International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-8 Issue-5, March 2019

Network Application Identification Using Deep Learning

Anjali T

bstract: ***The network traffic is increasing exponentially. Managing the vulnerabilities and threats become a major issue due to the heavy volume of data involved. To deal with such problems, network administrators use their experience and understanding of different applications running in their network, to monitor the packet traffic. Here identification and classification of different application that dumps data into the network, becomes challenging. The traditional way is by using behavioral signatures such as port number, application header, transmission frequency, destination IP etc. Although this is still the popular method, it can be beaten by malicious apps and users, by random port changes, proxies, protocol tunneling, and many other tricks. To overcome this issue a technique called flow feature-based analysis can be employed. In this paper, we present a deep learning-based data signature analyses which will identify applications by analyzing the information in traffic flow and some results we have observed. Mainly we are using convolutional neural network based classification and autoencoder based feature extraction to improve the efficiency.***

Index Terms: ***Auto encoders, Convolutional Neural Networks, Deep Learning, Internet Applications, Web Browser.***

I. INTRODUCTION

Network applications are interactive and compiled application that can perform several tasks either as client or a server. The increasing network traffic became a headache to the network administrators since it is difficult for them to identify traffic. Traditionally network traffic identification is done on the basis of some behaviors of these applications. These features may include port numbers, Transmission rate, transmission frequency, application and protocol header information etc. To deal with the increasing number of threats in a network the network administrators need a good understanding of different applications using the network [1]. The major challenge here, has been, to identify applications in real time. The problem is caused by a large increase in the number of web and mobile applications and users. Along with this, administrators also face issues like tunneling, random port usage, proxy and encryptions that makes detection almost impossible. Generally, network operators use port numbers to identify network traffic. For example, port number 80 to be HTTP traffic, 25 to be SMTP so on and so forth. This is computationally efficient and simple and was very effective decades ago. The problem became tougher, when applications like P2P network have started using some non-standard ports known as Random port usage. Sometimes illegal or malicious applications try to hide their traffic inside another applications' data gram or a different legal protocol, which is knows as Tunneling. To avoid all such mockery of behaviors, a flow feature-based analysis or data signature analysis is mandatory. Behavioral signature can be mocked, copied, changed or tampered with, but data signature is abstract and cannot be bypassed that easily.

This can be defined as a multi class machine learning problem, trying to identify in which class an application belongs to by analyzing information from traffic flow. This can be done by retrieving features such as number and size of packets per flow, flow duration and inter-packet arrival time and many more. Selection and processing of the right features from a frenzy of unintelligible data is not humanly possible. This will be an ideal deep learning-based problem [2].

Deep learning is a new revolutionary machine learning strategy, where in the machine will extract features automatically from the given data for the best classification possible. These features are at best, non-orthogonal and significantly enhance the accuracy of classification or regression, compared to human hand-crafted feature [3] [4]. Some supervised learning algorithms include logistic regression, multilayer perceptron, deep convolutional network etc. Semi or unsupervised learning include stacked auto encoders, restricted Boltzmann machines (RBMs), deep belief networks (DBNs) etc. [5]. Now a days Deep learning is used in variety of artificial intelligence problems like speech recognition, computer vision, and natural language processing [6] [7].

A. Convolutional Neural Network

Convolutional Neural Networks are very similar to the normal neural networks but CNN has the ability to visualize data as images and this property allows users to encode certain properties into the architecture. CNNs are biologically-inspired variants of multi-layer perceptron which are designed to use minimal amount of preprocessing [8].

