ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-7605-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICASSP39728.2021.9414532

VOTING-BASED ENSEMBLE MODEL FOR NETWORK ANOMALY DETECTION

*Tzu-Hsin Yang, Yu-Tai Lin, Chao-Lun Wu, Chih-Yu Wang**

Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan, R.O.C.
{tzuhsin, ytlin1993, allenwu, cywang}@citi.sinica.edu.tw

ABSTRACT

Network anomaly detection (NAD) aims to capture potential abnormal behaviors by observing traffic data over a period of time. In this work, we propose a machine learning framework based on XGBoost and deep neural networks to classify normal traffic and anomalous traffic. Data-driven feature engineering and post-processing are further proposed to improve the performance of the models. The experiment results suggest the proposed model can achieve 94% for F1 measure in the macro average of five labels on real-world traffic data.

Index Terms— Voting, XGBoost, Deep neural networks

1. INTRODUCTION

Cyber attacks that target the Internet users and the network are growing rapidly along with the growing demands on the Internet. Accordingly, network security and network anomaly detection (NAD) becomes a critical issue that needs to be tackle. Traditionally, NAD resort to signature-based (or rule-based) methodologies to detect anomalies[1]. However, such methodologies rely on manual approaches to detect anomalies patterns, which may not catch up with the rapid growth and evolution of cyber attacks recently.

Machine learning (ML) solutions are potentially perfect choices for NAD[2]. For instance, Duffield et al.[3] exploited correlations between packet and flow-level information via an ML approach. Gaddam et al.[4] presented a method combining K-means clustering and ID3 decision tree on several anomaly detection tasks. Chen et al.[5] utilized XGBoost[6] for DDOS detection in self defined networks. Shone et al.[7] constructed a deep autoencoder (NDAE) for unsupervised feature learning and using stacked NDAEs for intrusion detection. Ran et al.[8] provided a semi-supervised learning framework to learn from anomaly traffic. Yan et al.[9] designed an LSTM-based model with autoencoders for intrusion detection. Devan et al.[10] introduced an XGBoost-DNN model to detect cyber attacks. Ma et al.[11] combined reinforcement learning with SMOTE algorithm to address the imbalanced class problem. The previous works focus on model design. In contrast, in this paper, we argue that feature engineering and post-process are also critical for NAD.

As the participants of the ZYELL-NCTU NAD Challenge competition, we present an ensemble approach employing XGBoost and deep neural networks to detect network anomalies. According to the dataset analysis, we further present the important insights and the corresponding feature engineering and post-processing techniques. The full version of this paper can be found at our Github repo[12].

2. ATTACK ANALYSIS

We first analyze the training dataset to identify useful patterns. The attacks listed in the training dataset, which are also the target of this competition, are - **DDOS-Smurf**, **Probing-IP Sweep**, **Probing-Port Sweep**, and **Probing-Nmap Sweep**, which are popular attacks on the Internet. Firstly, We discovered that the Probing-IP sweep consists of an interesting pattern in the "dst IP" feature; most of them only differ from one another in the last octet but be the same in the first three octets. The behavior makes sense to us since the Probing-IP sweep scans hosts in the particular network. We also found that for DDOS-smurf attack, all the "dst IP" are ended with 255. That discovery also makes sense to us because DDOS-smurf is targeting broadcast addresses in order to enlarge the number of victims. Furthermore, we found that Probing-Nmap sweep's destination IP all starts with "172.24" in the first two octets. However, since the amount of data regarding the Probing-Nmap sweep is significantly smaller than other attacks in the dataset, further studies are needed to confirm whether this pattern would apply to general cases.

3. METHODOLOGY

3.1. Feature Engineering

We first introduce how we process features before modeling. We used raw features of "duration," "in (bytes)," "out (bytes)," and all "cnt" features. Plus, we transformed some features to make them meaningful for modeling.

When digging into data, we discovered that although time series of traffic data are significant for anomaly detection, the specific time is not important as both normal and anomaly

*This work was supported by the Ministry of Science and Technology under Grant MOST 108-2628-E-001-003-MY3, 109-3111-8-002-002-, and the Academia Sinica under Thematic Research Grant.

traffic can happen at any time. Therefore, we use a time-series relationship in post-processing.

