Deep Learning for Botnet Attack Detection

Dr.R.Prema
AP/CSE
SCSVMV University
Kanchipuram

Settemoni Manohar
B.E(CSE)
SCSVMV University
Kanchipuram

Sai Ganesh
B.E(CSE)
SCSVMV University
Kanchipuram

ABSTRACT:
An effective technique for identifying botnet attacks is deep learning (DL). Yet, the amount of memory space and network traffic that is needed is typically substantial. Therefore, it is extremely difficult to implement the DL approach in IoT devices with little memory. Using dimensionality reduction approaches, we lower the feature dimensionality of large-scale IoT network traffic data. The most pertinent publicly accessible dataset for network-based botnet attack detection in IoT networks is the Bot-IoT dataset. With Bot-IoT, millions of samples of botnet attack traffic were included. The four IoT botnet scenarios that these attack traffic samples fall under are DDoS, DoS, reconnaissance, and information theft. The most common method for reducing the dimensionality of a feature set is to apply a linear or non-linear transformation approach.

Keywords: Botnet, Decision Trees, Dimensionality Reduction, Feature Extraction, Protocols,

INTRODUCTION:

A recently developed dataset is utilised to pick features effectively and identify Bot-IoT assaults accurately in an IoT network environment. The dataset includes information on botnet assaults, the Internet of Things, regular traffic flows, and many different cyberattacks. The production of this dataset with effective information characteristics uses a realistic test environment to track correct traffic and produce an effective dataset. Likewise, more features were extracted and added to the extracted feature set in order to increase the performance of machine learning models and create more accurate prediction models.However, the retrieved features are labelled to produce superior performance outcomes, such as attack flow, categories, and subcategories. Today's Internet of Things (IoT) technology is expanding rapidly, and every minute, a large number of devices are connecting to it. Using this technology makes daily living easier and more structured. IoT technology, for instance, was once only used in homes and small offices, but it is now integrated into other industries for increased reliability and time savings. IoT, however, is quickly taking over our daily lives and becoming indispensable.More than 27 million IoT devices will link in 2021 as IoT technology matures, ushering in a sea change for the industry. An rising number of cyberattacks are focusing on Internet of Things (IoT) devices as a result of their quick growth and widespread adoption. According to certain reports, botnet-based attacks make up the majority of attacks in IoT contexts. IoT devices still have many security flaws since the majority of them lack the memory and computing power necessary for strong security solutions. Additionally, many existing rule-based detection systems can be evaded by attackers. In this paper, we suggested a sequential detection architecture, machine learning (ML)-based botnet attack detection system. Using a lightweight feature selection strategy is effective.A botnet is a collection of multiple bots that have been programmed to engage in harmful activity on the target network and are managed by a single unit known as the botmaster using command and control protocol. Bots are compromised computers that are remotely managed by the botmaster without showing any signs of hacking and are used to carry out destructive actions. From small botnets with a few hundred bots to massive botnets with 50,000 hosts, botnet size varies. Hackers can distribute botnet software and carry out their operations covertly for years without leaving any visible signs of their existence.Network intrusion detection systems have been created using Machine Learning (ML) techniques to protect connected IoT devices from sophisticated botnet attacks (NIDS). At key nodes in an IoT network, such NIDS can be deployed.A unique capability for the automatic extraction of features from massively scaled, high-speed network data created by interconnected heterogeneous IoT devices is provided by Deep Learning (DL), a cutting-edge machine learning approach. IoT devices have a limited amount of processing power and memory, making it impossible to identify botnets using typical computer network NIDS approaches.To create an efficient DL technique for botnet detection in IoT networks, large enough network traffic data must be available to offer good classification performance. Yet, processing and analysing highly dimensional network traffic data may lead to the dimensionality curse. Moreover, training DL models with such high-dimensional data can lead to the Hughes phenomena. It is challenging and requires a lot of processing power and storage space to process high-dimensional data. The enormous volumes of network traffic data required for DL cannot be stored on IoT devices due to memory constraints.In order to reliably identify complicated and recent botnet attacks based on low-dimensional network traffic data and while also decreasing the high dimensionality of huge network traffic features, an end-to-end DL-based botnet detection approach is needed. The Bot-IoT dataset is currently the most pertinent publicly available dataset for botnet attack detection in IoT networks because it contains IoT network traffic samples, complete network information, a variety of complex IoT botnet attack scenarios, accurate ground truth labels, and a sizable amount of labelled data required for efficient supervised DL.The initial feature dimensionality1 of the Bot-IoT dataset is 43, and

1.085 GB of memory is required to store this network traffic information. To date, all feature dimensionality reduction methods used on the Bot-IoT dataset were based on feature selection methods.