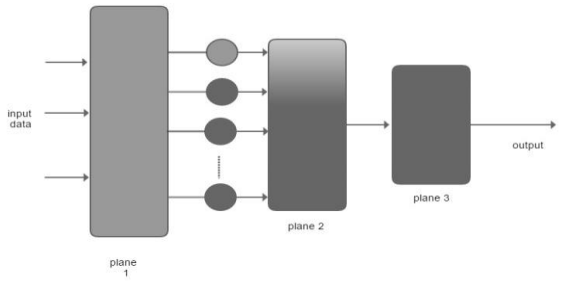

Figure 1: Basic Convolutional Neural Network Architecture

Manuscript published on 30 March 2019.
***Correspondence Author(s)**
Anjali T, Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chenai

CNNs are widely used for solving various machine learning problems related to image and video recognition. The layers of a CNN have neurons in three dimensions: width, height, and depth.

Here, plane 1 represents input layer of a convolutional network. The elements of plane 1 are locally processed by the weights and biases and then acted upon by the activation function. In the fig 1 the activation functions are represented with circles. This operations results in plane 2 and dimensions of plane 2 can be different from that of plane 1 depending on zero padding was done or not. Plane 2 may sometimes down sample based on some criteria and this forms plane 3. Plane 3 can be an input to either another convolutional layer or a fully connected layer. A CNN may have number of convolutional layers and finally it ended with a fully or partially connected layers. The performance of a CNN architecture is decided by the number of filters in each layer and number of layers the CNN has. Number of filters in each layer decides the feature extracted form that layer. As the number of layers are getting increased, computational complexity also increased. Techniques like dropout [9] are used during training process to avoid the problems of over fitted model that can cause poor prediction accuracy [10].

B. Auto Encoders

Auto encoders are implemented similar to other neural network, but in this case, it is trained to learn the input itself. Number of input and output dimensions will be same in the case of auto encoders, which means that the network is built in such a way that it should be able to reconstruct the input data. Auto encoders are the neural network typically made for the purpose of dimensionality reduction. The architecture of an auto encoder is similar to the MLP, it is having one input layer, one or more hidden layers and finally the output layer. If the Auto encoder is having multiple hidden layers, then the features extracted from one layer are further processed to different features and these should be capable of reconstructing the data. This ANN is used for unsupervised learning where an Auto encoder is able to learn the representation for a set of data. In usual classification methods, some particular features are selected initially and then calculated for all data points and the it is fed to classification algorithm as input, but in auto encodes since it is following unsupervised approach, different features are extracted from different layers and this can be used for classification.

II. METHODOLOGY

A. Data Collection and Preprocessing

Deep Learning specifically is very heavy, on the computational time and resource required while training with large amount of data. But with GPUs, it is proved to be faster and better for large scale data processing. The popular deep learning platforms like Tensor Flow, Theano and Caffe has come up to resolve the problems of large data handling. We have chosen the Google's Tensor Flow framework to implement our network, since it offers a relatively easy Python API [11].

The data sets for the experiment is collected using Wire-shark network analyzer [12]. Wireshark puts the network interface controller on a promiscuous mode, so that all the traffic on that interface is visible to the users. This work proposes methods to identify two different applications by analyzing packet attributes which can be collected using Wireshark. The data set consists of packet attributes collected for two different browsers. Google chrome, Internet Explorer, Mozilla Firefox, Safari, Opera are some of the commonly used web browsers.

The first class contains 25,000 packets of Mozilla Firefox and in the second class 24,987 packets of Opera browser. The approach is to try and identify the two browsers by analyzing their packet attributes. The specific choice was made under the belief that they have no common code or development library which will result in very less similarity in transmission and data signatures. The identification is done using auto encoders which was already done by [13] with a suggestion to implement the same problem using CNN. However, we are not using the full data and use only meta data making the whole process computationally. simple. The captured packet attributes are further preprocessed in order to feed it into the deep networks. The data in the original form of alpha numeric values is directly fed in.

B. Implemented Convolutional Neural Network Architecture

The packet attributes collected using Wireshark packet analyzer is preprocessed and given to a convolutional neural network. A four-layer convolutional neural network was implemented with two convolutional layers and two fully connected. The network topology with weight dimensions is shown in fig 2.