For IPs, we originally assumed that they are useful features because, as mentioned in Section 2, the attack contains patterns that relate to IP addresses. However, this assumption holds only if the collected traffic in both training and testing datasets belong to the same network area. We have applied a Principal Component Analysis[13] (PCA) to visualize the distributions of the dataset to see if the IPs in the training set are similar to testing sets. It turned out that they are not similar, which suggests the assumption is invalid. Therefore, we remove features of IPs as inputs of the model. Regarding ports, although some applications or protocols correspond to a common port, ports can be a random number in many cases. Thus, we removed these features as well.

For the additional features, as mentioned in Section 2, each attack contains specific patterns. Thus, we craft additional features indicating these patterns to further boost the prediction performance. We list the most effective features:

- **inner_src, inner_dst**: We discovered that all the destination IPs of Probing-Nmap start with "172.24". Thus, we think the features of Inner IPs may help to detect Probing-Nmap. We consider the IP as inner IP if it belongs to the IPv4 address ranges reserved for a private network.
- **dst_ip_end_with_255**: Since we discovered that the destination IPs of DDOS-smurf data are all ended with 255, we added this binary feature in the input set. The experiment results in Section 4.3 show that the performance improves dramatically in predicting DDOS-smurf.
- **is_sweep**: We discovered that, for Probing-Port sweep and Probing-IP sweep, the destination IP often differs only in the last octet from the consecutive data with the same label in the neighboring. Therefore, we calculate the difference of "src" and "dst" between each row. If "src" is the same as the previous row and "dst" is only different in the last octet. Then we assign the value of "is_sweep" as 1. The experiment result shows this feature will improve the detection of Probing-IP sweep, Probing-Port sweep, and DDOS-smurf as well due to that DDOS-smurf usually attacks the same broadcast addresses.

3.2. Ensemble method

3.2.1. XGBoost Classifier

XGBoost[6] is a scalable boosting framework, which is widely used in both industry and academia. We chose this model as our main framework. To tackle the considered multi-classification problem, we use softmax as the learning objective. Softmax normalizes the outputs and gives a probability of each category $\hat{y}_i$.

The loss function of XGBoost is categorical cross-entropy loss shown in (1). C is the number of the category, which is 5 in our task. y_i is the ground truth of traffic. The value is 1 if traffic belongs to its corresponding category; otherwise, it is 0. $\hat{y}_i$ is the prediction probability of the proposed model.

$$Loss = -\sum_{i}^{C} y_i log(\hat{y}_i) \tag{1}$$

The XGBoost's learning parameters are shown below. The learning rate is 0.1. The maximum depth of a tree is 6. The minimum child weight is 1. L2 regularization is used on weights. We also use early stop for the validation and the test, with the round set as 10.

3.2.2. Deep Neural Networks

We also designed a deep neural network to detect anomalous traffic. There are 7 layers (ReLU-Dropout*3+Softmax) in total in our proposed neural network model. The structure of the proposed method contains three fully connected layers with ReLU[14] activation function. Each layer has 2048 neurons. After each layer, a dropout[15] layer with a dropout rate of 0.5 are added. The output layer has 5 softmax neurons that predict the probability of each traffic category. We use categorical cross entropy loss (1) for optimization, and the optimizer is Adam[16]. Details will be described in 3.2.3.

The learning and inference of neural networks follow the below procedure. Before training the proposed model, we normalized the preprocessed data that processed with feature engineering mentioned in Section 3.1. In the training process, we train the model with early stop patience 10, and the batch size is 128. The optimizer's parameters are the learning rate $\alpha = 0.001$, the exponential decay rates $\beta_1 = 0.9$, $\beta_2 = 0.999$ and the small constant avoiding dividing by zero $\epsilon = 10^{-7}$. In the inference process, we chose the category with the highest probability. Then we refine with results with post-processing.

