International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 3, June 2020, pp. 2734~2741
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i3.pp2734-2741

A predictive model for network intrusion detection using stacking approach

Smitha Rajagopal, Poornima Panduranga Kundapur, Hareesha. K. S.
Department of Computer Applications, Manipal Institute of Technology,
Manipal Academy of Higher Education, India

Article Info

Article history:

Received Jul 19, 2019
Revised Oct 21, 2019
Accepted Nov 29, 2019

Keywords:

Graphlab create
Network intrusion detection
SFrames
Stacking
UGR'16
UNSW NB-15

ABSTRACT

Due to the emerging technological advances, cyber-attacks continue to hamper information systems. The changing dimensionality of cyber threat landscape compel security experts to devise novel approaches to address the problem of network intrusion detection. Machine learning algorithms are extensively used to detect intrusions by dint of their remarkable predictive power. This work presents an ensemble approach for network intrusion detection using a concept called Stacking. As per the popular no free lunch theorem of machine learning, employing single classifier for a problem at hand may not be ideal to achieve generalization. Therefore, the proposed work on network intrusion detection emphasizes upon a combinative approach to improve performance. A robust processing paradigm called Graphlab Create, capable of upholding massive data has been used to implement the proposed methodology. Two benchmark datasets like UNSW NB-15 and UGR' 16 datasets are considered to demonstrate the validity of predictions. Empirical investigation has illustrated that the performance of the proposed approach has been reasonably good. The contribution of the proposed approach lies in its finesse to generate fewer misclassifications pertaining to various attack vectors considered in the study.

Corresponding Author:

Hareesha. K. S,
Department of Computer Applications,
Manipal Institute of Technology, Manipal Academy of Higher Education,
Manipal, India.
Email: hareesh.ks@manipal.edu

1. INTRODUCTION AND RELATED WORK

Computer Security is one such research area that has garnered lot of attention in modern era due to the increased occurrence of cyber-attacks [1]. Network intrusion detection systems (NIDS) are extensively investigated in the literature to protect the seemingly vulnerable networks from external and internal intruders [2]. NIDSs have been in use since 1980's after Dorothy Denning [3] delineated that intrusion detection systems are critical to maintain the confidentiality, integrity and availability of computer resources [4, 5]. Prolific approaches exist in the field of network intrusion detection that have met with different scales of success. Some significant research efforts are recapitulated in this section. A hybrid model was devised in [6] to choose the optimal subset of features using Gini index and gradient boosted decision tree was used as a classifier. Another algorithm called particle swarm optimization was used to enhance the performance of the classifier by fine tuning the parameters. As discussed in [7], heterogeneous classification ensemble was employed to detect Distributed Denial of Service (DDOS) attacks. Singular Value Decomposition (SVD) was used to formulate the model that resulted in good attack detection rate.

A hybrid machine learning approach was designed to recognize zero-day attacks in Supervisory Control and Data Acquisition (SCADA) networks [8]. In order to achieve better results, filter-based feature selection approach was used to elicit appropriate features. This model, built using a combination of J48 and BayesNet classifiers was competent enough and exhibited promising performance considering an industrial

