Network Intrusion Detection using Natural Language Processing and Ensemble Machine Learning

Saikat Das*, Mohammad Ashrafuzzaman†, Frederick T. Sheldon† and Sajjan Shiva*
*Department of Computer Science, The University of Memphis, Memphis, TN, USA
†Department of Computer Science, University of Idaho, Moscow, ID, USA
Emails: {sdas1, sshiva}@memphis.edu, ashr3866@vandals.uidaho.edu, sheldon@uidaho.edu

***Abstract*—We propose an intrusion detection system (NLPIDS) that utilizes natural language processing and ensemble-based machine learning. The proposed NLPIDS converts natural language HTTP requests into vectors which are then used to train several supervised and ensemble-based machine learning models. The trained models are then used to detect anomalous traffic. We validated our method using HTTP DATASET CSIC 2010. The results show the efficacy of the NLPIDS by producing better F1-score (0.999) and negligible false alarms (0.007) compared to existing methods. The NLPIDS does not depend on attack methods and feature vectors.**

***Index Terms*—Anomaly Detection, Network Intrusion Detection, Natural Language Processing, Ensemble Machine Learning, NLPIDS.**

I. Introduction

An intrusion detection system (IDS) is a software or hardware implementation that monitors a system (software, hardware, OS, network, etc.) for malicious activities and policy violations. Deviating from the normal activity or any policy violation is treated as a potential threat and typically reported to an administrator or centralized collection for further investigation and mitigation.

Traditional intrusion detection techniques use either pattern matching or blacklisting. A pattern matching IDS stores a long list of known attack patterns, and checks if the incoming traffic matches with certain string (exact match) or a pattern (by using regular expression). In a blacklisting IDS a firewall or a proxy server maintains a list of malicious servers and denies access for any server in the list. In both cases they fail to identify unknown or zero-day attacks or new malicious server. On the other hand, behavior based detection technique analyzes the attack behavior, and detects attacks with a higher detection rate than the previous two types. However, all three types of IDSs are unable to detect some well-known attacks, like Drive-by Download attack (DbD) [1], C&C traffic [2], and unseen malicious traffic.

Machine learning (ML) based IDS can better detect new patterns or behaviors. Addition of natural language processing (NLP) can enhance the detection accuracy of these IDSs as NLP-based detection mechanisms do not rely on the attack techniques. The NLP and ML based system can be used as an additional module to a traditional IDS. Whenever, this module identifies a so far unseen attack, the pattern or origin of this malicious activity can be added to the database of known attacks.

In this paper, we propose an NLP and ML based network intrusion detection system (NLPIDS). The NLPIDS converts HTTP requests from natural language text to feature vectors using NLP techniques. These feature vectors are used to train machine learning models. The machine learning module consists of a number of supervised methods and ensembles of those supervised methods.

The main contribution of this paper is the development of an intrusion detection mechanism that analyzes natural language based network traffic using NLP techniques and detects anomalous, potentially malicious, traffic using an ensemble-based machine learning scheme. The proposed method was validated using a public dataset, namely the HTTP DATASET CSIC 2010 [3], and evaluated using standard metrics. The results of these experiments show that our proposed method performs better than existing methods discussed in Section II.

The rest of the paper is organized as follows. Section II summarizes some of the related works. Section III briefly describes two natural language tools used in the proposed method. Section IV presents the natural language and ensemble-based method to detect network intrusions. Section V describes an experiment using the proposed method and discusses the results. Section VI concludes the paper.

II. Related Work

In detecting network intrusions, packet traces are the most popular methods on traffic classification. However, analyzing network packets became intractable in today's broadband network. Traffic classification based on network logs is the alternative solution of packet analysis. Alternatively, traffic classification based on network logs such as NetFlow [2], DNS records [4] and proxy server logs [5] are popular way to detect network intrusions.