3.2.3. Model Ensembling

Finally, we utilize ensemble methods to integrate prediction results from XGBoost and neural networks. We found out XGBoost and neural networks are capable of predicting different types of anomalous traffic. Specifically, we discovered that XGBoost made better predictions in most classes except Probing-IP sweep. In XGBoost evaluation, the F1 score of Probing-IP sweep was 80%, while in neural networks evaluation, the F1 score of Probing-IP sweep was 85%. Therefore, we use neural networks to predict Probing-IP sweep and use XGBoost to predict other kinds of attacks to generate the final result. The overall structure is shown in Fig. 1.

3.3. Voting Method In Post-Processing

By studying the given dataset, we discovered that the consecutive data with the same source IP should have the same

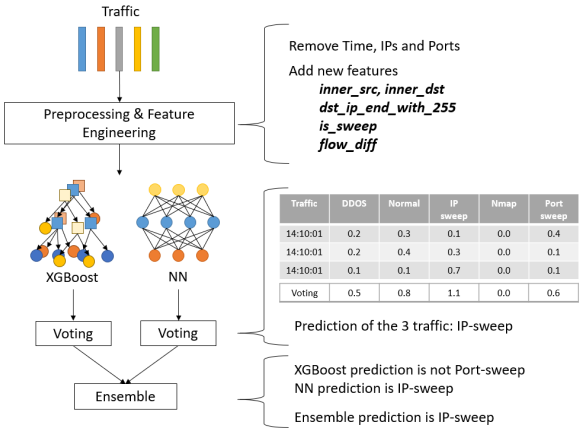

Fig. 1. The proposed model

label in a certain time range. Thus, we compose a voting post-processing method to adjust the prediction result. For starters, the model predicts the probabilities of each label with given data. Then, the voting procedure would sum all output probabilities within the same consecutive data and assign all the involved cases with the highest candidate label. In this way, we expect those cases which do not have an apparent tendency can be assigned the correct label with their neighboring rows.

Firstly, we propose a fixed time splitting method by defining the data are consecutive is whether they are in the same hour or minutes. That is, if two traffic happened in the same hour and with the same source IP, we view them as consecutive data. They would get involved in the same voting session. We have to note that we also discovered when we used different time periods to define consecutive data, the result would be slightly different for different attacks. For the best efforts, we compose a combined voting method that uses "same minute" for DDOS-smurf and "same hour" for all the others.

Besides the fixed time splitting, we further propose a dynamic time splitting method by defining two traffic in the same group if their source IPs are the same and the time difference between two traffic is less than one hour. The result shows the improvement in predicting testing data.

4. EXPERIMENTS

In this section, we will introduce the dataset and the preprocessing to conduct our experiments. Then, we will present our experimental results and discussions.

The data are time-series network traffic records captured by ZYELL's firewall[17]. Each traffic record is a network connection session and labeled as either normal or an attack type. By analyzing the training dataset, we discovered that the training dataset collected from different dates distribute considerably different. It shows the labels of data are imbalanced in different datasets. Also, we discover that data labeled Probing-Nmap sweep are too few for training models.

Plus, since most occurrences have more than one connection, each connection usually relates to its previous and subsequent cases. Therefore, when generating training and validation sets, we should not shuffle the raw dataset directly. Otherwise, the model may peek at traffic data which also appears in the validation set. In this training stage, we take the dataset on 12/03, and partial dataset on 12/10 as the training set, and take the dataset on 12/16 and the partial dataset on 12/10 as the validation set. We carefully split the data to make sure they have similar label distribution as well as not having data within the same occurrence. Moreover, to tackle the problem of data imbalance, we downsample some Normal data and keep Normal data being 1.3 times the size of data labeled Probing-IP sweep.

4.1. Performance of Ensemble Model

We first evaluate the performance of the proposed ensemble model. Note that in this experiment, we use the voting method with "minute" as the time range. Also, the evaluation matrices used in the experiments, i.e., F2 score and Criteria are defined in (2) and (3) where $\alpha = 0.3$, $totalcost$ = cost value calculated by the provided cost matrix, $maxcost$ = max cost $\times$ number of total entities, and $macro$ = macro averaging. Both F2 score and Criteria are provided by the contest committee.

$$F_2\ score = (1+2^2) \times \frac{Precision \times Recall}{2^2 \times Precision + Recall} \quad (2)$$

$$Criteria = \alpha \times (1 - \frac{log(total\ cost)}{log(max\ cost)}) + (1-\alpha) \times (macro\ F_\beta\ score) \quad (3)$$

Table 2 shows the performance of our proposed ensemble method across labels. We can see that performance on either DDOS-smurf or Probing-Nmap sweep is poor. We think the reason is that the data number of DDOS-smurf and Probing-Nmap sweep, especially for Probing-Nmap sweep, are far fewer than other labels. Also, we found that the model is easily confused DDOS-smurf for Normal data, i.e., the model easily miss-predict DDOS-smurf data to Normal.