The input data is fed into the network as a two-dimensional vector, each samples of size 512. The first 2 convolutional layers implemented has 64 and 32 size filters. The first layer is one dimensional and second layer, two dimensional. Two fully connected layers at the end predict the label corresponding to the data.

The network uses a soft-max activation. To avoid over fitting the parameters were regularized along with the error function. Drop out is done during training in order to avoid over fitting [9].

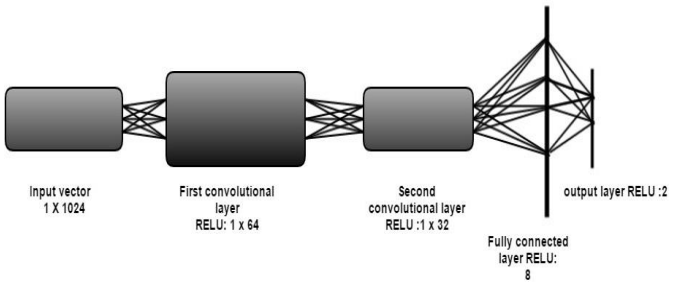

Fig 2. Implemented CNN Architecture

Let the set of unlabeled training set be $\{x^{(1)}, x^{(2)}, x^{(3)}\ldots\}$ where $x^{(i)} \epsilon R^{(n)}$. The input of the network is N here N is equal to 6 and the input x is reconstructed as $x\hat{}$. Our aim is to choose weights and biases in such a way that error between x and is x^ as small as possible. y is constructed form x using the transformation given in the eqn 1.

$$y_{1\times m} = f(x_{1\times n} w_{n\times m} + b_{1\times m}) \qquad (1)$$

where w and b represent weights and biases respectively for the first layer and f is any activation function.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

In the next layer x is reconstructed form y as x^ using the transformation as given in the eqn 2.

$$\hat{x}_{1\times n} = g(y_{1\times m}\,\tilde{w}_{n\times m} + \tilde{b}_{1\times m}) \quad (2)$$

Where $\tilde{w}$ and $\tilde{b}$ represents weights and biases and g is the activation function corresponding to that layer.

C. *Implemented Auto Encoder Architecture*

The collected data samples were fed into the implemented auto encoder shown in the fig 3. This is a single layer auto encoder with 512 nodes in the middle layer and tanh is used as the activation function in all the layers 3.

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (3)$$

The input to this network is 1024 length packet attributes. The result of this is fed into the 512-node layer. The objective function of the optimization problem for training was taken as the root mean square error between the outputs of final nodes and the inputs. Gradient descent is used to resolve this optimization function. The middle layer samples were taken as features for classification. These selected features are fed into the convolutional neural network and classified the features.

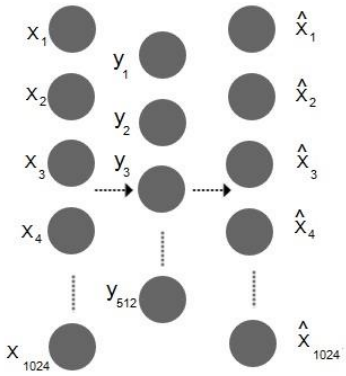

Fig 3. Implemented Auto Encoder Architecture

III. RESULT

A. *CNN based classification*

This section discusses the classification results using the proposed convolutional neural network. The dataset consists of 49,987 data points. 25,000 data packets in class 1, 24,987 data packets in class 2. From the dataset 70% was taken for training and remaining 30% was taken for testing. Classification accuracy of 80.1% was given for 10,000 iterations. In this approach, we are classifying only on the basis of the packet attributes.

It is not easy to deal with the entire packet payload since the computation overhead is high. But this proposed method is able to give significant results by just considering the packet attributes only. The accuracy obtained for different number of iterations is given in the table I.

