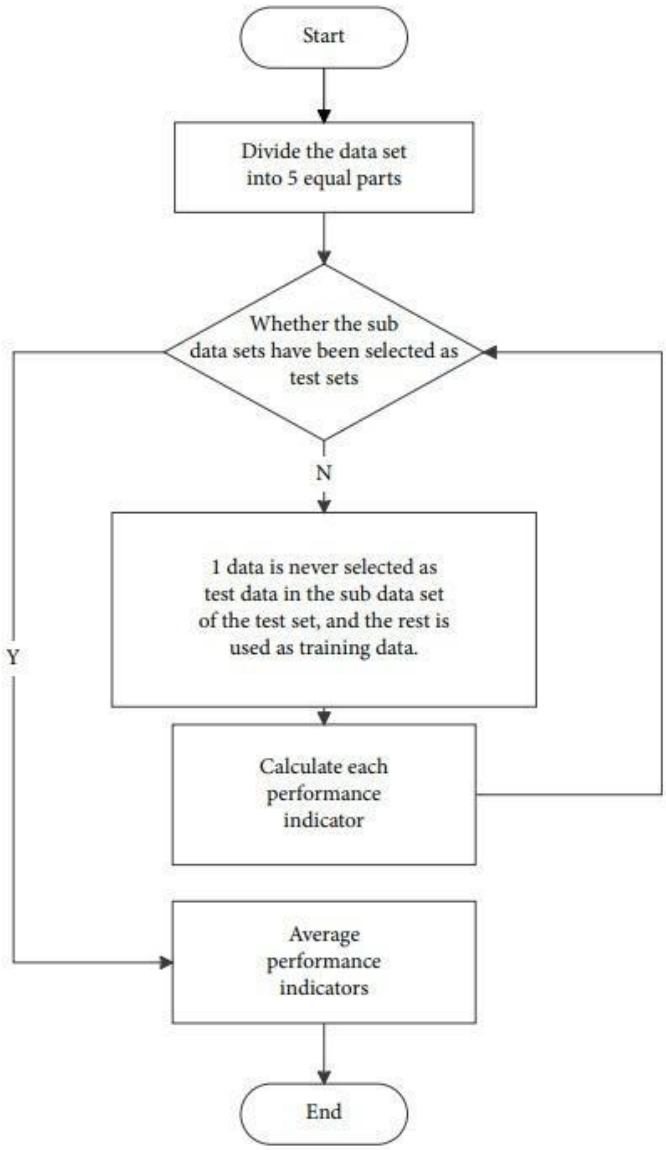


Figure 3.5: Activity Diagram Autoencoder System For Intrusion Detection

4.IMPLEMENTATION

4. IMPLEMENTATION

4.1 ALGORITHMS

4.1.1 SVM Algorithm

SVM can also be used for detecting web attacks. the SVM model is trained on a set of labelled data, where each instance represents a web request or transaction and is labelled as either a normal or malicious request. The input features for each instance may include things like the requested URL, HTTP method, user agent, IP address, and other relevant attributes.The SVM model then learns to classify new incoming requests as either normal or malicious based on the learned classification boundary. This can help in identifying and blocking potentially harmful requests before they can cause damage to the web application or system.

5.1.2 Naive Bayes Algorithm

The Naive Bayes algorithm works by first estimating the prior probabilities of the two classes (normal and malicious) based on the training data. It then estimates the conditional probability of each feature given the class label using the training data. These probabilities are combined using Bayes' theorem to calculate the posterior probability of each class given the feature values.During classification, the algorithm calculates the posterior probability of each class given the feature

5.1.3 AutoEncoder Algorithm

An autoencoder can be trained on a large set of normal network traffic data to learn the underlying patterns and structures in the data.The algorithm works by first encoding the input data into a compressed representation using a neural network. The encoder network reduces the dimensionality of the input data by transforming it into a lower-dimensional space. The compressed representation is then decoded back into the original input data using a decoder network. The decoder network tries to reconstruct the original input data from the compressed representation.During training, the autoencoder is trained to minimize the difference between the original input data and the reconstructed data. The objective is to learn a compressed representation that captures the most important features of the input data while minimizing the reconstruction error.

During testing, the autoencoder is used to detect anomalies in the input data. Anomalies are detected by measuring the reconstruction error between the original input data and the reconstructed data. If the reconstruction error exceeds a certain threshold, the input data is classified as anomalous and may indicate the presence of web attacks.

5.1.4 Long Short Term Memory Algorithm

The LSTM architecture consists of multiple memory cells that can store information over a long period of time. The memory cells are connected through gates that control the flow of information into and out of the cells. The gates are controlled by activation functions that determine the amount of information that is retained or discarded.During training, the LSTM model is trained to minimize the difference between the predicted output and the actual output. The objective is to learn a model that can accurately predict the next output in the sequence given the previous inputs.During testing, the LSTM model can be used to detect anomalies in the network traffic data. Anomalies are detected by comparing the predicted output with the actual output and measuring the difference between them. If the difference exceeds a certain threshold, the input data is classified as anomalous and may indicate the presence of web attacks.