Recently, in detecting network intrusions, machine learning (ML) techniques are being used to discriminate malicious traffic from benign traffic. Detecting phishing attacks [6] using

978-1-7281-2547-3/20/$31.00

NLP and ML, log-mining using NLP [7], intrusion detection using NLP [8], intrusion detection in cloud [9], etc. are found from the recent research. Various types of supervised [10], unsupervised [11], and ensemble-based [12] machine learning approaches including various feature selection approaches [13] are also used in detecting anomalies.

Lilleberg et al. [14] proposed a text classification using support vector machine (SVM) and word2vec. They showed that the combination of word2vec and tf-idf outperform tf-idf. However, they haven't mentioned SVM or any ML classification model anywhere in their paper. Also, doc2vec is more efficient than word2vec method. Min et al. [15] implemented an NIDS based on text-convolutional neural network (CNN) and random forest(RF) using statistical and payload features. They used CNN to extract the effective information from the payloads in network traffic from the ISCX2012 dataset and random forest to classify them. However, there is no proper justification of choosing RF model in their research, as well as comparison of other classification results are missing. Mimura et al. [16] proposed a network interface packet analysis method using source and destination IP in their NLP, which does not make any significant difference in detecting anomaly. They reported F1-score of 98.00% as the best performing metric for their method. Generally, a lot more information is captured in a log file instead of a network packet. In addition, a comparative performance analysis using several ML models are missing here. Li et al. [17] used weighted word2vec, and LightGBM and CatBoost. Their method achieved 99.49% as accuracy. Althubiti et al. [18] used six different ML methods, namely RF, LR, J48, ABc, SGDc and NB, to analyze HTTP DATASET CSIC 2010. They found RF to be the best-performing model with an F1-score of 99.90%.

In contrast, our proposed method works on the HTTP requests that contain textual details of activity history which is more suitable for NLP. Moreover, detailed information enables the IDS to detect the anomalies better. In addition, it generates a fixed length of vectors from any arbitrary length paragraph where the vectors are further used in model classification. The dataset we used, namely the HTTP DATASET CSIC 2010, is a collection of natural language HTTP requests and hence is more suitable for NLP compared to KDD99 dataset.

III. Background: Natural Language Tools

In our proposed method, we use natural language techniques to build the vector spaces from text corpus and use the vector spaces to train machine learning models to detect the anomaly. Here we briefly discuss two NLP methods and an open-source software used in our work.

A. Word2Vec

Word embedding is a popular technique used to preprocess linguistic texts for natural language processing using machine learning or statistical analyses. Word embedding process maps each unique word or phrase in a text corpus to vectors of real numbers. Word2vec [19] is a popular word embedding method. Word2vec uses a shallow two-layer neural network to learn word and phrase associations from a corpus of text and the trained model can predict suitable words or phases for incomplete sentences. Word2vec is based on the distributional hypothesis and uses two algorithms: 1) continuous-bag-of-words (CBoW) to predict correct word or phrase from the contextual window of surrounding words, and 2) skip-gram to predict window of surrounding words from a word or phrase. Both the algorithms use vector representation of words and phrases.

B. Doc2Vec

While word2vec produces just word embedding, an extension of it, called doc2vec or paragraph2vec, is used to construct embedding from entire documents by representing documents as vectors [20]. The document or paragraph vectors are also trained along with word vectors to predict a missing word or phrase. Similar to word2vec, doc2vec is also based on the distributional hypothesis, and uses two algorithms: 1) distributed-memory (DM), an extension of CBoW, to predict a word or phrase from a contextual windows of a paragraph or document, and 2) distributed-bag-of-words (DBoW), an extension of skip-gram, to predict a contextual window of paragraph or document from a word or phrase.

Doc2vec enables calculation of the semantic similarity between two paragraphs or documents and thus infer similar paragraphs or documents semantically. This function is important to develop a practical system to detect unseen malicious traffic by comparing traffic data from network packets, and locating words or sentences in traffic data that are not contextually similar.