LITERATURE SURVEY:

[1] K. Anoh et al., "Virtual microgrids: A management model for peer-to-peer energy trading," in Proceedings of the Second International Conference on Future Network Distribution Systems, pp. 1–5, 2018.

The widespread use of smart technologies and distributed energy resources (DERs) has made it possible to incorporate microgrid generation into the energy supply chain. In order to reduce operating costs, this article suggests using energy trading agents (ETA) in a neighbourhood area network (NAN) where a number of microgrids (MGs) are joined together into logical clusters known as virtual MGs (VMGs). Each VMG is given an ETA so that prosumers in a VMG exchange can divorce the communication network from the grid topology and study the communication performance.

[2] Y. Hongxu, A. O. Akmandor, and N. K. Jha, "Smart safe but energy-efficient Internet-of-Things sensors," IEEE Trans. Multi-Scale Comput. Syst., vol. 4, no. 4, Oct./Dec. 2018, pp. 914–930.
Every year, zettabytes of sensitive data are produced due to the growth of the Internet of Things (IoT). The generated data are frequently unprocessed, necessitating cloud resources for processing and decision-making processes to extract useful information (i.e., distil smartness). Using cloud resources brings up important design concerns, like low bandwidth, insufficient energy, and security risks. To circumvent these issues, edge-side computing and cryptographic methods have been suggested. It is challenging to attain smartness, security, and energy efficiency at the same time due to increased computing load and energy consumption.

[3] Lightweight privacy-preserving Q-learning based energy management for the IoT-enabled smart grid, IEEE Internet Things J., vol. 7, no. 5, pp. 3935-3947, May 2020.
The smart grid implements exceptionally low energy dissipation for the functioning of the smart city as the largest Internet-of-Things (IoT) deployment in the world. Yet, the smart grid's electrical data contains a tonne of sensitive data, including dispatching instructions and bills. For the purpose of developing Q-learning-based energy strategies, the data are always disclosed to cloud servers in plaintext, which presents an opportunity for the adversary to misuse the user data. Hence, in this paper, we suggest a simple Q-learning architecture that protects privacy (LiPSG)

[4] S. Sankaranarayanan, J. J. Rodrigues, R. J. Tom "With an IoT-based power distribution system, customers and utilities can reduce demand for energy by using smart energy management.system "IEEE Internet Things J., Oct. 2019, vol. 6, no. 5, pp. 7386-7394.

Electricity companies are switching from nonrenewable to renewable energy as a result of the rising global demand for energy and the expanding carbon footprint. The electric system is incorporating more renewable energy every day. Energy use by consumers must be properly and wisely managed. All homes and appliances are now online thanks to the Internet of Things. Smart homes make it feasible to examine consumer usage trends and energy consumption. The demand for during the busiest times of the day

[5] L. Chettri and R. Bera, "A thorough review on Internet of Things (IoT) towards 5G wireless systems," IEEE Internet Things J., vol. 7, no. 1, Jan. 2020, pp. 16–32.

Wireless technology has recently experienced rapid global growth. Fifth-generation (5G) technology has emerged as one of the most difficult and intriguing research areas in wireless technology. An overview of the Internet of Things (IoT) in 5G wireless technologies is given in this article. IoT in the 5G network will transform the game in the next years. It will pave the way for innovative wireless architecture and intelligent services.In order to fulfil the needs of multiple device connectivity, high data rates, increased bandwidth, low-latency quality of service (QoS), and minimal interference, recent cellular network LTE (4G) will not be sufficient and efficient. We believe that 5G is the most promising technology to address these issues.