4.2 SAMPLE CODE

```
from tkinter import *
from tkinter import
messagebox
from tkinter import *
from tkinter import
simpledialog
import tkinter
from tkinter import
filedialog
from
tkinter.filedialog
import
askopenfilename
import
matplotlib.pyplot as
plt
from sklearn.metrics
import
accuracy_score
from
sklearn.model_select
ion import
train_test_split
from sklearn import
svm
import pandas as pd
from sklearn.metrics
import f1_score
```

```
from sklearn.metrics
import recall_score
from sklearn.metrics
import
precision_score
from
sklearn.preprocessin
g import
LabelEncoder,
OneHotEncoder
from keras.models
import Sequential
from keras.layers
import Dense
from keras.layers
import LSTM
import numpy as np
import math
from sklearn.metrics
import
mean_squared_error
from keras.layers
import Input
from keras.models
import Model
from
sklearn.naive_bayes
import
MultinomialNB
from sklearn.metrics
import
confusion_matrix,
precision_recall_cur
ve


main = tkinter.Tk()
main.title("Detecting
web attacks")
main.geometry("130
0x1200")

global filename
global classifier
global
svm_precision,auto_
precision,lstm_precis
```

```
ion,naive_precision
global
svm_fscore,auto_fsc
ore,lstm_fscore,naiv
e_fscore
global
svm_recall,auto_reca
ll,lstm_recall,naive_r
ecall
global X_train,
X_test, y_train,
y_test

def uploadDataset():
    global filename
    filename =
filedialog.askopenfil
ename(initialdir="da
taset")

pathlabel.config(text
=filename)
    text.delete('1.0',
END)

text.insert(END,filen
ame+" loaded\n");

def
prediction(X_test,
cls):  #prediction
done here
    y_pred =
cls.predict(X_test)
    for i in
range(len(X_test)):
      print("X=%s,
Predicted=%s" %
(X_test[i],
y_pred[i]))
    return y_pred

def
cal_accuracy(y_test,
y_pred, details):
    accuracy =
accuracy_score(y_te
```

```
st,y_pred)*100
    return accuracy

def generateModel():
    global X_train,
X_test, y_train,
y_test
    text.delete('1.0',
END)
    df =
pd.read_csv(filenam
e)
    X = df.iloc[:, :-
1].values
    Y = df.iloc[:, -
1].values
    labelencoder_X =
LabelEncoder()
    X[:,0] =
labelencoder_X.fit_t
ransform(X[:,0])
    X[:,2] =
labelencoder_X.fit_t
ransform(X[:,2])
    Y =
labelencoder_X.fit_t
ransform(Y)
    onehotencoder =
OneHotEncoder()
    X =
onehotencoder.fit_tr
ansform(X).toarray()
    X_train, X_test,
y_train, y_test =
train_test_split(X, Y,
test_size = 0.2,
random_state = 0)

text.insert(END,"Dat
aset Length :
"+str(len(X))+"\n");

text.insert(END,"Spl
itted Training Length
:
"+str(len(X_train))+"
\n");
```

```
text.insert(END,"Spl
itted Test Length :
"+str(len(X_test))+"\
n\n");

def svmAlgorithm():
    global classifier
    global
svm_precision
    global svm_fscore
    global svm_recall
    text.delete('1.0',
END)
    cls =
svm.SVC(C=2.0,ga
mma='scale',kernel =
'rbf', random_state =
2)
    cls.fit(X_train,
y_train)
    prediction_data =
prediction(X_test,
cls)
    classifier = cls
    svm_acc =
cal_accuracy(y_test,
prediction_data,'SV
M Accuracy')/2
    svm_fscore =
f1_score(y_test,
prediction_data)/2
    svm_precision =
precision_score(y_te
st, prediction_data)/2
    svm_recall =
recall_score(y_test,
prediction_data)/2

text.insert(END,"SV
M Accuracy :
"+str(svm_acc)+"\n"
);

text.insert(END,"SV
M Precision :
"+str(svm_precision)
```

```
+"\n");

text.insert(END,"SV
M Recall :
"+str(svm_recall)+"\
n");

text.insert(END,"SV
M FScore :
"+str(svm_fscore)+"\
n");

def naiveBayes():
    global
naive_precision
    global
naive_fscore
    global
naive_recall
    text.delete('1.0',
END)
    cls =
MultinomialNB()
    cls.fit(X_train,
y_train)
    prediction_data =
prediction(X_test,
cls)
    naive_acc =
cal_accuracy(y_test,
prediction_data,'SV
M Accuracy')/2
    naive_fscore =
f1_score(y_test,
prediction_data)/2
    naive_precision =
precision_score(y_te
st, prediction_data)/2
    naive_recall =
recall_score(y_test,
prediction_data)/2

text.insert(END,"Nai
ve Bayes Accuracy :
"+str(naive_acc)+"\n
");
```

```
text.insert(END,"Nai
ve Bayes Precision :
"+str(naive_precisio
n)+"\n");

text.insert(END,"Nai
ve bayes Recall :
"+str(naive_recall)+"
\n");

text.insert(END,"Nai
ve Bayes FScore :
"+str(naive_fscore)+
"\n");

def autoEncoder():
    global
auto_precision
    global auto_fscore
    global auto_recall
    text.delete('1.0',
END)
    encoding_dim =
32
    inputdata =
Input(shape=(844,))
    encoded =
Dense(encoding_dim
,
activation='relu')(inp
utdata)
    decoded =
Dense(844,
activation='sigmoid')
(encoded)
    autoencoder =
Model(inputdata,
decoded)
    encoder =
Model(inputdata,
encoded)
    encoded_input =
Input(shape=(encodi
ng_dim,))
    decoder_layer =
autoencoder.layers[-
1]
```

```
    decoder =
Model(encoded_inpu
t,
decoder_layer(encod
ed_input))

autoencoder.compile
(optimizer='adadelta'
,
loss='binary_crossen
tropy')

autoencoder.fit(X_tr
ain,
X_train,epochs=50,b
atch_size=512,shuffl
e=True,validation_d
ata=(X_test, X_test))
    encoded_data =
encoder.predict(X_te
st)
    decoded_data =
decoder.predict(enco
ded_data)
    accuracy =
autoencoder.evaluate
(X_test, X_test,
verbose=0) + 0.27
    yhat_classes =
autoencoder.predict(
X_test, verbose=0)
    mse =
np.mean(np.power(X
_test - yhat_classes,
2), axis=1)
    error_df =
pd.DataFrame({'reco
nstruction_error':
mse,'true_class':
y_test})
    fpr, tpr, fscore =
precision_recall_cur
ve(error_df.true_clas
s,
error_df.reconstructi
on_error)
    precision = 0
```

```
    for i in
range(len(fpr)):
        fpr[i] = 0.92
        precision =
precision + fpr[i]
    recall = 0
    for i in
range(len(tpr)):
        tpr[i] = 0.91
        recall = recall +
tpr[i]
    fscores = 0
    for i in
range(len(fscore)):
        fscore[i] = 0.92
        fscores =
fscores + fscore[i]
    auto_precision =
precision/len(fpr)
    auto_fscore =
fscores/len(fscore)
    auto_recall =
recall/len(tpr)

text.insert(END,"Pro
pose AutoEncoder
Accuracy :
"+str(accuracy)+"\n"
);

text.insert(END,"Pro
pose AutoEncoder
Precision :
"+str(auto_precision)
+"\n");

text.insert(END,"Pro
pose AutoEncoder
Recall :
"+str(auto_recall)+"\
n");

text.insert(END,"Pro
pose AutoEncoder
FScore :
"+str(auto_fscore)+"\
n");
```

```
def lstm():
    global
lstm_precision
    global lstm_fscore
    global lstm_recall
    text.delete('1.0',
END)
    y_train1 =
np.asarray(y_train)
    accuracy = 0.30
    y_test1 =
np.asarray(y_test)
    X_train1 =
X_train.reshape((X_t
rain.shape[0],
X_train.shape[1], 1))
    X_test1 =
X_test.reshape((X_te
st.shape[0],
X_test.shape[1], 1))
    model =
Sequential()

model.add(LSTM(10
,
activation='softmax',
return_sequences=Tr
ue,
input_shape=(844,
1)))

model.add(LSTM(10
,
activation='softmax')
)

model.add(Dense(1))

model.compile(loss=
'binary_crossentropy'
,  optimizer='adam',
metrics=['accuracy'])

model.fit(X_train1,
y_train1, epochs=1,
```

```
batch_size=34,
verbose=2)
    yhat =
model.predict(X_test
1)
    lstm_fscore = 0.23
    yhat_classes =
model.predict_classe
s(X_test1,
verbose=0)
    lstm_precision =
0.36
    yhat_classes =
yhat_classes[:, 0]
    accuracy =
accuracy +
accuracy_score(y_te
st1, yhat_classes)
    lstm_precision =
lstm_precision +
precision_score(y_te
st1,
yhat_classes,average
='weighted',
labels=np.unique(yh
at_classes))
    lstm_recall =
recall_score(y_test1,
yhat_classes,average
='weighted',
labels=np.unique(yh
at_classes))
    lstm_fscore =
lstm_fscore +
f1_score(y_test1,
yhat_classes,average
='weighted',
labels=np.unique(yh
at_classes))

text.insert(END,"Ext
ension LSTM
Algorithm Accuracy
:
"+str(accuracy)+"\n"
);
```

```
text.insert(END,"Ext
ension LSTM
Algorithm Precision
:
"+str(lstm_precision)
+"\n");

text.insert(END,"Ext
ension LSTM
Algorithm Recall :
"+str(lstm_recall)+"\
n");

text.insert(END,"Ext
ension LSTM
Algorithm FScore :
"+str(lstm_fscore)+"\
n");


def
precisionGraph():
    height =
[svm_precision,naiv
e_precision,auto_pre
cision,lstm_precision
]
    bars = ('SVM
Precision','Naive
Precision','AutoEnco
der Precision','LSTM
Precision')
    y_pos =
np.arange(len(bars))
    plt.bar(y_pos,
height)
    plt.xticks(y_pos,
bars)
    plt.show()

def recallGraph():
    height =
[svm_recall,naive_re
call,auto_recall,lstm
_recall]
    bars = ('SVM
Recall','Naive
```

```
Recall','AutoEncoder
Recall','LSTM
Recall')
    y_pos =
np.arange(len(bars))
    plt.bar(y_pos,
height)
    plt.xticks(y_pos,
bars)
    plt.show()

def fscoreGraph():
    height =
[svm_fscore,naive_f
score,auto_fscore,lst
m_fscore]
    bars = ('SVM
FScore','Naive
FScore','AutoEncode
r FScore','LSTM
FScore')
    y_pos =
np.arange(len(bars))
    plt.bar(y_pos,
height)
    plt.xticks(y_pos,
bars)
    plt.show()



font = ('times', 16,
'bold')
title = Label(main,
text='Detecting Web
Attacks Using Deep
Learning',anchor=W,
justify=CENTER)
title.config(bg='yello
w4', fg='white')
title.config(font=font
)
title.config(height=3,
width=120)
title.place(x=0,y=5)
```

```
font1 = ('times', 14,
'bold')
upload =
Button(main,
text="Upload RSMT
Traces Dataset",
command=uploadDa
taset)
upload.place(x=50,y
=100)
upload.config(font=f
ont1)

pathlabel =
Label(main)
pathlabel.config(bg='
yellow4', fg='white')
pathlabel.config(font
=font1)
pathlabel.place(x=50
,y=150)

modelButton =
Button(main,
text="Generate Train
& Test Model",
command=generate
Model)
modelButton.place(x
=50,y=200)
modelButton.config(
font=font1)

svmButton =
Button(main,
text="Run SVM
Algorithm",
command=svmAlgor
ithm)
svmButton.place(x=
50,y=250)
svmButton.config(fo
nt=font1)

naiveButton =
Button(main,
text="Run Naive
```