C. Gensim

Gensim [21] is an open-source library for topic modeling, document indexing and natural language processing, using machine learning. It includes implementation of word2vec and doc2vec. Gensim, implemented in the Python and Cython programming languages, is used to analyze plain-text documents for inferring semantic similarity.

IV. Proposed Method

This section provides an overview of the natural language processing and machine learning based scheme to detect network intrusion. Fig. 1 shows the workflow of the proposed method. The scheme has three phases: 1) data pre-processing phase, 2) natural language processing phase, and 3) ensemble-based machine learning phase. In the first phase, natural language content of the file of HTTP requests is converted into bag of words, and the bag of words creates sentences. Then the collection of these sentences are converted to labeled vector space. These vector-spaces are used to train ensemble machine learning framework, and the trained models are used to detect network intrusion as anomalies in the data.

A. Conversion of HTTP Requests to Corpus

An HTTP request command in a network contains many different kinds of information. Relevant items from the HTTP requests are extracted using a pre-processing tool and converted

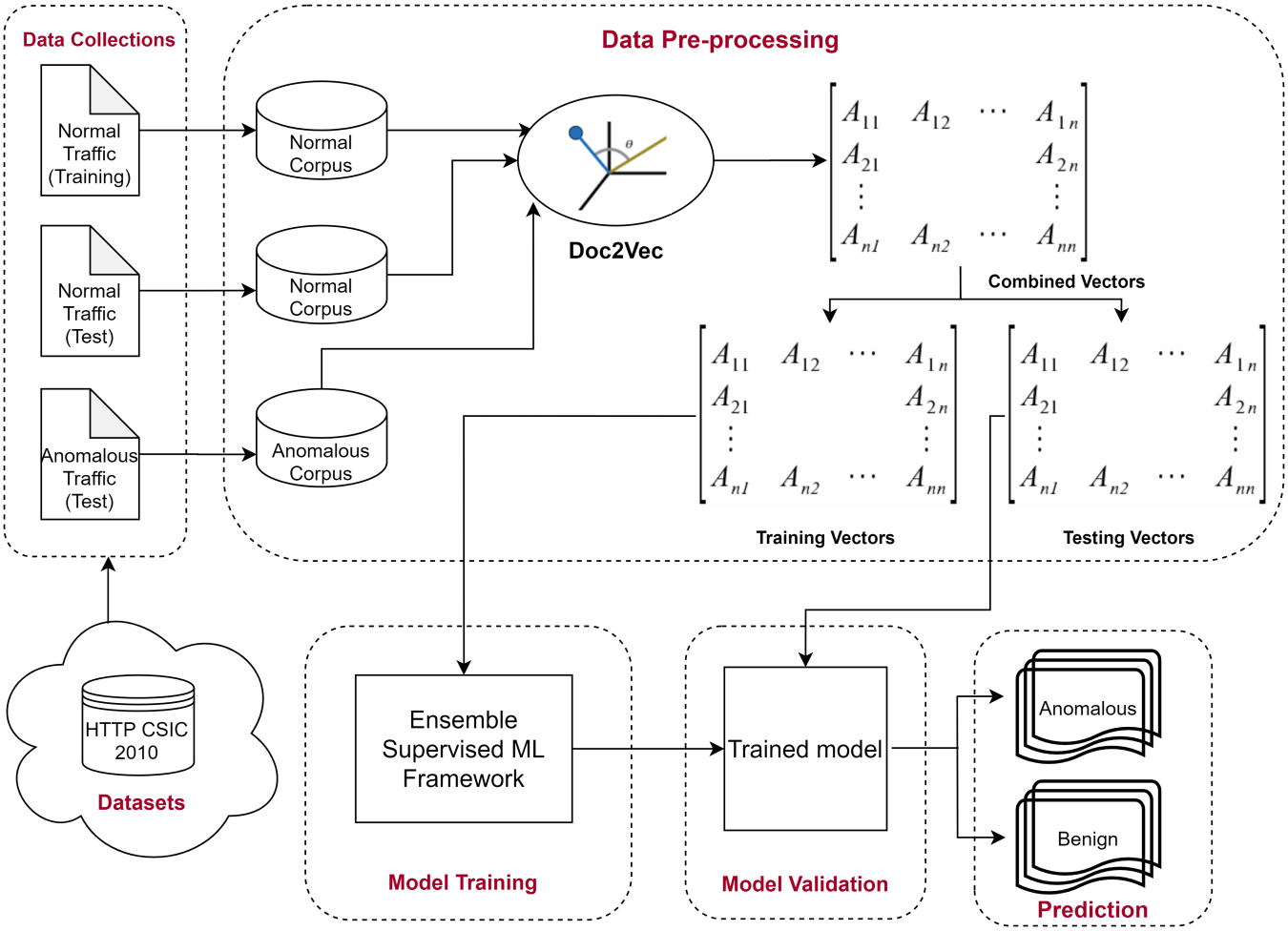

Fig. 1. Overview of the NLPIDS.

into words. These words generate a single natural language sentence for a corresponding HTTP request. A paragraph is constructed by collating a number of sentences and stored in a document. Collection of all these documents constitute a corpus. Fig. 2 shows an HTTP request and Fig. 3 shows a document containing a paragraph of ten sentences extracted from ten HTTP requests. From these figures it can be seen that only relevant information was extracted to construct a sentence from one HTTP request. The pre-processing phase also parses the encoded strings in the HTTP requests into tokens.

B. Construction of Vector Space Model

In this phase, the documents in the corpus as created above are converted into vector space model using gensim [21]. For all three corpora, the consisting documents are converted into feature vectors using doc2vec. Each arbitrary length document generates a fixed length of feature vectors using the semantic similarities among all documents. So, the M number of documents in a single corpus generate a matrix of M x N, where N is the fixed length of feature vectors. We combined all three corpora by combining all three tagged M x N matrices. Then split the combined matrix into training and testing data which are stored in CSV files.