Table 1. Performance Across Models

	XGBoost	NN	ensemble
F2 score	0.6666	0.6750	0.6958
Criteria	0.6408	0.6512	0.6631

Table 2. Ensemble Method Performance Across Labels

	Precision	Recall	F1 score	data counts
DDOS	0.0344	0.1007	0.0513	1033
Normal	0.9995	0.9931	0.9963	3164376
IP sweep	0.7592	0.9814	0.8561	59899
Nmap sweep	0.2290	0.8716	0.3626	109
Port sweep	0.9018	0.9337	0.9173	8644

4.2. Time range in voting method

Table 3 shows the results of applying our proposed voting method. Note that "voting combined" indicates the time range combination across different kinds of attacks, and "voting dynamic" indicates the dynamic time splitting approach.

Table 3. F1 score with different time splitting approach

	original	voting min	voting hour	voting combined	voting dynamic
DDOS	0.0546	0.0562	0.0115	0.0560	0.0134
Normal	0.9960	0.9961	0.9976	0.9976	0.9976
IP sweep	0.8081	0.8067	0.8952	0.8942	0.8942
Nmap sweep	0.3626	0.6620	0.4204	0.5901	0.6985
Port sweep	0.9173	0.9415	0.9549	0.9620	0.9582
Criteria	0.6408	0.6554	0.6595	0.6688	0.6857

As we can see in Table 3, the performance is significantly boosted in all kinds of attacks, especially for Probing-IP sweep, Probing-Port sweep, and Probing-Nmap sweep. Specifically, the XGBoost model originally was easily confusing Probing-IP sweep for Normal and Probing-Port sweep, and easily confusing Probing-Nmap sweep for Probing-IP sweep and Probing-Port sweep. However, the major portion of confusing data is the small portion of the consecutive data with the same source IP. Thus, by applying the voting post-processing procedure, the incorrect result would be correct and consequently improve the prediction accuracy. As for time range choosing, the result shows that Probing-IP sweep and Probing-Port sweep perform better in prediction when choosing hour as time range. Typically, Probing-IP sweep and Probing-Port sweep continue for a long time, usually exceed 1 min; thus, using "hour" as time range in Probing-IP and Probing-Port sweep would make the voting fairer. For "voting dynamic", although it performs worse in predicting DDOS-smurf, this could be resolved by adding "dst_ip_end_with_255" feature. On the other hand, "voting dynamic" perform better in predicting Probing-Nmap sweep.

4.3. Performance with additional feature

We then present the experiment results in feature engineering. The base model we use in this experiment is XGBoost with voting post-processing. Table 4 shows the F1 score for each label across different combination of feature engineering mentioned in Section 3.1. The result shows that performance further boosts in predicting DDOS-smurf, IP sweep, and Port sweep by adding additional features "is_sweep" and "dst_ip_end_with_255". For adding "is_sweep", the experiment result shows that the performance improves not only in predicting IP sweep and Port sweep but also in predicting Nmap sweep. We believe that the model could learn the sweeping behavior referring to this feature and, thus, could separate sweeping kinds of attacks from others well in prediction. For adding "dst_ip_end_with_255", the model could filter out DDOS-smurf attacks by this feature since it could strongly represent the DDOS-smurf behavior.