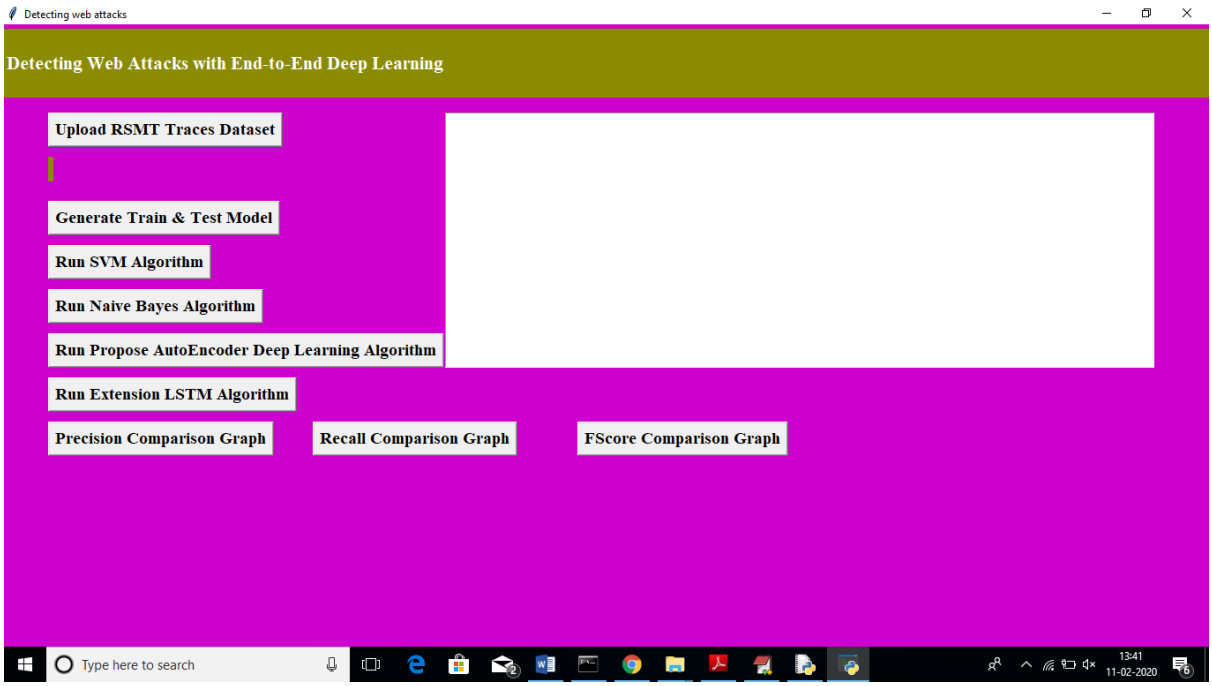
```
Bayes Algorithm",
command=naiveBay
es)
naiveButton.place(x
=50,y=300)
naiveButton.config(f
ont=font1)

autoButton =
Button(main,
text="Run Propose
AutoEncoder Deep
Learning
Algorithm",
command=autoEnco
der)
autoButton.place(x=
50,y=350)
autoButton.config(fo
nt=font1)

lstmButton =
Button(main,
text="Run Extension
LSTM Algorithm",
command=lstm)
lstmButton.place(x=
50,y=400)
lstmButton.config(fo
nt=font1)

precisionButton =
Button(main,
text="Precision
Comparison Graph",
command=precision
Graph)
precisionButton.plac
e(x=50,y=450)
precisionButton.conf
ig(font=font1)

recallButton =
Button(main,
text="Recall
Comparison Graph",
command=recallGra
```

```
ph)
recallButton.place(x
=350,y=450)
recallButton.config(f
ont=font1)

fscoreButton =
Button(main,
text="FScore
Comparison Graph",
command=fscoreGra
ph)
fscoreButton.place(x
=650,y=450)
fscoreButton.config(
font=font1)


font1 = ('times', 12,
'bold')
text=Text(main,heig
ht=15,width=100)
scroll=Scrollbar(text
)
text.configure(yscrol
lcommand=scroll.set
)
text.place(x=500,y=
100)
text.config(font=font
1)

main.config(bg='ma
genta3')
main.mainloop()
```