C. Ensemble Machine Learning

The third phase in the workflow uses a machine learning framework to classify the vector data to detect network anomalies. The machine learning framework uses ensemble-based learning, where multiple classifiers are used together and the results given by these constituent classifiers are further classified by another classifier [22, 23].

Ensemble is a two-stage framework. In the first stage, supervised classifiers, namely logistic regression (LR), support vector machines (SVM), naïve Bayes with Gaussian function (NB), decision tree (DT), and neural networks (NN), are used to classify the data. In the second stage, the classification results by these individual classifiers are used as input to another classifier, called the *ensemble classifier*. In our framework, majority voting (Ens_MV), logistic regression (Ens_LR), naïve Bayes (Ens_NB), neural network (Ens_NN), decision tree (Ens_DT), and support vector machine (Ens_SVM) are used as the ensemble classifiers. Details of this ensemble mechanism can be found in Das et al. [24].

```
GET http://localhost:8080/tienda1/publico/anadir.jsp?id=2&nombre=Jam%F3n+Ib
%E9rico&precio=85&cantidad=%27%3B+DROP+TABLE+usuarios%3B+SELECT+*+FROM+datos+WHERE
+nombre+LIKE+%27%25&B1=A%F1adir+al+carrito HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml
+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=B92A8B48B9008CD29F622A994E0F650D
Connection: close
```

Fig. 2. Details of a single HTTP request.

```
1) id 2 nombre Jam◆n Ib◆rico precio 85 cantidad '; DROP TABLE usuarios; SELECT * FROM
datos WHERE nombre LIKE '% B1 A◆adir al carrito
2) id 2/ nombre Jam◆n Ib◆rico precio 85 cantidad 49 B1 A◆adir al carrito
3) modo entrar login bob@<SCRipt>alert(Paros)</scrIPT>.parosproxy.org pwd 84m3ri156
remember on B1 Entrar
4) modo entrar login grimshaw pwd G//lAc,IAr remember on B1 Entrar
5) modo entrar login grimshaw pwd 84m3ri156 rememberA on B1 Entrar
6) idA 2
7) errorMsg +
8) errorMsgA Credenciales incorrectas
9) modo insertar precio 183','0','0');waitfor delay '0:0:15';-- B1 Pasar por caja
10) id 2 nombre Jam◆n Ib◆rico precio 85 cantidad '; DROP TABLE usuarios; SELECT * FROM
datos WHERE nombre LIKE '% B1 A◆adir al carrito
```

Fig. 3. A single document file showing a paragraph consisting of ten extracted sentences.