	No of Iterations	Accuracy obtained
1	300	71.05
2	350	72.85
3	400	71.39
4	450	74.73
5	500	74.80
6	1000	75.78
7	2000	76.25
8	5000	78.78
9	7000	78.90
10	**10000**	**80.10**

Table I. Result obtained for CNN

B. *Autoencoder based classification*

The features extracted using the implemented auto encoder was fed into the already designed CNN. The results obtained for auto encoder-based classification for different number of iterations is given the table II. The result observed in this method is compared with the result of table I. The highest accuracy obtained is 85.5%It was found that the accuracy getting after the feature extraction using one-layer auto encoder is higher than the classification results obtained for CNN.

	No of Iterations	Accuracy obtained
1	300	69
2	350	72.25
3	400	72.57
4	450	74.89
5	500	74.98
6	1000	78
7	2000	79.7
8	5000	80.45
9	7000	82.7
10	**10000**	**85.5**

IV. CONCLUSION

This project identifies two different browsers (Opera and Mozilla Firefox) by analyzing packet attributes only. The deep learning platform of Tensor Flow from Google were used for implementation of different deep learning architectures. The classification was done in two different methods. In the first approach, the preprocessed data packets were directly given to a convolutional neural network. In the second approach the features were extracted from a single layer auto encoder and these features were further fed into the convolutional neural network. The results obtained were compared and observed that the accuracy obtained for the second method is considerably high as compared to that of first method. In the second method, auto encoders were extracting the features. So, it will be more effectively classified by the CNN classifier. The contribution of this work is it can achieve significant accuracies by just taking the packet attributes only.

FUTURE WORK

Presently this work has considered data from only two different browsers.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

The identification and classification are done only by considering only the payload information. This needs to be further extended to classification of different types of applications by collecting the complete payload information. The payload information contains the actual data transmitting in a network. There is a scope for identifying anomalous activities in a network by extracting features from the captured information. This method can be further elaborated using various deep learning frameworks.

REFERENCES

1. B. Ashwini, V. K. Menon, and K. Soman, "Prediction of malicious domains using smith waterman algorithm," in International Symposium on Security in Computing and Communication. Springer, 2016, pp. 369–376.
2. A. Tongaonkar, R. Keralapura, and A. Nucci, "Challenges in network application identification." in LEET, 2012.
3. "Deep learning," [Accessed : 29-NOVEMBER-2016]. [Online].
4. Available: https://en.wikipedia.org/wiki/Deep learning
5. C. Nagananthini and B. Yogameena, "Crowd disaster avoidance system (cdas) by deep learning using extended center symmetric local binary pattern (xcs-lbp) texture features," in Proceedings of International Conference on Computer Vision and Image Processing. Springer, 2017,487–498.
6. "Deep learning tutorials," [Accessed : 29-NOVEMBER-2016]. [Online].
7. Available: http://deeplearning.net/tutorial/
8. K. Soman, R. Loganathan, and V. Ajay, Machine learning with SVM and other kernel methods. PHI Learning Pvt. Ltd., 2009.
9. K. Soman, S. Diwakar, and V. Ajay, Data Mining: Theory and Practice [WITH CD]. PHI Learning Pvt. Ltd., 2006.
10. "Neural networks," [Accessed : 02-July-2016]. [Online]. Available: http://ufldl.stanford.edu/wiki/index.php/neural networks
11. N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." Journal of Machine Learning Research, vol. 15, no. 1,1929–1958, 2014.
12. P. P. S. K. P. Athira S, Rohit Mohan, "Automatic modulation classifi-cation using convolutional neural network," IJCTA 9(16) pp 7733-7742, 2016.
13. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane,´ R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar,
14. P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas,´ O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available:
15. http://tensorflow.org/
16. A. Orebaugh, G. Ramirez, and J. Beale, "Wireshark & ethereal network protocol analyzer toolkit," 2006.
17. Z. Wang, "The applications of deep learning on traffic identification," 2010

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)