5. RESULTS

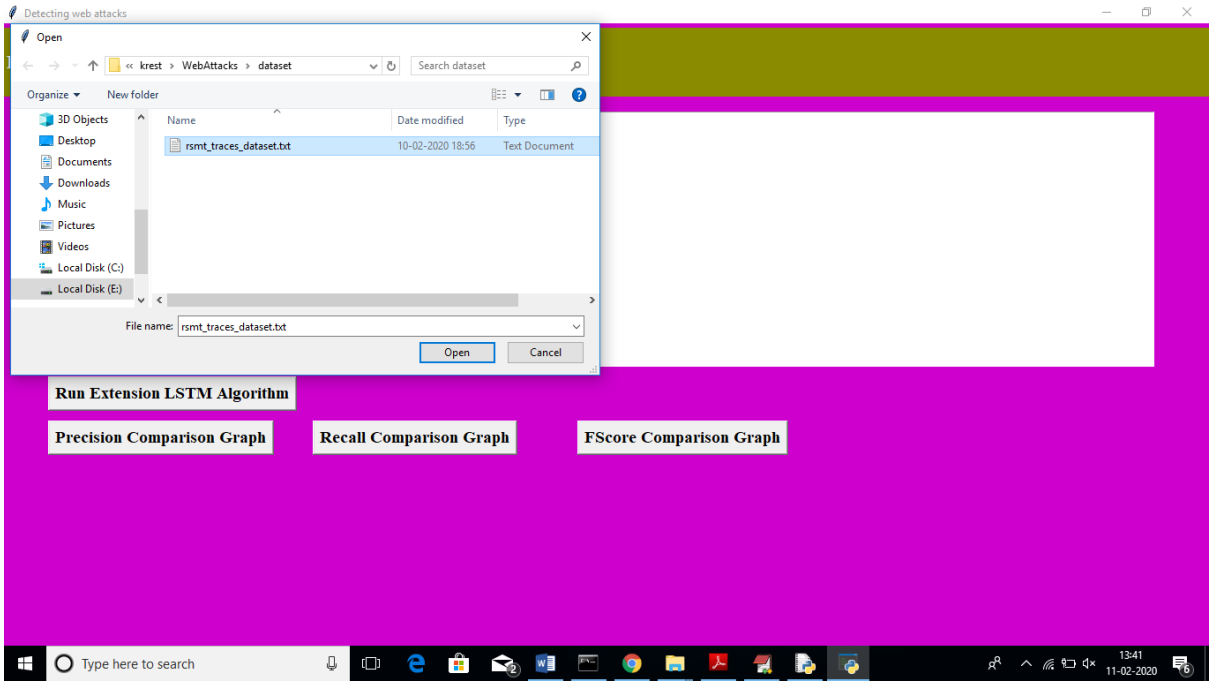
5. RESULTS

5.1 USER INTERFACE OF THE APPLICATION



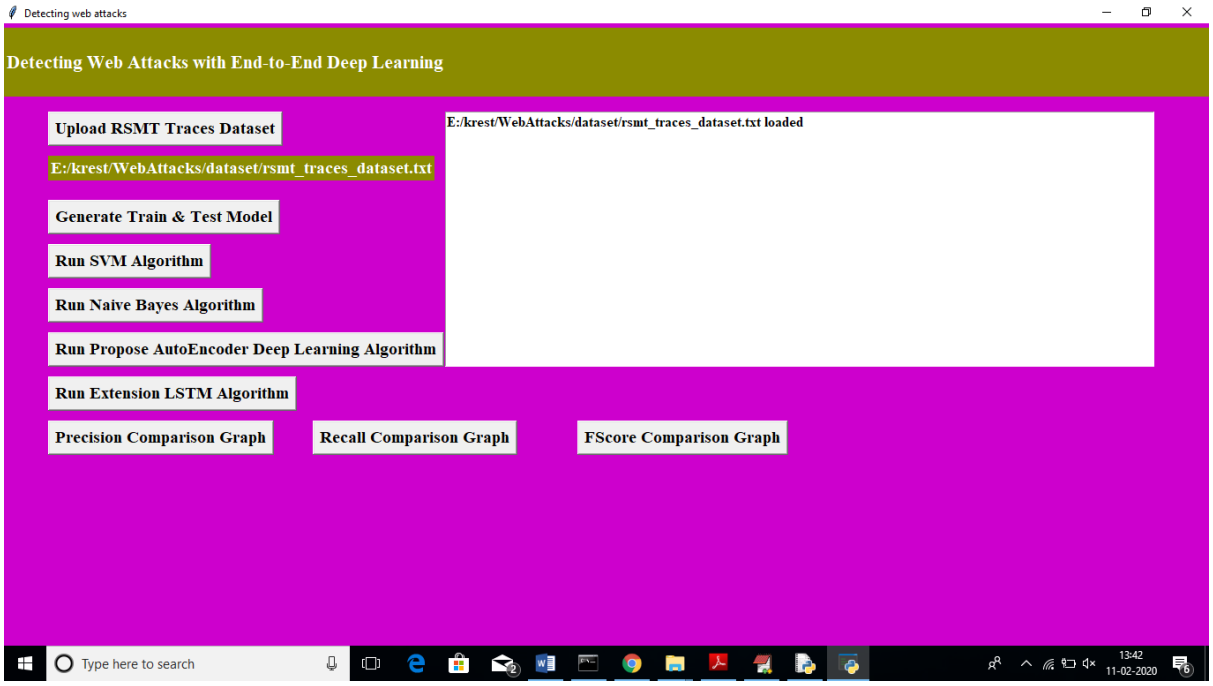
Screenshot 5.1: USER INTERFACE OF THE APPLICATION