control system dataset. An ensemble learning technique was put forth in [9]. The dimensionality of the dataset was reduced using Principal Component Analysis (PCA) and the classification outcome was enhanced using a fusion of classifiers namely logistic regression, neural networks and decision trees through a weighted majority voting strategy. Typically, such ensemble approaches are implemented in machine learning research because a single classifier cannot excel in distinguishing all the attack types and the trade-off should be balanced prudently by introducing different classifiers in order to augment the overall performance. The research endeavor explained in [10] elaborated on the combination of Particle Swarm Optimization and Fast Learning Network (PSO-FLN) to address the problem of network intrusion detection. The model, as described in [10] was compared against some meta-heuristic algorithms to test its proficiency. Results were indicative of the fact that the integrated approach performed considerably well despite varying the number of neurons in the hidden layer. In order to process data on a massive scale, powerful machine learning platforms are required. The study of network intrusion detection involves large scale data analysis. The workflows created through machine learning platforms should be capable enough to hold enormous network instances and should not relent. Owing to the needs of proliferating data, Graphlab Create was selected as the processing platform. As mentioned in [11], Graphlab Create has superior capabilities than existing Python packages like Pandas in processing terabytes of data at interactive speeds. In recent scenario, researchers are using Big data processing paradigms for network intrusion detection to generate reliable predictions. Such persuasive paradigms when used certainly help in achieving faster predictions [12]. The problem of network intrusion detection becomes computationally complex as and when classifiers ingest humongous data. As explained further in [13], robust computing environments help towards cost-effective classification. Therefore, authors in [13] used Hadoop based parallel binary bat algorithm to extract the prominent features and applied Naive Bayes to classify the network instances of KDD cup 99 dataset. Upon selecting only 24 features, the technique proposed in [13] could improve attack detection rate of Probe and Remote2Local (R2L) types in a coherent manner. Another powerful computing paradigm for analyzing Big data is Apache Spark that is being considered lately by researchers to advance the study of network intrusion detection. As described in [12], Apache Spark, a Big data platform was considered to investigate network data. ChiSqSelector was used for feature selection. The classification outcome of Chi-SVM and Chi-Logistic regression was compared and eventually Chi-SVM model on Spark produced better results. Authors in [14] contrived a Big data framework using various machine learning algorithms on Apache Spark by considering synchrophasor dataset. The overall inference, derived from the study was that Apache Spark framework could decrease the processing time considerably. Furthermore, as mentioned in [14], multiclass classification task should be also initiated and subsequently, time taken for specific predictions can be comprehended.

As elaborated in [15], there is an immediate need to propose efficient intrusion detection frameworks based on real-time Big Data processing. There is also an on-going requirement to offer meaningful research directions for cloud based NIDS as explained in [16]. However, some challenges are associated with respect to Big data classification of network traffic like data visualization and data uncertainty as enumerated by Suthaharan [17]. A research endeavor was undertaken to detect cyber-targeted attacks based on Big Data [18]. This approach proposed by Kim et al. [18] used MapReduce to analyze anomalous behavior from different sources.

The proposed approach elucidated in this article has used the notion of stacking to build a predictive model, capable of generating decisive predictions by considering two datasets namely UNSW NB-15 and UGR' 16. The idea behind choosing these two datasets for experimentation is due to their contrasting nature i.e., UNSW NB-15 [19, 20] is a dataset developed through emulated network traffic whereas UGR'16 [21] was developed by considering cyclostationary evolution of network traffic. Additionally, UNSW NB-15 and UGR'16 are packet-based and flow-based datasets respectively [22]. UNSW NB-15 dataset was formulated by generating artificial traffic using IXIA perfect storm tool. This dataset is accessible in CSV, Bro and Pcap formats. Forty-seven features are present in this dataset with two class labels. Nine attack categories are found in this dataset and it is available in pre-determined train and test splits as delineated by Nour Moustafa and Jill Slay [19]. UGR'16 is a relatively new dataset that consists of 16,900 million flows. A significant feature of this dataset is that it is successful in capturing network traffic periodicity. Founders of UGR'16 dataset [21] mentioned explicitly that both background and attack traffic were captured systematically during the formation of this dataset. Network data required to develop this dataset was procured from tier-3 Internet Service Provider (ISP) for an ample duration of four months. A detailed explanation about the inception of UGR'16 dataset can be obtained from [21].

2. RESEARCH METHOD

Graphlab Create (GC), a Python based machine learning framework [23] was chosen for experimentation. The entire sequence of experimentation was conducted on ASUS VivoBook with Windows

10, 8GB RAM, an inbuilt 8th generation Intel core i5 processor and 64-bit architecture that facilitated Python 2.7. Classifiers used for the proposed study include logistic regression, K nearest neighbor, decision tree and random forest. A stacking approach devised using Graphlab Create has been proposed in this study. Random forest was used for meta classification. In order to maximize the performance of classifiers, key hyper-parameters were configured. Unless otherwise mentioned, default values of hyper-parameters were used to execute all the trials. The experimental strategy is explained in this section. Table 1 and Table 2 enumerate the number of network instances used for training and testing from UNSW NB-15 and UGR'16 respectively.