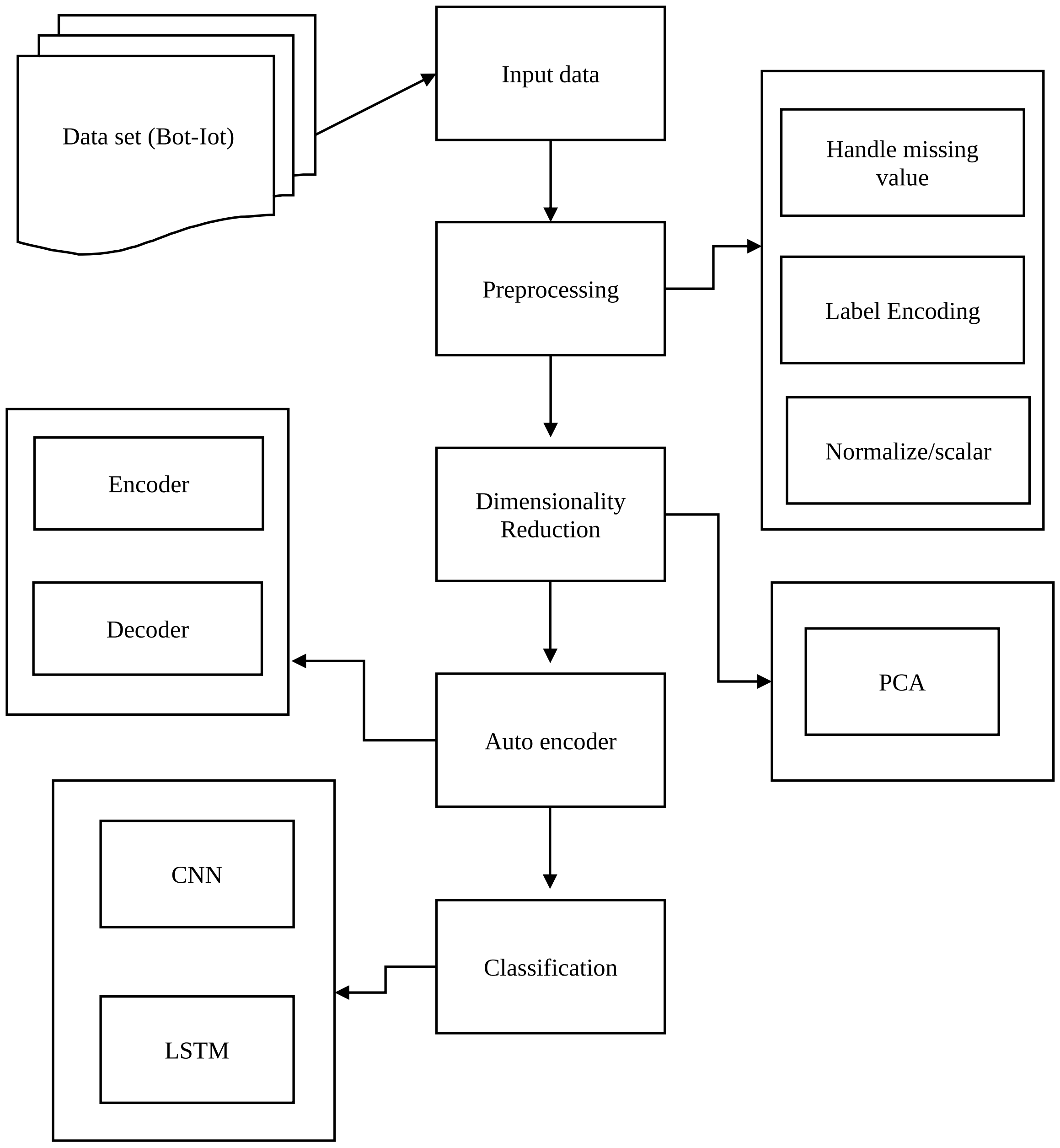
PROBLEM STATEMENT:

Tracking and analysing the attacks themselves, into which typical security solutions provide visibility, and identifying which assaults were launched from botnets are more often used methods for identifying botnets. Many other cyber-based attacks, including overloading targets' networks and devices with traffic and stealing data from hosts infected with the bots, can be carried out via botnets, which can have thousands of host.

PROPOSED SYSTEM:

In this system, the Bot-Iot dataset was taken as input from the dataset repository. Then, we have to implement the data preprocessing step. In this step, we have to handle the missing values for avoid wrong prediction, to encode the label for input data and normalize/ scaling the input data. Then, we have to implement the feature dimensionality reduction such as Principal Component Analysis (PCA). we have to implement the deep learning algorithms such as Long Short term Memory (LSTM) and Convolutional Neural Network (CNN). Finally, the experimental results shows that the performance metrics such as accuracy, precision, recall and confusion matrix.

SYSTEM ARCHITECTURE:



MODULE DESCRIPTION:

DATA SELECTION

The source of the input data was a dataset repository. The Bot-IoT dataset is used in our procedure. Detecting malicious traffic is done through the data selection method. The dataset includes information about botnet attacks, regular traffic flows, the Internet of Things, and countless more cyberattacks. However, the recovered features are tagged for better performance results, such as attack flow, categories, and subcategories.

PRE-PROCESSING

Pre-processing involves removing extraneous data from the dataset. Pre-processing data transformation techniques are used to turn the dataset into a machine learning-friendly structure. Subtract missing data: In this process, the null values, such as missing values and Nan values, are changed to 0. categorising data for encoding Category data refers to variables with a constrained range of label values.