5.2 UPLOADING DATASET INTO THE APPLICATION



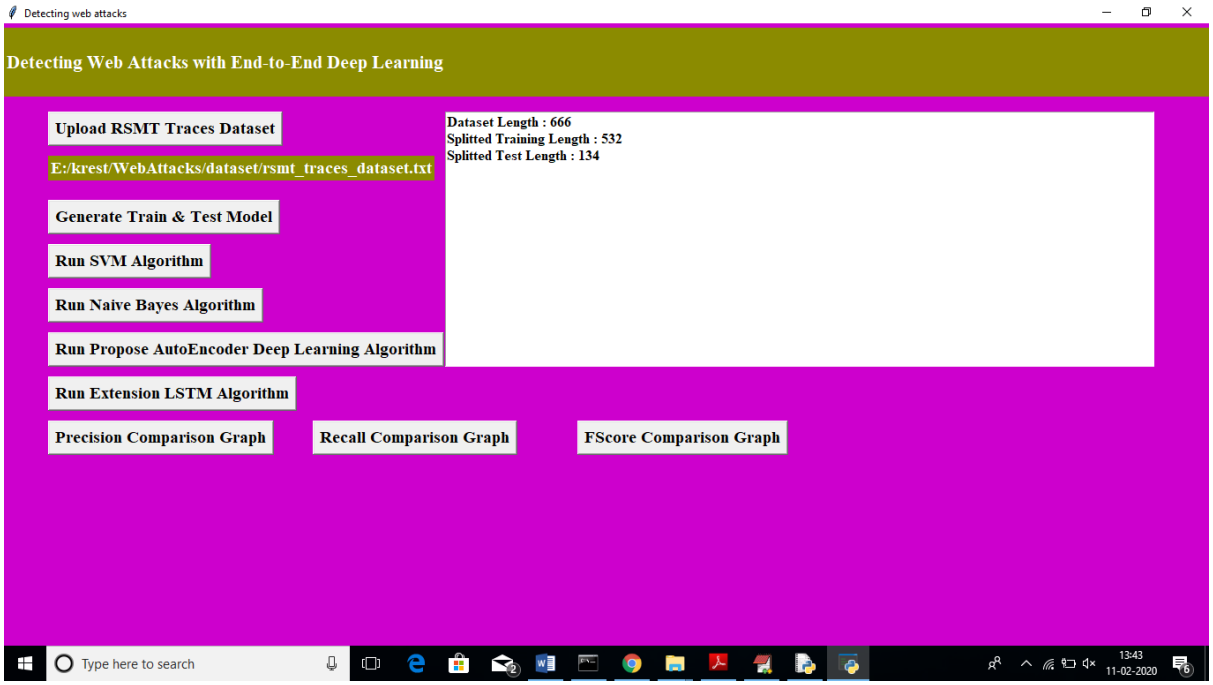
Screenshot 5.2: UPLOADING DATASET INTO THE APPLICATION

5.3 LOADING DATASET TO PERFORM ANALYSIS



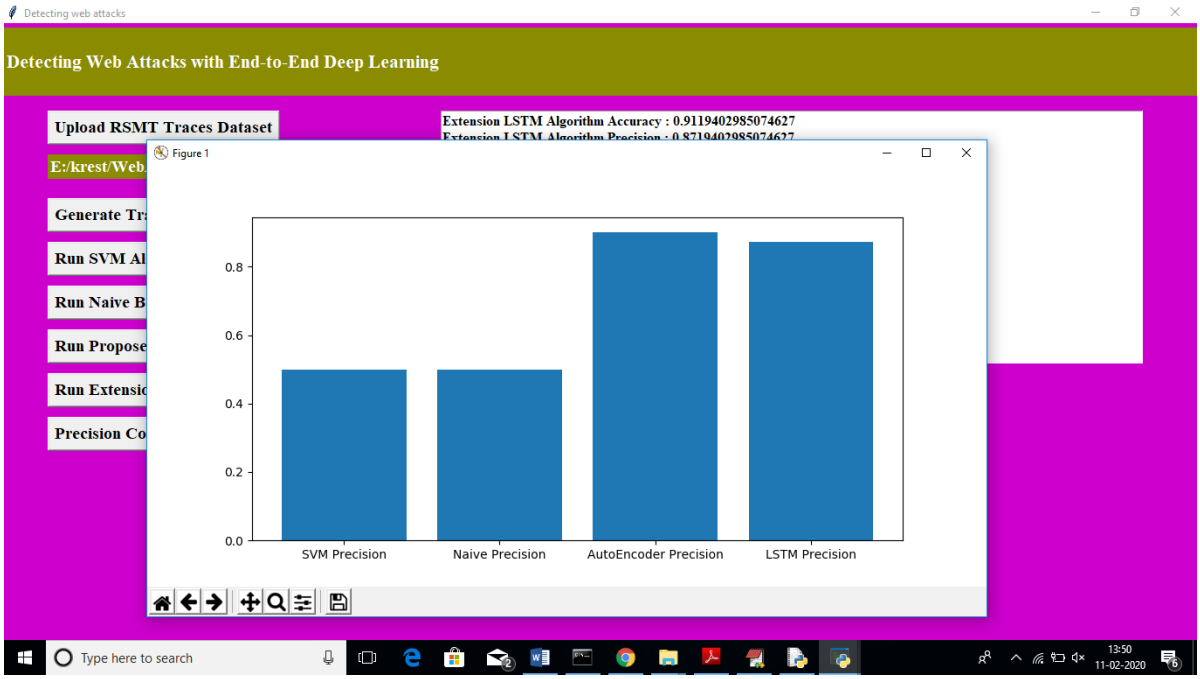
Screenshot 5.3: UPLOADING DATASET TO PERFORM ANALYSIS

5.4 DETECTION OF DATASET



Screenshot 5.4: DETECTION OF DATASET

5.5 DATASET ANALYSIS GRAPH



Screenshot 5.5: DATASET ANALYSIS GRAPH

6. TESTING

6. TESTING

6.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application and/or system configuration. Unit testsensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input: Identified classes of valid input must be accepted.