Table 4. F1 score for different additional feature

	voting	voting + is_sweep	voting + is_sweep + dst_end_with_ip_255
DDOS	0.0546	0.1677	0.9918
Normal	0.9960	0.9973	0.9973
IP sweep	0.8081	0.8755	0.8670
Nmap sweep	0.3626	0.6352	0.6352
Port sweep	0.9173	0.9201	0.9191
Criteria	0.6408	0.6873	0.8083

4.4. Testing Results

Table 5 shows the 8 testing results we get after submitting our models. In the previous experiments, the results show that our proposed ensemble method and additional features improve the performance. However, for the testing results, only adding "dst_ip_end_with_255" feature and use "voting_combined" get better. We believe the inconsistency comes from the overfitting to the training dataset. For 4th submissions, for instance, we find that the sweeping behavior may randomly choose a different network to attack. Thus, the model would be overfitting to training data by adding these two additional features. For 8th submission, it shows that the dynamic time splitting paradigm performs better than the original one.

Table 5. Testing Results

	Combination	Criteria
1st submit	xgboost + voting_min	0.5476
2nd submit	xgboost + voting_min + is_255	0.5876
3rd submit	ensemble + voting_min	0.4982
4th submit	xgboost + voting_min + is_sweep	0.5003
5th submit	xgboost + voting_combined	0.6060
6th submit	xgboost + voting_combined + is_255	0.6469
7th submit	ensemble + voting_combined + is_255	0.5887
8th submit	xgboost + voting_dynamic + is_255	0.6494

5. CONCLUSIONS

We propose an ensemble model with feature engineering and voting-based post-processing to identify various types of cyber attacks. We first analyze the characteristics of these attacks and develop feature engineering techniques based on our observations. Then, we design an ensemble model including XGBoost and deep neural networks for detecting attacks. We also use the voting method in post-processing which boosts the performance. The experiment results show the superior of the proposed models, while the testing results suggest overfitting is still a challenge in the NAD problem.

6. REFERENCES

[1] Claude Turner, Rolston Jeremiah, Dwight Richards, and Anthony Joseph, "A rule status monitoring algorithm for rule-based intrusion detection and prevention systems," *Procedia Computer Science*, vol. 95, pp. 361–368, 2016.

[2] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[3] N. Duffield, P. Haffner, B. Krishnamurthy, and H. Ringberg, "Rule-based anomaly detection on ip flows," in *IEEE INFOCOM*, 2009, pp. 424–432.

[4] S. R. Gaddam, V. V. Phoha, and K. S. Balagani, "K-means+id3: A novel method for supervised anomaly detection by cascading k-means clustering and id3 decision tree learning methods," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 3, pp. 345–354, 2007.

[5] Z. Chen, F. Jiang, Y. Cheng, X. Gu, W. Liu, and J. Peng, "Xgboost classifier for ddos attack detection and analysis in sdn-based cloud," in *IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2018, pp. 251–256.

[6] Tianqi Chen and Carlos Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd international conference on knowledge discovery and data mining (SIGKDD)*, 2016, pp. 785–794.

[7] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[8] J. Ran, Y. Ji, and B. Tang, "A semi-supervised learning approach to ieee 802.11 network anomaly detection," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, 2019, pp. 1–5.

[9] Y. Yan, L. Qi, J. Wang, Y. Lin, and L. Chen, "A network intrusion detection method based on stacked autoencoder and lstm," in *IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[10] P. Devan and N. Khare, "An efficient xgboost–dnn-based classification model for network intrusion detection system," *Neural Computing and Applications*, vol. 32, 2020.

[11] X. Ma and W. Shi, "Aesmote: Adversarial reinforcement learning with smote for anomaly detection," *IEEE Transactions on Network Science and Engineering*, 2020.

[12] "Snaclab nad challenge - ensemble method with voting post processing," `https://github.com/snaclab/NAD_Challenge/blob/main/paper/VOTING-BASED_ENSEMBLE_MODEL_FOR_NETWORK_ANOMALY_DETECTION.pdf`.

[13] Svante Wold, Kim Esbensen, and Paul Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1, pp. 37–52, 1987, Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

[14] Vinod Nair and Geoffrey E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML)*, 2010, p. 807–814.

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[16] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," 2017.

[17] Lei Chen, Shao-En Weng, Chu-Jun Peng, Hong-Han Shuai, and Wen-Huang Cheng, "Zyell-nctu nettraffic-1.0: A large-scale dataset for real-world network anomaly detection," *arXiv preprint arXiv:2103.05767*, 2021.