Table 1. Training and Testing instances applicable to UNSW NB-15 dataset

Class	Training Samples	Testing Samples
Normal	56000	37000
Analysis	2000	677
Backdoor	1746	583
Reconnaissance	10491	3496
Shellcode	1133	378
Worms	130	44
DOS	12264	4089
Fuzzers	18184	6062
Generic	40000	18871
Exploits	33393	11132
Total	1,75,341	82,332

Table 2. Training and Testing instances applicable to UGR'16 dataset

Type	Count
Blacklist	
Spam	
SSHscan	1 million flows in each partition of the dataset with 500,000 normal flows used for training & testing
UDPscan	
DOS	
DDOS	
Scan	

The usage of Graphlab Create became essential for the proposed study owing to the presence of numerous instances pertaining to UGR'16 dataset even though UNSW NB-15 dataset consisted of comparatively fewer instances. The concept of stacking is explained below through a series of steps and depicted in Figure 1.

- Divide the training data into k folds
- Level 0 classifier is built for k-1 parts and predictions are obtained for each kth segment.
- The same procedure is repeated for all level 0 classifiers involved in the study.
- Meta classifier is applied for test data
- The final predictions are made by the meta classifier using the outputs generated by level 0 classifier

2.1. Feature importance

In order to select the salient features from UNSW NB-15 dataset, Permutation Feature Importance (PFI) has been used in the proposed study. The concept of PFI was introduced by Breiman in 2001 [24]. A model-specific version of the same concept was put forth in [25]. As explained by Breiman [24] in his seminal work, a feature can be considered important if and only if shuffling its values result in an increase of model's error, which suggests that the model depended on a specific feature to form prediction. On the contrary, a feature is insignificant if changing its value does not impact the model's performance, thereby no change is visible in the model's error. The feature importance scores were calculated using permutation_importance function available in Scikitlearn. n_repeat is a parameter that indicates the number of times a specific feature needs to be permuted and the default value was set to 5. Trials were conducted by selecting the top 14, 16 and 18 features respectively. Results indicated that the accuracy was the highest when 16 features were selected from 47 features, that eventually formed the salient set as shown in Table 3. Firstly, the model was trained using Out-of-Bag samples (OOB) set and accuracy was recorded. The values of features were re-structured and the resulting accuracy was compared against the previously obtained accuracy scores. A remarkable advantage of using Graphlab Create is that it offers SFrames, a component capable of storing data efficiently on the server side. As mentioned in [26] data when stored on SFrames scale better because it is not limited by RAM [27]. Since the data on SFrames is stored on persistent storage, memory is not a constraint for storing and processing mammoth data of varying complexities.

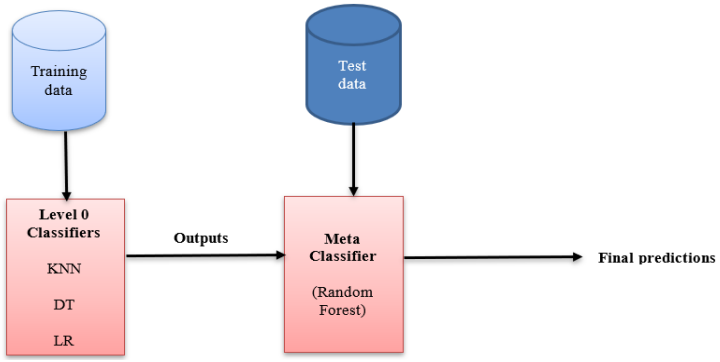

Figure 1. Stacking approach devised on graphlab create (GC)