After comparing performance of all the eleven models, we retain the best performing model to be deployed as part of NLPIDS for real-time intrusion detection.

D. Intrusion Detection

The NLPIDS can be deployed at the network perimeter of the servers being protected from intrusions. The NLPIDS will capture all the incoming HTTP requests to the server using a traffic capture tool like WireShark and store the request details in a log file. Then the request details will be converted to natural language corpus and vector spaces. The best performing trained model, obtained from the training phase, will classify the vector spaces into benign and anomalous HTTP requests.

V. Experimental Results

This section presents an experiment using the proposed scheme and discusses the results.

A. Dataset

The dataset used for experimental validation of our proposed method is the HTTP DATASET CSIC 2010 web penetration and hacking dataset consisting of normal and anomalous HTTP requests [25]. The dataset contains the generated traffic targeted to an e-Commerce web application. The HTTP requests are labeled anomalous when the requests contain attacks such as SQL injection, buffer overflow, information gathering, files disclosure, CRLF injection, XSS, server side include, parameter tampering, etc. The dataset is generated automatically and contains 36,000 normal requests and more than 25,000 anomalous requests.

B. Experimental Setup

The proposed scheme was implemented in the Python programming language and using the machine learning library scikit-learn [26]. As noted before, the scheme also uses the gensim library. The experiments were executed on a PC with x64 Intel®CORE™i5-6600K CPU @ 3.50GHz, 8GB memory and running a 64-bit Ubuntu Linux operating system.

C. Data Preparation

The file containing the HTTP requests is processed to convert different relevant entries in the requests into natural language sentences. Then paragraphs containing 10 sentences are created and stored in documents. Collections of such documents create three corpora corresponding to three data files, such as normal traffic (training), normal traffic (test) and anomalous traffic (test). After that, the gensim tool is used to convert the documents into vectors using doc2vec model. We used the following parameters for the doc2vec model: 1) Dimensionality of the feature vectors = 100, 2) Window = 15, 3) Number of epochs = 30, and 4) Training Model = DBoW. The window is the maximum distance between the predicted word and context words used for prediction within a document.

TABLE I
EVALUATION METRICS VALUES FOR INDIVIDUAL AND ENSEMBLE MODELS USING THE TEST DATASET.

Models	F1-Score	Accuracy	Precision	Sensitivity	Specificity	FPR	ROC AUC	Elapsed Time (s)
LR	0.9937	0.9916	0.9979	0.9897	0.9957	0.0042	0.9996	1.15
NB	0.9855	0.9802	0.9719	0.9994	0.9411	0.0588	0.9917	0.25
NN	0.9995	0.9994	0.9995	0.9995	0.9991	0.0008	0.9992	0.59
DT	0.9736	0.9644	0.9692	0.9780	0.9367	0.0632	0.9907	3.83
SVM	0.9992	0.9990	0.9995	0.9989	0.9991	0.0008	0.9999	13.84
Ens_MV	0.9990	0.9987	0.9985	0.9995	0.9970	0.0029	N/A	2.14
Ens_LR	0.9735	0.9634	0.9483	1.0000	0.8884	0.1115	0.9999	0.21
Ens_NB	0.9989	0.9986	0.9996	0.9982	0.9992	0.0007	0.9989	0.19
Ens_NN	0.9996	0.9995	0.9996	0.9996	0.9992	0.0007	0.9999	0.29
Ens_DT	0.9996	0.9995	0.9996	0.9996	0.9992	0.0007	0.9999	0.19
Ens_SVM	0.9996	0.9995	0.9996	0.9996	0.9992	0.0007	0.9992	0.27

The dataset is now prepared to be used as input to any machine learning method.