DATA NORMALISATION

Handling Missing values

DIMENSIONLITY REDUCTION

The term "dimensionality reduction technique" refers to a method of reducing the number of dimensions in a dataset while maintaining the information content. We must utilise PCA (principal component analysis) in this stage. Visualisation, noise filtering, and feature extraction all utilise PCA.

AUTO ENCODER

it is possible to learn a compressed representation of raw data using a particular kind of neural network called an auto encoder. An encoder and a decoder sub-model make up an auto encoder. The input is compressed by the encoder, and the decoder makes an attempt to reconstruct the input from the compressed version that the encoder has provided. The decoder model is deleted after training and the encoder model is saved.

CLASSIFICATION

Deep learning uses an artificial recurrent neural network architecture called LSTM (Long Short-Term Memory). The LSTM has feedback connections, unlike conventional feed-forward neural networks.

ALGORITHM USED:

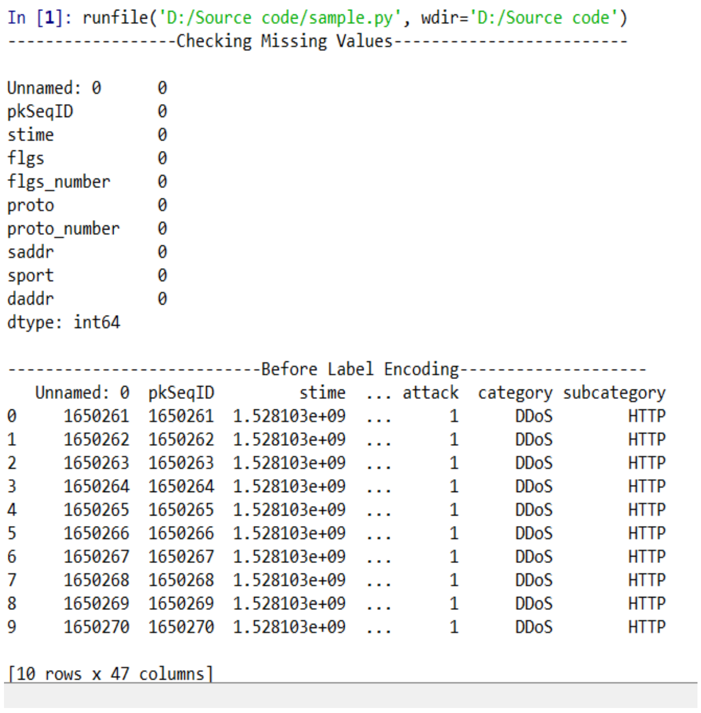
```
Algorithm 1 Pre-Processing
Require: Dataset files Dâ, Dataset Rows R, Dataset Columns C, Combined Dataset D, One Hot Encoding
OHE Ensure: Pre-processing of D
Step 1: function Pre-processing (Dâ)
Step 2: D = Merge (Dâ)
Step 3: D = Convert Datetime to integer
Step 4: D = Labels = OHE(Labels)
Step 5: D = Convert Source ip to integer
Step 6: D = Convert Destination ip to integer
Step 7: for R ← 1 to Rows do
Step 8: for C ← 1 to Columns do
Step 9: if D[R][C] in [‘nan’, ‘infinity’, ‘null’, ‘’, ‘NaN’] then
Step 10: Drop Sample (Row)
Step 11: end if
Step 12: end for
Step 13: end for
Step 14: Save D as CSV
Step 15: end function
```

Algorithm 2 DNN-LSTM Data training input: There are X network features for every Y unit of network traffic.