Table 3. Salient features of UNSW NB-15 dataset

Sttl	ct_state_ttl	dload	dmean	dbytes	dpkts	ct_dst_sport_ltm	dloss
Sloss	swin	dwin	spkts	ct_src_dport_ltm	sbytes	stcpb	dtcpb

On the UGR'16 front, one million flows were given as input to the stacking framework to learn and produce optimal predictions. A fairly good enough performance was exhibited by the model. Unlike UNSW NB-15 dataset, no feature importance scores were extracted for UGR'16 dataset since the latter has fewer features. Timestamp, duration, source IP address, destination port, protocol, type of service, packets exchanged during flow and the number of bytes are the features from UGR'16 included for classification task. The outcome of any classification task relies largely on three critical factors [28, 29]. 1) Feature selection 2) Appropriate tuning of hyper-parameters and 3) Performance of state-of-the art classifier

2.2. Hyper-parameters

Hyper-parameters contribute immensely towards the performance of machine learning models [28] and are known as configuration knobs. As discussed in [29], for the same training set, an algorithm may perform differently for distinct values of hyper-parameters. As a matter of fact, since hyper-parameters are deemed confidential, authors in [29] propsed an attack framework capable of stealing hyper-parameters. Quite often in machine learning research, hyper-parameters eventually become trade secrets as they heavily influence machine learning outcome. Owing to the criticality of hyper-parameters, the following values were assigned to enhance the performance of stacking ensemble as explained below.

2.2.1. K nearest neighbor (KNN)

Being a simple classifier to implement, KNN attempts to classify a data point by keenly observing its neighbors. Graphlab Create allows the data scientist to explicitly mention the value of max_neighbours that refers to the utmost number of neighbors to be considered for each new data point. In the case of UNSW NB-15, the value was set as 5 and for UGR'16 dataset, a value 10 was assigned to max_neighbours.

2.2.2. Decision tree (DT)

Tree based classifiers are widely used in many applications due to their classification competence by conducting recursive partitioning. Being a white-box machine learning algorithm [30], decision tree is touted to be good for promoting better classification accuracy. Although in some cases, classifier works reasonably well using only default values of hyper-parameters; varying the default values is primarily employed by machine learning practitioners to inspect the wavering performance of classifiers. Max_depth indicates the longest path starting from the root to leaf node. Sometimes, large values when assigned to Max_depth result in overfitting since trees tend to grow excessively. In the case of UNSW NB-15, Max_depth was set to 6 and 7 was the Max_depth for UGR'16 to regulate overfitting and obtain a legitimate estimation of the classifier. Class_weights denote the weights corresponding to each class. Auto was used for both datasets which suggests that the class weight is inversely proportional to the samples found in training set.

2.2.3. Logistic regression (LR)

LR is one of the go-to algorithms extensively used for binary classification. In the proposed study, LR has performed considerably well for multiclass classification too when combined with other classifiers. 0.01 was assigned as penalty while conducting trials using both datasets. This was done so that bias variance trade-off could be balanced. L- Broyden–Fletcher–Goldfarb–Shanno (BFGS) optimization algorithm was

employed for UNSW NB-15 dataset while the option auto was selected for UGR'16. Auto option is usually selected so that optimal solver can be suggested by the processing system (Graphlab Create) and no explicit mention is made by the programmer.

2.2.4. Random forest (RF)

Being one of the versatile algorithm for classification, ensembling technique is innately used by RF to optimize its performance. The proposed approach designated RF to be the meta classifier for spawning final predictions. Max_iterations in the case of both datasets were set to 100 in order to avoid overfitting. The reason to choose random forest as the meta classifier can be attributed to its ability to decrease bias. Row_subsample and Column_subsample are two important parameters that help in randomly splitting the samples row wise and column wise so that the possibility of overfitting could be curtailed. 0.5 was the value designated for row and column subsamples pertaining to both datasets.