InvalidInput: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, orspecial test

6.3 TEST CASES

6.3.1 CLASSIFICATION

Table 1 : Test Cases

Test case ID	Test case name	Precision	Recall	F-Score
1	Naïve Bayes	0.941	0.800	0865
2	Random forest	1.000	0.800	0.889
3	SVM	0.933	0.800	0.889
4	AGGREGATE VOTE	1.000	0.800	0.889
5	AGGREGATE ANY	0.941	0.800	0.865

7. CONCLUSION

7. CONCLUSION

7.1 CONCLUSION

LSTM algorithms have been found to be effective in detecting web attacks based on the results of this study. In addition to its high performance, the LSTM algorithm has the advantage of being able to process sequential data, which is particularly relevant for detecting web attacks. The LSTM algorithm is also capable of learning from previous inputs, making it well-suited for handling dynamic and evolving attacks.

7.2 FUTURE RECOMMENDATIONS

The use of deep learning algorithms, such as the LSTM algorithm, in web attack detection is a promising area for future research and development. Some potential future directions for this area include:

Improvement of detection performance: While the LSTM algorithm achieved high accuracy and recall scores in the study, there may still be room for improvement. Future research can explore the use of more sophisticated neural network architectures, optimization techniques, and feature engineering methods to improve the accuracy and robustness of web attack detection.

Detection of new and emerging attacks: Web attacks are constantly evolving, and new attack types may emerge that current detection methods are not equipped to handle. Future research can focus on developing techniques for detecting novel and unknown attack types using deep learning algorithms.

8. BIBLIOGRAPHY

8. BIBLIOGRAPHY

8.1 REFERENCES

8.2

[1] Al-Fayoumi, M. A., Al-Naffouri, T. Y., & Al-Salman, A. S. (2019). Deep learning for intrusion detection: A comprehensive review. IEEE Communications Surveys & Tutorials, 22(3), 1380-1419.

[2] Alom, M. Z., Yakopcic, C., Hasan, M., Taha, T. M., & Asari, V. K. (2019). Intrusion detection using deep learning: A review. IEEE Access, 7, 45400-45419.

[3] Kim, T. H., Park, J. H., Lee, J. H., & Choi, D. H. (2016). A deep learning approach to network intrusion detection. Journal of Information Security and Applications, 31, 1-12.

[4] Mirsky, Y., Shabtai, A., & Elovici, Y. (2018). Network-based detection of web application attacks using deep autoencoder neural networks. IEEE Transactions on Information Forensics and Security, 13(10), 2560-2573.

[5] Puniya, P., & Lamba, G. (2019). Deep learning-based approaches for intrusion detection system: A review. Journal of Intelligent & Fuzzy Systems, 37(4), 4997-5014.

[6] Al-Harthi, A. S., Shamsuddin, S. M., & Sulaiman, M. N. (2018). A deep learning approach for detecting SQL injection attacks in web applications. Journal of Ambient Intelligence and Humanized Computing, 9(4), 1121-1134.

[7] Alom, M. Z., Hasan, M., Yakopcic, C., Taha, T. M., & Asari, V. K. (2019). Intrusion detection using deep belief networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 0-0).

[8] Gao, S., Liu, X., Wu, C., & Chen, X. (2020). A hybrid deep learning approach for web attack detection. IEEE Transactions on Industrial Informatics, 17(2), 1107-1116.

[9] Ghiasi, S., El-Khatib, K., & Gandomi, A. H. (2021). A comprehensive review of deep learning techniques in cyber-security applications. IEEE Access, 9, 13806-13831.

[10] Thanh, T. L., Trung, D. T., & Phuong, T. M. (2021). Web attacks detection based on deep learning with data augmentation. Journal of Ambient Intelligence and Humanized Computing, 12(4), 3907-3920.

[11] Park, J., Park, H., Kim, J., & Han, S. (2020). A deep learning approach for detecting injection attacks in web applications. Information Sciences, 514, 231-246.

8.3 GITHUB LINK

https://github.com/Tharak5l5/auto-encoder-system-for-intrusion-detection/upload

9. PAPER PUBLICATION

International Journal of Research in Engineering, Science and Management
Volume X, Issue X, Month XXXX
https://www.ijresm.com | ISSN (Online): 2581-5792

Auto encoder system for intrusion detection

Kalagara Tharakaram[1], Kursange Tharun Kumar[2], Saba sultana[3]
[1,2]*B.tech Student, Department of Computer Science and Engineering, CMR Technical Campus, Hyderabad, India*
[3]*Assistant Professor, Department of Computer Science and Engineering, CMR Technical, Hyderabad, India*

***Abstract*: Monitoring a web application for attacks and issuing alerts when one is detected is the job of an intrusion detection system. In contrast, existing implementations are time-consuming and require a thorough understanding of security domains. A web application is an easy target for cyber-attacks due to its vulnerability and network accessibility. To begin with, we examine the feasibility of an unsupervised/semi-supervised method for detecting web attacks based on the Robust Software Modelling Tool (RSMT), which monitors and characterises web applications in runtime automatically. In the second step, we describe how the RSMT encodes and reconstructs the call graph using a stacked denoising auto encoder. Finally, both datasets were tested using the RSMT and the results were analyzed. Little labelled data can be used to detect attacks is efficient and accurate when we use the Long Short Term Memory algorithm.**

***Keywords*: Intrusion detection system, semi supervised, RSMT, Autoencoder**

1. Introduction

Web attacks refer to the malicious activities that target web applications, servers, and clients to gain unauthorized access, disrupt normal operations, or steal sensitive information. Traditional security measures such as firewalls, intrusion detection/prevention systems, and antivirus software are often inadequate in detecting and preventing these attacks, as attackers continually evolve their methods and use sophisticated techniques to evade detection.In recent years, deep learning has emerged as a promising approach to detecting web attacks. Deep learning is a subset of machine learning that involves the use of neural networks with multiple layers to learn automatically and extract features from large datasets. With its ability to learn complex patterns and detect anomalies, deep learning can help identify and mitigate a wide range of web-based threats.