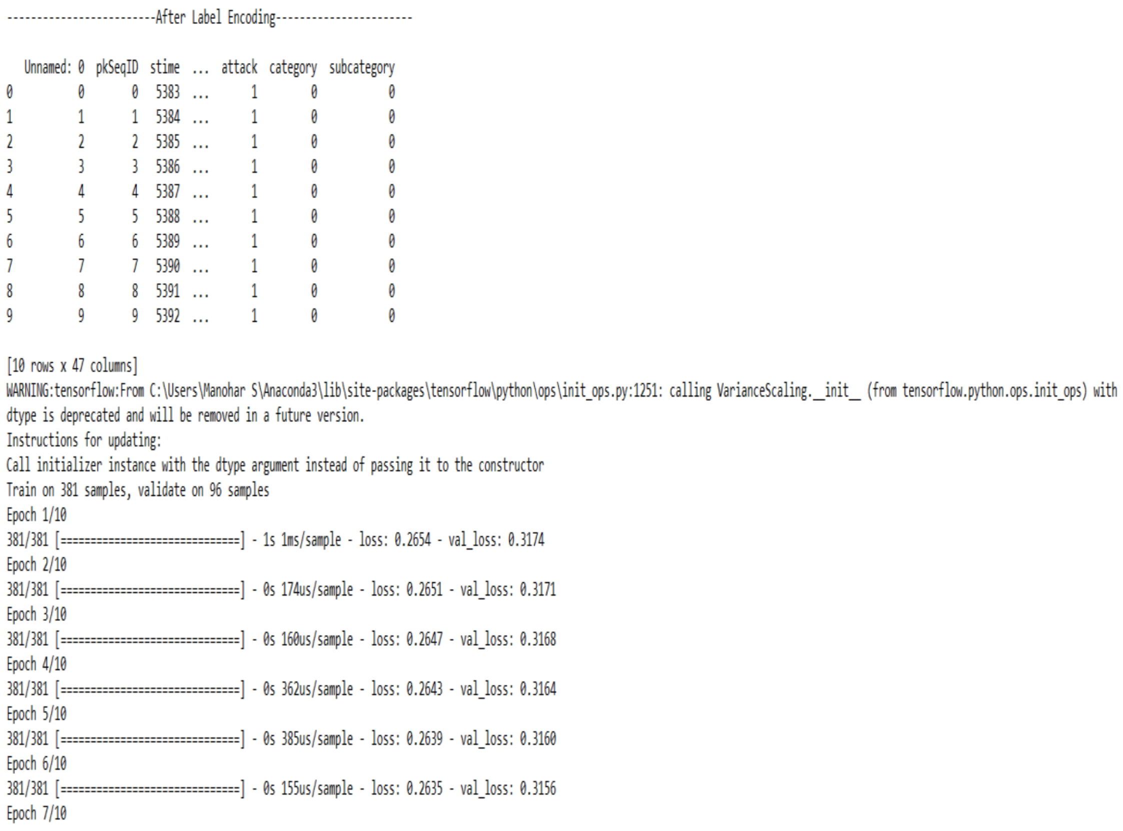
```
 Output: is N for normal or label attacks For each Yi for N
 |
Ci = CNN (Yi) process
 |
End
|
For each Ci process
Li = LSTM (Ci) process
 |
 Merge
 |
 End
 For Each Li Process
N = SoftMax (Li) end.
```

RESULTS:

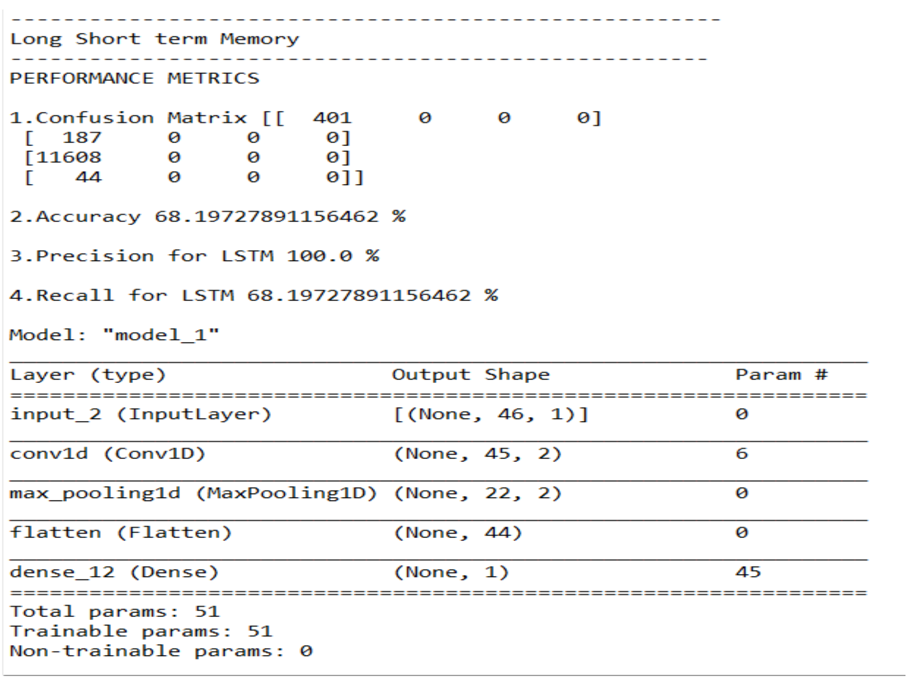
We made use machine learning algorithms, out of which LSTM proved to be the best. Finally, the experimental results shows that the performance metrics such as accuracy, precision, recall and confusion matrix.