3. RESULTS

The proposed classification ensemble has been evaluated in terms of some standard performance metrics as defined below from Equations (1) to (5).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F\ 1-score = 2\frac{Precision\ X\ Recall}{Precision+Recall} \quad (4)$$

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP+TN} \quad (5)$$

Evaluating the classification model only on the basis of accuracy may not suffice. Therefore, the remaining metrics serve as supplementary. The aim of any intrusion detection system is to maximize attack detection rate and mitigate false alarms. Overall accuracy of a classification model denotes the percentage of correct predictions out of the total number of samples. Precision depicts the prediction capability of the model by considering false positives whereas false negatives are included by recall. F1-score is another helpful metric that highlights the weighted average of precision and recall. It is a common practice in machine learning research to illustrate the performance of a classifier by considering various metrics since each metric holds its own relevance.

3.1. Results obtained for UNSW NB-15 dataset

Each instance in UNSW NB-15 dataset has 2 labels attached to it. The primary label is used to determine whether a particular network instance is an attack or normal. The secondary label is also equally important to decide the attack type of every single network instance. Therefore, for UNSW NB-15 dataset, both binary and multiclass classification tasks become mandatory. Table 4 outlines the performance of the stacking ensemble in view of binary classification.

The above table is an illustration of the fact that the ensemble model has been successful, to a large extent in identifying normal and attack samples quite well. The precision and recall metrics that consider false positives and false negatives respectively, have been able to produce a fairly good enough score. The false positive rate generated by the ensemble is also not seemingly high and is an indication that the model has performed quite appropriately. As an affirmation to the model's performance, it is logical to inspect its performance pertaining to each attack type i.e., its capability to discern between various attack types in order to fathom its predictive power. Matrix 1 represents the dexterity of the model to distinguish between various attack types by putting forth the results of multiclass classification pertaining to UNSW NB-15 dataset. Typically, a confusion matrix represents actual versus predicted classification of a model. Nine attack categories namely, Normal (N), Reconnaissance (R), Backdoor (B), Denial of Service (D), Exploits (E), Analysis (A), Fuzzers (F), Worms (W), Shellcode (S) and Generic (G) were considered for evaluation and the following results were obtained as shown in Table 5.

Table 4. Results obtained for binary classification

TP	TN	FN	FP	Accuracy	Precision	Recall	F1-score	FPR
42316	35346	3016	1654	0.943	0.962	0.933	0.948	4.4%

Table 5. Matrix 1 of multiclass classification results obtained for UNSW NB-15 dataset

Index	A	B	D	E	F	G	N	R	S	W	Recall (%)
Analysis	55	0	83	521	14	0	4	0	0	0	8.1
Backdoor	0	57	44	462	16	0	2	2	0	0	9.7
DOS	0	1	2227	1745	21	5	71	16	2	1	54.4
Exploits	0	0	926	10070	37	1	77	15	6	0	90.45
Fuzzers	0	0	167	1057	4134	3	693	3	5	0	68.19
Generic	0	0	30	39	5	18768	22	1	6	0	99.45
Normal	0	0	16	41	134	14	36759	9	27	0	99.34
Reconnaissance	0	0	126	223	7	0	19	3119	2	0	89.21
Shellcode	0	0	8	7	0	6	23	9	325	0	85.97
Worms	0	0	0	3	0	2	0	0	0	39	88.63
Precision (%)	100%	98.27	61.4	71	94.64	99.83	97.58	98.26	87.13	97.5	

Recall and Precision scores are the two pivotal evaluation parameters to comprehend the performance of any multiclass classification model. The analysis of the results pertaining to multiclass classification task indicates that the misclassification rate corresponding to Analysis and Backdoor attack types is on the higher side. The model was not efficient in classifying the instances of Analysis and Backdoor aptly. The model wrongly classified many Analysis samples as Exploits. Additionally, majority of the samples pertaining to Backdoor attack type were incorrectly classified as Exploits. Barring these two attack types, the model performed exceptionally well in distinguishing Generic and Normal type samples because the recall and precision scores are consistent with respect to these two categories. The proposed model exhibited a favorable performance by classifying the samples of Exploit attack type fairly well. Attack types like Reconnaissance, Shellcode and Worms were also detected reasonably well by the model. It is worthwhile to note that the number of testing samples pertaining to Shellcode and Worms considered for evaluation were comparatively fewer than other attack types. However, the model exhibited a decent performance by learning minority samples also adeptly.