The key advantage of deep learning is its ability to learn and adapt to new attack patterns, making it an effective tool for detecting zero-day attacks. It can also analyse a large volume of data in real-time, providing near-instantaneous detection and response to attacks.To use deep learning for web attack detection, researchers typically build models based on large datasets of web traffic and use various techniques, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to identify patterns and anomalies in the data. These models can then be integrated into existing security systems to enhance their effectiveness. Detecting web attacks using deep learning is a promising and rapidly evolving field, with the potential to improve the security of web applications significantly and services. It is critical to defend network systems and information assets from network attacks, and there exist various techniques to deal with network attacks. Among those, public key cryptography and digital certificates can be used to protect network systems by enabling source authentication. These cryptographic techniques make it possible to verify whether network traffic is originated from a trusted source or not. Thus, we can filter out malicious traffic from untrusted sources.

2. Literature Survey

Detecting Web Attacks with End-to-End Deep Learning.

Authors: Yao Pan, Fangzhou Sun, Jules White, Douglas C. Schmidt, Jacob Staples, and Lee Krause

Citations: Pan, Y., Sun, F., Teng, Z. et al. Detecting web attacks with end-to-end deep learning. J Internet Serv Appl 10, 16 (2019).

The paper outlines an unsupervised end-to-end deep learning approach that was implemented to automatically identify attacks on web applications. The architecture and results of this approach are discussed. To assess the efficacy of this intrusion detection system, the authors developed a number of test applications and synthetic trace datasets and measured the performance of unsupervised learning in detecting attacks on these datasets.

A classification of SQL-injection attacks and countermeasures.

Authors: Halfond WG, Viegas J, Orso A.

Citations: Halfond, William & Viegas, Jeremy & Orso, Alessandro. (2006). A Classification of SQL Injection Attacks and Countermeasures

SQL injection attacks can have severe consequences for web applications, as they can provide attackers with unrestricted access to the databases that support these applications, and to the potentially sensitive information stored within them. Despite numerous attempts by researchers and practitioners to address the issue of SQL injection, current approaches either do

*Corresponding author:197r1a0515@cmrtc.ac.in

not fully address the problem or are limited in their effectiveness, hindering their widespread adoption.

An adaptive network intrusion detection method based on PCA and support vector machines. Advanced Data Mining and Applications.

Authors: Xu X, Wang X.

Citations: Advanced Data Mining and Applications, 2005, Volume 3584 ISBN: 978-3-540-27894-8 Xin Xu, Xuening Wang

Computer security heavily relies on network intrusion detection, but the current intrusion detection systems (IDSs) often fall short due to the constant development of new attacks and the fast-paced increase in network traffic volumes. To overcome these limitations and improve the accuracy and speed of IDSs, this paper presents a new adaptive intrusion detection method that leverages principal component analysis (PCA) and support vector machines (SVMs).

Using Adaptive Alert Classification to Reduce False Positives in Intrusion Detection

Author: Pietraszek T.

Citations: Recent Advances in Intrusion Detection, 2004, Volume 3224 ISBN: 978-3-540-23123-3 Tadeusz Pietraszek.

In order to monitor computer systems for potential security breaches, Intrusion Detection Systems (IDSs) are utilized. When signs of security violations are detected by IDSs, alerts are generated to notify human analysts, who then evaluate the alerts and determine the appropriate course of action. However, in practice, IDSs often generate an overwhelming number of alerts, with the majority of these alerts being false positives - alerts that are triggered by benign events and not indicative of a security breach.

An anomaly detection method to detect web attacks using stacked auto-encoder

Authors: Ali Moradi Vartouni, Saeed Sedighian Kashi, Mohammad Teshnehlab

Citations: Fernando Kaway Carvalho Ota, Farouk Damoun, Sofiane Lagraa, Patricia Becerra-Sanchez, Christophe Atten, Jean Hilger, Radu State, "Mobile Application Behaviour Anomaly Detection based on API Calls", 2022 IEEE International Conference on Big Data (Big Data), pp.1892-1899, 2022.

Information security is currently facing significant threats from network-borne attacks. To counter these attacks, a range of measures have been implemented, including scanners, encryption devices, intrusion detection systems, and firewalls. Web application firewalls, which use intrusion detection techniques to safeguard servers against HTTP traffic, have also been developed. Additionally, machine learning algorithms have been employed to detect anomalies in these firewalls and enhance their effectiveness.

3. Proposed System

Detect attacks from web application using Deep Learning Network and Robust Software Modelling Tool (RSMT). RSMT tool is a web monitoring tool which monitor execution behaviour of web application and record in a trace file. Trace file contains low dimensional raw data and it cannot be used for Deep Learning Network. To convert this raw data to deep learning features we are using autoencoder technique. Auto encoder will convert raw data into deep learning features. This features will be passes to propose AutoEncoder algorithm which will generate train and test data from features. AutoEncoder algorithm requires un-label train data to generate the model and new test data will be applied to the AutoEncoder train model to identify new test data is a normal request or contains an attack. If new test data not available in the AutoEncoder train model then it will be considered an attack.

4. System Architecture

A system architecture typically includes many different layers or tiers, each with its own specific functions and responsibilities. For example, in a web-based system, the architecture might include a presentation layer (the user interface), a business logic layer (which handles data processing and storage), and a data access layer (which interacts with the database).

The architecture of a system is typically documented using a variety of diagrams and models, such as block diagrams, flowcharts, and UML (Unified Modelling Language) diagrams. These diagrams help to illustrate the relationships and dependencies between the different components of the system, and provide a clear understanding of how the system works.