D. Model Training

In this phase, the processed dataset was used to train our existing ensemble supervised ML framework [24] which consists of five individual classifiers and six ensemble classifiers. The models were trained using grid-search and the best values of the hyper-parameters given by grid-search were retained. The dataset was split into two subsets: 70% for training and 30% for testing. To avoid over-fitting and to obtain robust models, 10-fold cross validation over randomly divided training data was used. Then the test data was used for prediction and for measuring model performance.

E. Evaluation Metrics

Six metrics are used to evaluate the performance of our proposed method. *Accuracy* is the percentage of correct identification over total data instances. *Precision*, also known as the positive predictive value, represents how often the model correctly identifies an attack. *Sensitivity*, also known as the true positive rate (TPR), recall, or detection rate, indicates how many of the attacks the model does identify correctly. Sensitivity intuitively gives the ability of the classifier to find all the positive samples and the precision intuitively gives the ability of the classifier not to label as positive a sample that is negative. *Specificity*, also known as the true negative rate, measures the proportion of actual negatives, i.e., non-attacks, that are correctly identified as such. *F1-score*, also known as F-measure, provides the harmonic average of precision and sensitivity.

The six metrics are formulated as follows [27]:

1) Accuracy $= (TP + TN)/TotalPopulation$
2) Precision $= TP/(FP + TP)$
3) Sensitivity $= TP/(FN + TP)$
4) False Positive Rate (FPR) $= FP/(FP + TN)$
5) Specificity $= TN/(FP + TN)$ or (1 - FPR)
6) F1-score $= 2TP/(2TN + FP + FN)$

where 1) True positive (TP): when the model correctly identifies an attack, 2) True negative (TN): when it correctly identifies a normal or non-attack, 3) False positive (FP): when a non-attack is incorrectly identified as an attack, and 4) False negative (FN): when an attack is incorrectly identified as a non-attack.

In addition, the receiver operating characteristic (ROC) curves are plotted to demonstrate the detection performance of different models over all possible thresholds. The run times (i.e., elapsed times) for training the models were measured for comparing the speed of different training models.

F. Discussion of Results

In this section, we present and discuss results for applying the proposed scheme on the dataset in terms of the evaluation

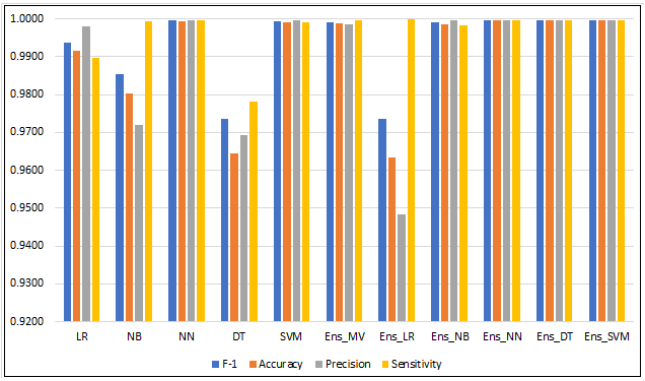

Fig. 4. Comparison of the individual and ensemble models using F1-score, accuracy, precision and sensitivity.

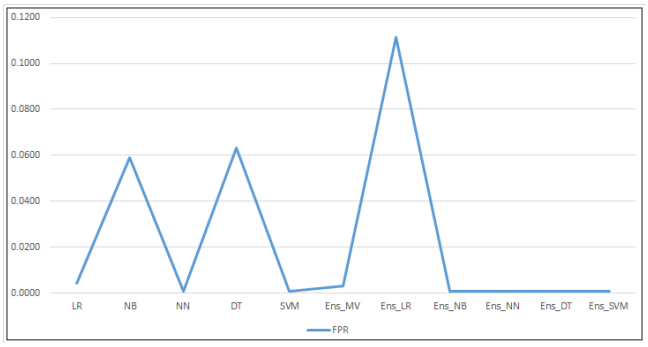

Fig. 5. Comparison of the individual and ensemble models using FPR.

metrics.