On the other hand, precision refers to the number of samples predicted by the model as belonging to a certain type, whereas in reality it does not. This characteristic of the model becomes extremely crucial to decide whether the model produces large number of false positives or not since the false positives are taken into account by the precision metric. The proposed model has produced a commendable precision score for analysis attack type because none of the other samples from any other category have been misclassified as Analysis. Another important finding about the proposed model's functionality is that only one sample belonging to Denial of Service was wrongly classified as Backdoor that obviously enhanced the precision score for Backdoor attack type. Very few samples from other categories were interpreted by the proposed model to be Normal, Reconnaissance or Generic. Hence, the precision scores of these three categories are quite satisfactory. The precision scores of Denial of Service and Exploits are average due to the model's ability to classify quite a few samples from other categories as belonging to these two attack types. Several samples belonging to Exploits were interpreted by the model as Denial of Service that apparently decreased the precision score of Exploits. Similarly, majority of the samples pertaining to Denial of service were predicted as Exploits and Fuzzers that reduced the precision score of Exploits to a considerable extent. Attack types like Shellcode and Worms were predicted by the model quite pertinently because only few instances belonging to other categories were construed by the proposed model as Shellcode and Worms.

3.2. Results obtained for UGR'16 dataset

UGR'16 dataset has millions of flows that needed a comprehensive investigation. In order to validate the performance of the stacking ensemble, 1,000,000 flows were used to train and test the model. Since the processing paradigm considered in the proposed study is quite reliable, the experiments could be conducted in an efficient manner. It can be noticed that only Denial of Service attack is common to both UNSW NB-15 and UGR'16 datasets but it is worthwhile to note that both datasets were developed in diverse network traffic environments using different test beds. The results are explained clearly to affirm that the proposed model is vigorous enough to differentiate between the various attacks types of UGR'16 dataset. For the purpose of experimentation, seven different partitions of the datasets were considered, each comprising of 1 million samples. Upon experimentation, the following results were obtained as illustrated below from Table 6 to Table 12.

Table 6. Results pertaining to blacklist attack

TP	TN	FN	FP	Accuracy	Precision	Recall	F1-score	FPR
489,100	460,021	12,555	38,324	0.94	0.927	0.97	0.948	7.7%

Table 7. Results pertaining to spam attack

TP	TN	FN	FP	Accuracy	Precision	Recall	F1-score	FPR
473,005	456,506	45,219	25,270	0.929	0.949	0.91	0.93	5.24%

Table 8. Results pertaining to SSHScan attack

TP	TN	FN	FP	Accuracy	Precision	Recall	F1-score	FPR
469,556	429,008	87,144	14,292	0.89	0.97	0.84	0.9	3.22%

Table 9. Results pertaining to DDOS attack

TP	TN	FN	FP	Accuracy	Precision	Recall	F1-score	FPR
402,616	488,925	71,156	37303	0.89	0.91	0.84	0.87	7%

Table 10. Results pertaining to DOS attack

TP	TN	FN	FP	Accuracy	Precision	Recall	F1-score	FPR
483,510	442,397	67,574	6519	0.92	0.98	0.88	0.92	1.45

Table 11. Results pertaining to scan attack

TP	TN	FN	FP	Accuracy	Precision	Recall	F1-score	FPR
486,925	472,413	19,145	21,517	0.959	0.957	0.962	0.959	4.35%

Table 12. Results pertaining to UDPScan attack

TP	TN	FN	FP	Accuracy	Precision	Recall	F1-score	FPR
490,003	456,809	17,154	36,034	0.94	0.93	0.96	0.944	7.3%