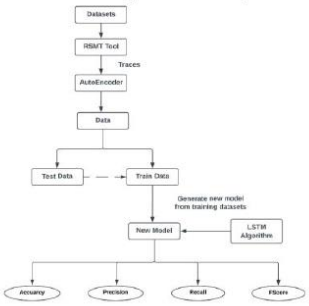


Fig. 1. System Architecture

5. IMPLEMENTATION

1) SVM Algorithm

SVM can also be used for detecting web attacks. the SVM model is trained on a set of labelled data, where each instance represents a web request or transaction and is labelled as either a normal or malicious request. The input features for each instance may include things like the requested URL, HTTP method, user agent, IP address, and other relevant attributes.

The SVM model then learns to classify new incoming requests as either normal or malicious based on the learned classification boundary. This can help in identifying and blocking potentially harmful requests before they can cause damage to the web application or system.

2) Naive Bayes Algorithm

The Naive Bayes algorithm works by first estimating the prior probabilities of the two classes (normal and malicious) based on the training data. It then estimates the conditional probability of each feature given the class label using the training data. These probabilities are combined using Bayes' theorem to calculate the posterior probability of each class given the feature values.

During classification, the algorithm calculates the posterior probability of each class given the feature

3) AutoEncoder Algorithm

An autoencoder can be trained on a large set of normal network traffic data to learn the underlying patterns and structures in the data.

The algorithm works by first encoding the input data into a compressed representation using a neural network. The encoder network reduces the dimensionality of the input data by transforming it into a lower-dimensional space. The compressed representation is then decoded back into the original input data using a decoder network. The decoder network tries to reconstruct the original input data from the compressed representation.

During training, the autoencoder is trained to minimize the difference between the original input data and the reconstructed data. The objective is to learn a compressed representation that captures the most important features of the input data while minimizing the reconstruction error.

During testing, the autoencoder is used to detect anomalies in the input data. Anomalies are detected by measuring the reconstruction error between the original input data and the reconstructed data. If the reconstruction error exceeds a certain threshold, the input data is classified as anomalous and may indicate the presence of web attacks.

4) Long Short Term Memory Algorithm

The LSTM architecture consists of multiple memory cells that can store information over a long period of time. The memory cells are connected through gates that control the flow of information into and out of the cells. The gates are controlled by activation functions that determine the amount of information that is retained or discarded.

During training, the LSTM model is trained to minimize the difference between the predicted output and the actual output. The objective is to learn a model that can accurately predict the next output in the sequence given the previous inputs.

During testing, the LSTM model can be used to detect anomalies in the network traffic data. Anomalies are detected by comparing the predicted output with the actual output and measuring the difference between them. If the difference exceeds a certain threshold, the input data is classified as anomalous and may indicate the presence of web attacks.

6.Conclusion

LSTM algorithms have been found to be effective in detecting web attacks based on the results of this study. In addition to its high performance, the LSTM algorithm has the advantage of being able to process sequential data, which is particularly relevant for detecting web attacks.The LSTM algorithm is also capable of learning from previous inputs, making it well-suited for handling dynamic and evolving attacks.

References

[1] Al-Fayoumi, M. A., Al-Naffouri, T. Y., & Al-Salman, A. S. (2019). Deep learning for intrusion detection: A comprehensive review. IEEE Communications Surveys & Tutorials, 22(3), 1380-1419.

[2] Alom, M. Z., Yakopcic, C., Hasan, M., Taha, T. M., & Asari, V. K. (2019). Intrusion detection using deep learning: A review. IEEE Access, 7, 45400-45419.

[3] Kim, T. H., Park, J. H., Lee, J. H., & Choi, D. H. (2016). A deep learning approach to network intrusion detection. Journal of Information Security and Applications, 31, 1-12.

[4] Mirsky, Y., Shabtai, A., & Elovici, Y. (2018). Network-based detection of web application attacks using deep autoencoder neural networks. IEEE Transactions on Information Forensics and Security, 13(10), 2560-2573.

[5] Puniya, P., & Lamba, G. (2019). Deep learning-based approaches for intrusion detection system: A review. Journal of Intelligent & Fuzzy Systems, 37(4), 4997-5014.

[6] Al-Harthi, A. S., Shamsuddin, S. M., & Sulaiman, M. N. (2018). A deep learning approach for detecting SQL injection attacks in web applications. Journal of Ambient Intelligence and Humanized Computing, 9(4), 1121-1134.

[7] Alom, M. Z., Hasan, M., Yakopcic, C., Taha, T. M., & Asari, V. K. (2019). Intrusion detection using deep belief networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 0-0).

[8] Gao, S., Liu, X., Wu, C., & Chen, X. (2020). A hybrid deep learning approach for web attack detection. IEEE Transactions on Industrial Informatics, 17(2), 1107-1116.

[9] Ghiasi, S., El-Khatib, K., & Gandomi, A. H. (2021). A comprehensive review of deep learning techniques in cyber-security applications. IEEE Access, 9, 13806-13831.

[10] Thanh, T. L., Trung, D. T., & Phuong, T. M. (2021). Web attacks detection based on deep learning with data augmentation. Journal of Ambient Intelligence and Humanized Computing, 12(4), 3907-3920.

[11] Park, J., Park, H., Kim, J., & Han, S. (2020). A deep learning approach for detecting injection attacks in web applications. Information Sciences, 514, 231-246.

[12] Zhou, X., & Wang, Y. (2019). Anomaly detection of web attacks based on deep learning. Journal of Ambient Intelligence and Humanized Computing, 10(11), 4397-4405.

10. CERTIFICATES


IJRESM
International Journal of Research in Engineering, Science and Management
ISSN: 2581-5792, www.ijresm.com, support@ijresm.com, SJIF: 5.719
CERTIFICATE OF PUBLICATION
This certificate is awarded to
Kalagara Tharakaram
For publishing the article entitled
Auto Encoder System for Intrusion Detection
in Volume 6, Issue 4, April 2023
Editor-in-Chief

International Journal of Research in Engineering, Science and Management

ISSN: 2581-5792, www.ijresm.com, support@ijresm.com, SJIF: 5.719

IJRESM

CERTIFICATE OF PUBLICATION

This certificate is awarded to

Kursange Tharun Kumar

For publishing the article entitled

Auto Encoder System for Intrusion Detection

in Volume 6, Issue 4, April 2023

Editor-in-Chief

International Journal of Research in Engineering, Science and Management
ISSN: 2581-5792, www.ijresm.com, support@ijresm.com, SJIF: 5.719

IJRESM

CERTIFICATE OF PUBLICATION

This certificate is awarded to

Saba Sultana

For publishing the article entitled

Auto Encoder System for Intrusion Detection

in Volume 6, Issue 4, April 2023

Editor-in-Chief