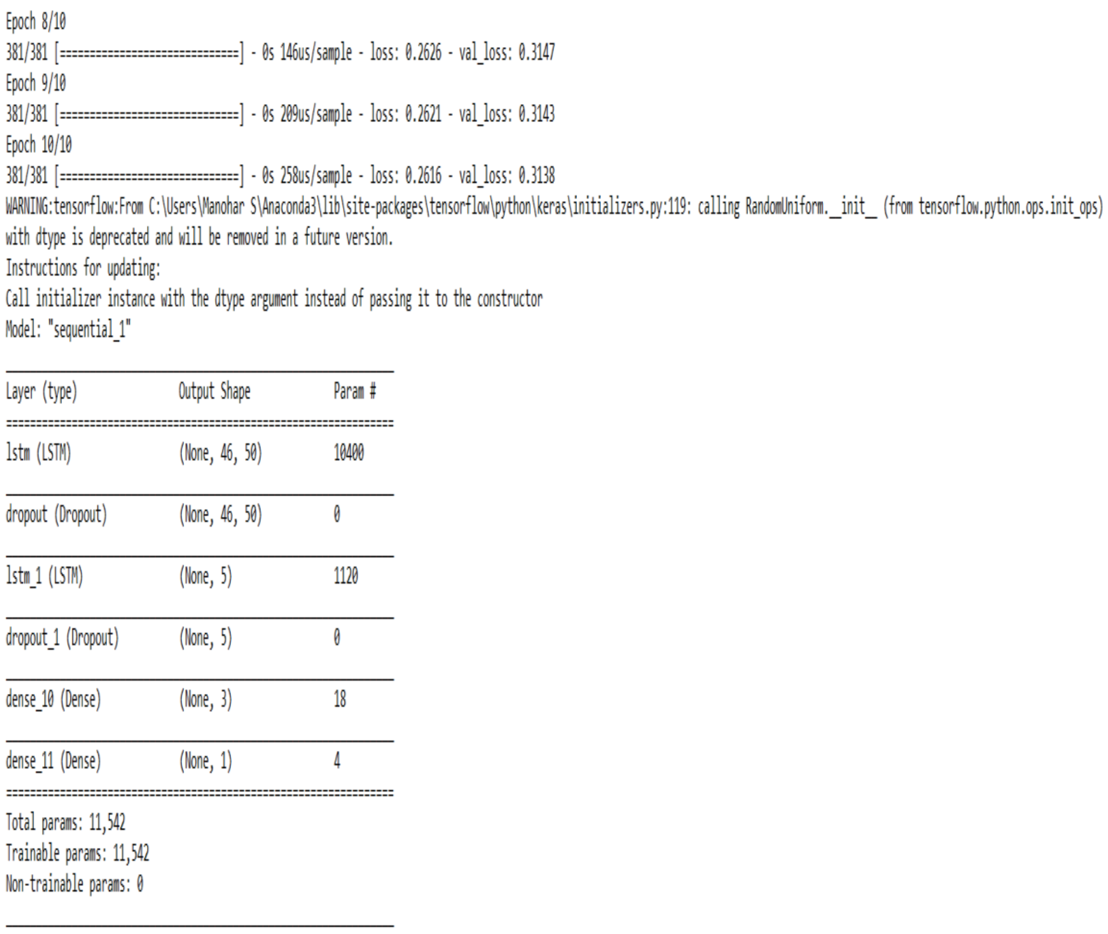
```
In [1]: runfile('D:/Source code/sample.py', wdir='D:/Source code')
-----------------Checking Missing Values------------------------

Unnamed: 0      0
pkSeqID         0
stime           0
flgs            0
flgs_number     0
proto           0
proto_number    0
saddr           0
sport           0
daddr           0
dtype: int64

--------------------------Before Label Encoding--------------------
   Unnamed: 0  pkSeqID         stime  ... attack  category subcategory
0     1650261  1650261  1.528103e+09  ...      1      DDoS        HTTP
1     1650262  1650262  1.528103e+09  ...      1      DDoS        HTTP
2     1650263  1650263  1.528103e+09  ...      1      DDoS        HTTP
3     1650264  1650264  1.528103e+09  ...      1      DDoS        HTTP
4     1650265  1650265  1.528103e+09  ...      1      DDoS        HTTP
5     1650266  1650266  1.528103e+09  ...      1      DDoS        HTTP
6     1650267  1650267  1.528103e+09  ...      1      DDoS        HTTP
7     1650268  1650268  1.528103e+09  ...      1      DDoS        HTTP
8     1650269  1650269  1.528103e+09  ...      1      DDoS        HTTP
9     1650270  1650270  1.528103e+09  ...      1      DDoS        HTTP

[10 rows x 47 columns]
```