The proposed model has recorded the highest recall percentage for Blacklist attack type i.e., the model exhibited noteworthy performance in detecting Blacklist attack instances quite efficiently. Additionally, the proposed model has achieved the highest precision score with respect to DOS attack type. The effectiveness of the proposed model in identifying UDPScan attack samples aptly has been consistently good. Apart from the above-mentioned findings of the work, it can be also stated that precision score of all attack types found in UGR'16 dataset are in the range 0.91 to 0.98 whereas recall scores range from 0.84 to 0.97. Quite a few SSHscan instances were inappropriately identified as normal by the proposed model. There is still some scope to improve the attack detection capability of the model by enhancing the recall score of DDOS. Besides, the predictive model has also demonstrated an impressive feat in correctly detecting Scan attack types with a decent enough score pertaining to precision and recall. The least false positive rate recorded by the proposed model is 1.45% with respect to DOS attack type whereas the highest false positive rate is with respect to Blacklist attack type i.e., 7.7%. This is due to the fact that 38,324 samples belonging to normal were misclassified as Blacklist. Considering the presence of 500,000 normal samples, the proportion of misclassification is quite less.

4. CONCLUSION

In the proposed work, an ensemble approach based on stacking has been presented to obtain reliable predictions by combining different algorithms. In order to substantiate the proposed design, two disparate datasets were considered. Results have indicated that the performance of the stacking ensemble has been considerably good. Most importantly, a robust processing paradigm called Graphlab Create (GC) was used to execute trials involving numerous instances. The choice of the processing paradigm becomes important because network intrusion detection intrinsically involves Big data analytics due to the size and complexity of network data involved. Any processing paradigm, considered for analysis should not succumb but should be time effective in generating alerts as and when malicious packets penetrate into the network. Owing to such considerations and relevance, modern datasets comprising recently compiled attack types from UNSW NB-15 and UGR'16 datasets were employed. Performance and scalability are the two major parameters while addressing the problem of network intrusion detection. The proposed work, in accordance with the aforementioned parameters, offers a slightly different perspective to network intrusion detection as explained in this article.

REFERENCES

[1] Carrasco, Rafael San Miguel, and Miguel-Angel Sicilia, "Unsupervised intrusion detection through skip-gram models of network behavior," *Computers & Security*, vol. 78, pp. 187-197, 2018.

[2] Hodo, Elike, Xavier Bellekens, Andrew Hamilton, Christos Tachtatzis, and Robert Atkinson, "Shallow and deep networks intrusion detection system: A taxonomy and survey," *arXiv preprint arXiv:1701.02145*, 2017.

[3] Zhang, Bing, Zhiyang Liu, Yanguo Jia, Jiadong Ren, and Xiaolin Zhao, "Network Intrusion Detection Method Based on PCA and Bayes Algorithm," *Security and Communication Networks*, 2018.

[4] Chellam, Aditya, L. Ramanathan, and S. Ramani, "Intrusion Detection in Computer Networks using Lazy Learning Algorithm," *Procedia computer science*, vol. 132, pp. 928-936, 2018.

[5] Shams, Erfan A., Ahmet Rizaner, and Ali Hakan Ulusoy, "Trust aware support vector machine intrusion detection and prevention system in vehicular ad hoc networks," *Computers & Security*, vol. 78, pp. 245-254, 2018.

[6] Li, Longjie, Yang Yu, Shenshen Bai, Jianjun Cheng, and Xiaoyun Chen, "Towards Effective Network Intrusion Detection: A Hybrid Model Integrating Gini Index and GBDT with PSO," *Journal of Sensors*, 2018.

[7] Jia, Bin, Xiaohong Huang, Rujun Liu, and Yan Ma, "A DDoS attack detection method based on hybrid heterogeneous multiclassifier ensemble learning," *Journal of Electrical and Computer Engineering*, 2017.

[8] Ullah, Imtiaz, and Qusay H. Mahmoud, "A hybrid model for anomaly-based intrusion detection in SCADA networks," *In 2017 IEEE International Conference on Big Data (Big Data). IEEE*, pp. 2160-2167, 2017.

[9] Mirza, Ali H., "Computer network intrusion detection using various classifiers and ensemble learning," *In 2018 26th Signal Processing and Communications Applications Conference (SIU). IEEE*, pp. 1-4, 2018.