Table I shows the evaluation metrics values for different classifier models and the corresponding graph in Fig. 4 depicts the values for F1-score, accuracy, precision and sensitivity. Fig. 5 From the table we find that the accuracy values for the models vary from 96.34% for LR-based ensemble classifier, Ens_LR, to 99.95% for three ensemble models, Ens_NN, Ens_DT, and Ens_SVM. The best precision value of 99.96% is given by four ensemble models, Ens_NB, Ens_NN, Ens_DT, and Ens_SVM.

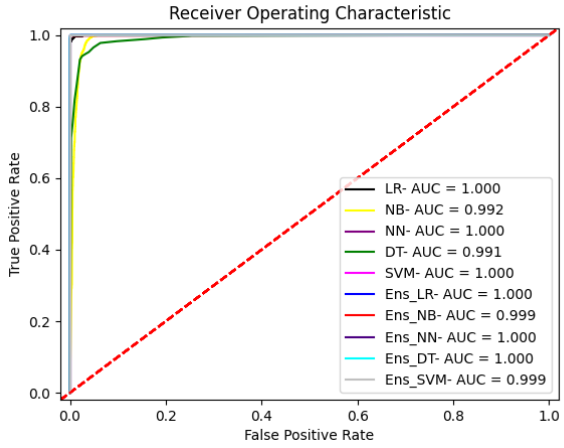

Fig. 6. ROC curves of the individual and ensemble models.

However, in a classification problem where the goal is to detect the minor class occurrences, the most important metrics are the sensitivity or recall which, in this case, measures the proportion of actual "anomalous" paragraphs that are correctly identified as such; the specificity which measures the proportion of actual "normal" paragraphs that are correctly identified as such; and the F1-score which is the weighted average of precision and recall. Considering these, we find Ens_NN, Ens_DT, and Ens_SVM are the best performing models with highest values of sensitivity, specificity, and F1-score. These models will detect almost 100% of the anomalous data, i.e., attacks, and will almost never raise false alarms. Fig. 6 shows the ROC curves of all the models.

VI. Conclusion

Natural language processing (NLP) is one of most powerful tools used in areas like speech recognition, sentiment analysis, question answering, anomaly detection, etc. In this paper, we implemented an NLP and ensemble machine learning based intrusion detection system to detect anomalous traffic from the traffic flow. We extracted relevant information from HTTP requests as natural language sentences, converted the sentences to vector space using NLP tools, and finally used ensemble machine learning models to classify normal and anomalous traffic. We experimented with HTTP DATASET CSIC 2010 to evaluate the performance of the proposed NLPIDS. Results of this experiment show that our NLPIDS outperformed other methods in the literature, as shown in Section II, in terms of detection rate (99.96%), false alarm rate (0.07%) and F1-score (99.96%).

As future work, we plan to test the efficacy of the method using various other datasets. We would also like to deploy the NLPIDS in real network to monitor [28] the network and measure NLPIDS's effectiveness in detecting anomalous traffic.

References

[1] M. Jodavi, M. Abadi, and E. Parhizkar, "Dbdhunter: an ensemble-based anomaly detection approach to detect drive-by download attacks," in *2015 5th International Conference on Computer and Knowledge Engineering (ICCKE)*, pp. 273–278, IEEE, 2015.

[2] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection," *17th USENIX Security Symposium*, 2008.

[3] C. T. Giménez, A. P. Villegas, and G. Á. Marañón, "Http data set csic 2010," *Information Security Institute of CSIC (Spanish Research National Council)*, 2010.

[4] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a dynamic reputation system for DNS," in *USENIX security symposium*, pp. 273–290, 2010.