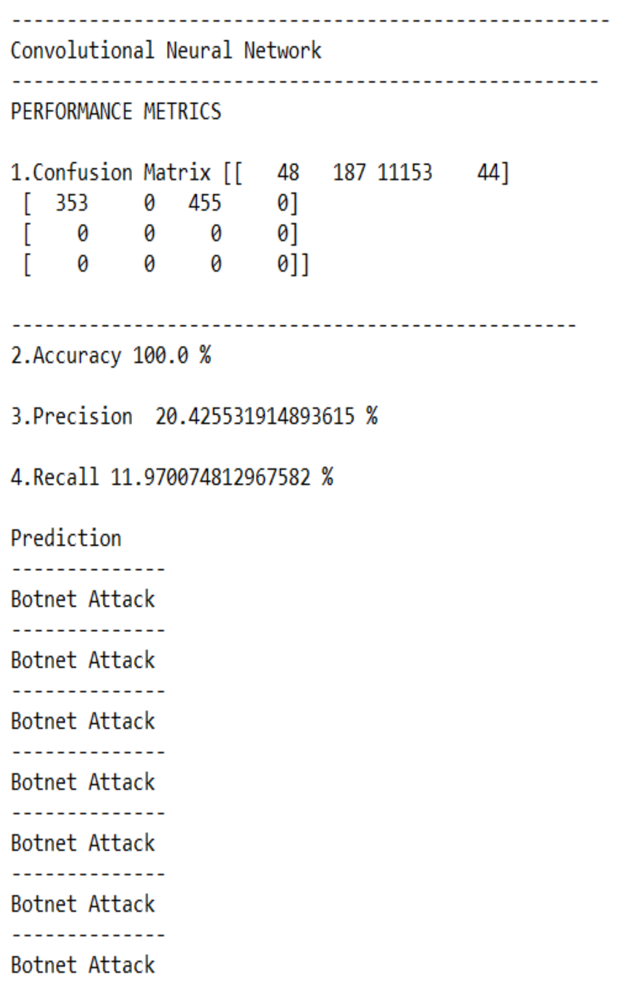
```
-----------------------After Label Encoding----------------------

   Unnamed: 0  pkSeqID  stime  ...  attack  category  subcategory
0           0        0   5383  ...       1         0            0
1           1        1   5384  ...       1         0            0
2           2        2   5385  ...       1         0            0
3           3        3   5386  ...       1         0            0
4           4        4   5387  ...       1         0            0
5           5        5   5388  ...       1         0            0
6           6        6   5389  ...       1         0            0
7           7        7   5390  ...       1         0            0
8           8        8   5391  ...       1         0            0
9           9        9   5392  ...       1         0            0

[10 rows x 47 columns]
WARNING:tensorflow:From C:\Users\Manohar S\Anaconda3\lib\site-packages\tensorflow\python\ops\init_ops.py:1251: calling VarianceScaling.__init__ (from tensorflow.python.ops.init_ops) with
dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Train on 381 samples, validate on 96 samples
Epoch 1/10
381/381 [==============================] - 1s 1ms/sample - loss: 0.2654 - val_loss: 0.3174
Epoch 2/10
381/381 [==============================] - 0s 174us/sample - loss: 0.2651 - val_loss: 0.3171
Epoch 3/10
381/381 [==============================] - 0s 160us/sample - loss: 0.2647 - val_loss: 0.3168
Epoch 4/10
381/381 [==============================] - 0s 362us/sample - loss: 0.2643 - val_loss: 0.3164
Epoch 5/10
381/381 [==============================] - 0s 385us/sample - loss: 0.2639 - val_loss: 0.3160
Epoch 6/10
381/381 [==============================] - 0s 155us/sample - loss: 0.2635 - val_loss: 0.3156
Epoch 7/10
```