[10] Ali, Mohammed Hasan, *et al.*, "A new intrusion detection system based on fast learning network and particle swarm optimization," *IEEE Access*, vol. 6, pp. 20255-20261, 2018.

[11] Nasr, Mona, Shaaban, Essam, Hafez, Ahmed, "Building Sentiment analysis Model using Graphlab," *International Journal of Scientific and Engineering Research*, vol. 8, pp. 1155-1160, 2017.

[12] Othman, Suad Mohammed, *et al.*, "Intrusion detection model using machine learning algorithm on Big Data environment," *Journal of Big Data*, vol. 5, no. 1, pp. 34, 2018.

[13] Natesan, P., *et al.*, "Hadoop based parallel binary bat algorithm for network intrusion detection," *International Journal of Parallel Programming*, vol. 45, no. 5, pp. 1194-1213, 2017.

[14] Vimalkumar, K., and N. Radhika, "A big data framework for intrusion detection in smart grids using Apache Spark," *In 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, pp. 198-204, 2017.

[15] Habeeb, *et al.*, "Real-time big data processing for anomaly detection: A Survey," *International Journal of Information Management*, vol. 45, pp. 289-307, 2019.

[16] Keegan, Nathan, *et al.*, "A survey of cloud-based network intrusion detection analysis," *Human-centric Computing and Information Sciences*, vol. 6, no. 1, 2016.

[17] Shan Suthaharan, "Big data classification: Problems and challenges in network intrusion prediction with machine learning," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 4, pp. 70-73, 2014.

[18] Kim H., Kim J., Kim I., Chung T., "Behavior-based anomaly detection on Big Data," *The Proceedings of the 13th Australian Information Security Management Conference 2015*, Perth, pp. 73-80, 2015.

[19] Moustafa, Nour, and Jill Slay, "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Information Security Journal: A Global Perspective*, vol. 25, no. 1-3, pp. 18-31, 2016.

[20] Moustafa, Nour, and Jill Slay, "A hybrid feature selection for network intrusion detection systems: Central points," *arXiv preprint arXiv:1707.05505*, 2017.

[21] Macia-Fernandez, *et al.*, "UGR '16: A new dataset for the evaluation of cyclostationarity-based network IDSs," *Computers & Security*, vol. 73, pp. 411-424, 2018.

[22] Ring, Markus, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, and Andreas Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, 2019.

[23] Goldstein, Anat, Lior Fink, Amit Meitin, Shiran Bohadana, Oscar Lutenberg, and Gilad Ravid, "Applying machine learning on sensor data for irrigation recommendations: revealing the agronomist's tacit knowledge," *Precision agriculture*, vol. 19, no. 3, pp. 421-444, 2018.

[24] Breiman, Leo., "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5-32, 2001.

[25] Aaron Fisher, *et al.*, "All Models Are Wrong, but Many Are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously," *arXiv*, 2019.

[26] Malandrino, Francesco, Scott Kirkpatrick, and Danny Bickson, "What is LTE actually used for? An answer through multi-operator, crowd-sourced measurement," *arXiv preprint arXiv:1611.07782*, 2016.

[27] Reddy, Kompelly Harshavardhan, and Vikram Goyal, "Performance analysis of graph processing frameworks," PhD diss., 2015.

[28] Luo, Gang., "A review of automatic selection methods for machine learning algorithms and hyper-parameter values," *Network Modeling Analysis in Health Informatics and Bioinformatics*, vol. 5, no. 1, pp. 18, 2016.

[29] Wang, Binghui, and Neil Zhenqiang Gong, "Stealing hyperparameters in machine learning," *In 2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, pp. 36-52, 2018.

[30] Qin, Biao, Yuni Xia, and Fang Li, "DTU: a decision tree for uncertain data," In *Pacific-Asia conference on knowledge discovery and data mining*, Springer, Berlin, Heidelberg, pp. 4-15, 2009.