[5] C. Kruegel and G. Vigna, "Anomaly detection of web-based attacks," in *Proceedings of the 10th ACM conference on Computer and communications security*, pp. 251–261, 2003.

[6] T. Peng, I. Harris, and Y. Sawa, "Detecting phishing attacks using natural language processing and machine learning," in *2018 ieee 12th international conference on semantic computing (ICSC)*, pp. 300–301, IEEE, 2018.

[7] C. Bertero, M. Roy, C. Sauvanaud, and G. Trédan, "Experience report: Log mining using natural language processing and application to anomaly detection," in *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 351–360, IEEE, 2017.

[8] A. Stone, "Natural-language processing for intrusion detection," *Computer*, vol. 40, no. 12, pp. 103–105, 2007.

[9] S. Das, A. M. Mahfouz, and S. Shiva, "A stealth migration approach to moving target defense in cloud computing," in *Proceedings of the Future Technologies Conference*, pp. 394–410, Springer, 2019.

[10] M. C. Belavagi and B. Muniyal, "Performance evaluation of supervised machine learning algorithms for intrusion detection," *Procedia Computer Science*, vol. 89, no. 2016, pp. 117–123, 2016.

[11] M. Almseidin, M. Alzubi, S. Kovacs, and M. Alkasassbeh, "Evaluation of machine learning algorithms for intrusion detection system," in *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, pp. 000277–000282, IEEE, 2017.

[12] S. Das, D. Venugopal, and S. Shiva, "A holistic approach for detecting ddos attacks by using ensemble unsupervised machine learning," in *Future of Information and Communication Conference*, pp. 721–738, Springer, 2020.

[13] S. Das, D. Venugopal, S. Shiva, and F. T. Sheldon, "Empirical evaluation of the ensemble framework for feature selection in ddos attack," in *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, pp. 56–61, IEEE, 2020.

[14] J. Lilleberg, Y. Zhu, and Y. Zhang, "Support vector machines and word2vec for text classification with semantic features," in *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, pp. 136–140, IEEE, 2015.

[15] E. Min, J. Long, Q. Liu, J. Cui, and W. Chen, "TR-IDS: anomaly-based intrusion detection through text-convolutional neural network and random forest," *Security and Communication Networks*, vol. 2018, 2018.

[16] M. Mimura and H. Tanaka, "Reading network packets as a natural language for intrusion detection," in *International Conference on Information Security and Cryptology*, pp. 339–350, Springer, 2017.

[17] J. Li, H. Zhang, and Z. Wei, "The weighted word2vec paragraph vectors for anomaly detection over HTTP traffic," *IEEE Access*, vol. 8, pp. 141787–141798, 2020.

[18] S. Althubiti, X. Yuan, and A. Esterline, "Analyzing HTTP requests for web intrusion detection," in *KSU Proceedings on Cybersecurity Education, Research and Practice*, vol. 2, 2017.

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[20] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, pp. 1188–1196, 2014.

[21] R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (Valletta, Malta), pp. 45–50, ELRA, May 2010.

[22] T. G. Dietterich, "Ensemble methods in machine learning," in *International Workshop on Multiple Classifier Systems*, pp. 1–15, Springer, 2000.

[23] R. Polikar, "Ensemble learning," in *Ensemble Machine Learning*, pp. 1–34, Springer, 2012.

[24] S. Das, A. M. Mahfouz, D. Venugopal, and S. Shiva, "DDoS intrusion detection through machine learning ensemble," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 471–477, IEEE, 2019.

[25] C. Torrano-Giménez, A. Perez-Villegas, and G. Alvarez Maranón, "An anomaly-based approach for intrusion detection in web traffic," *Journal of Information Assurance and Security*, vol. 5, no. 4, p. 446–454, 2010.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[27] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009.

[28] S. Das and S. Shiva, "Corum: collaborative runtime monitor framework for application security," in *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, pp. 201–206, IEEE, 2018.