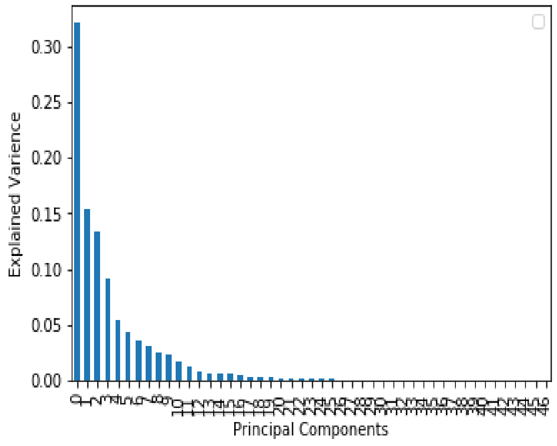
```
--------------------------------------------------------
Long Short term Memory
--------------------------------------------------------
PERFORMANCE METRICS

1.Confusion Matrix [[   401      0      0      0]
 [   187      0      0      0]
 [11608      0      0      0]
 [    44      0      0      0]]

2.Accuracy 68.19727891156462 %

3.Precision for LSTM 100.0 %

4.Recall for LSTM 68.19727891156462 %

Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 46, 1)]           0
_________________________________________________________________
conv1d (Conv1D)              (None, 45, 2)             6
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 22, 2)             0
_________________________________________________________________
flatten (Flatten)            (None, 44)                0
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 45
=================================================================
Total params: 51
Trainable params: 51
Non-trainable params: 0
```



```
Epoch 8/10
381/381 [==============================] - 0s 146us/sample - loss: 0.2626 - val_loss: 0.3147
Epoch 9/10
381/381 [==============================] - 0s 209us/sample - loss: 0.2621 - val_loss: 0.3143
Epoch 10/10
381/381 [==============================] - 0s 258us/sample - loss: 0.2616 - val_loss: 0.3138
WARNING:tensorflow:From C:\Users\Manohar S\Anaconda3\lib\site-packages\tensorflow\python\keras\initializers.py:119: calling RandomUniform.__init__ (from tensorflow.python.ops.init_ops)
with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 46, 50)            10400
_________________________________________________________________
dropout (Dropout)            (None, 46, 50)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 5)                 1120
_________________________________________________________________
dropout_1 (Dropout)          (None, 5)                 0
_________________________________________________________________
dense_10 (Dense)             (None, 3)                 18
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 4
=================================================================
Total params: 11,542
Trainable params: 11,542
Non-trainable params: 0
_________________________________________________________________
```



```
-------------------------------------------------------
Convolutional Neural Network
------------------------------------------------------
PERFORMANCE METRICS

1.Confusion Matrix [[   48   187 11153    44]
 [  353     0   455     0]
 [    0     0     0     0]
 [    0     0     0     0]]

----------------------------------------------------
2.Accuracy 100.0 %

3.Precision  20.425531914893615 %

4.Recall 11.970074812967582 %

Prediction
--------------
Botnet Attack
--------------
Botnet Attack
--------------
Botnet Attack
--------------
Botnet Attack
--------------
Botnet Attack
--------------
Botnet Attack
--------------
Botnet Attack
Botnet Attack
--------------
Botnet Attack
--------------
Botnet Attack
--------------
Botnet Attack
```



CONCLUSION:

This research builds a flexible hierarchical key access control mechanism using Shamir's secret sharing method that may be applied in a range of real-time scenarios, especially for any cloud infrastructures. The need for public and private storage is one of the major costs for data owners. Concerns concerning the security of a hierarchically structured data access policy have diminished as a result of our proposal. The proposed key access control technique provides a computationally efficient method for key generation.The suggested strategy combines the security of a private cloud with the public cloud's functionality, usability, and cost-savings. The dependability of the public cloud and its little upkeep and management requirements are additional advantages for enterprises. Here are some other advantages:

1) The data owner has full control over the data;

2) It allows for hierarchical and organisation unit-based management of the data to be processed on a public cloud;

3) It provides organisation unit and user group-based security level policies, specifically the multi-hierarchical security mechanism;

4) The data owner is able to dynamically adjust the company's security level policies; and

5) The user can be a member of any user group for any organisational unit. So, the information that belongs to a different organisational unit is under the user's control.

REFERENCES:

1. E. Thomas, P. Ricardo, and M. Zaigham, Cloud Computing: Concepts, Technology, and Architecture, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2013.
2. Information Technology-Cloud Computing-Reference Architecture, Standard ISO/IEC 17789:2014, 2014.
3. Information Technology-Cloud Computing-Overview and Vocabulary, Standard ISO/IEC 17788:2014, 2014.
4. S. J. Mackinnon, P. D. Taylor, H. Meijer, and S. G. Akl, ‘‘An optimal algorithm for assigning cryptographic keys to control access in a hierarchy,’’ IEEE Trans. vol. C-34, no. 9, pp. 797–802, Sep. 1985.
5. L. Harn and H.-Y. Lin, ‘‘A cryptographic key generation scheme for multilevel data security,’’ vol. 9, no. 6, pp. 539–546, Oct. 1990.
6. H. T. Liaw, S. J. Wang, and C. L. Lei, ‘‘A dynamic cryptographic key assignment scheme in a tree structure,’’ Math. Appl., vol. 25, no. 6, pp. 109–114, Mar. 1993.
7. M. S. Hwang, C. C. Chang, and W. P. Yang, ‘‘Modified Chang-Hwang-Wu access control scheme,’’ Electron. Lett., vol. 29, no. 24, pp. 2095–2096, 1993.