# NETWORK ANOMALY DETECTION USING ENSEMBLE LEARNING

*Shao-Wen Cheng, Shao-Peng Lai, Yun-Wen Lu, Yen-Chin Liao, Hsie-Chia Chang*

Department of Electronics Engineering and Institute of Electronics, National Chiao Tung University
Hsinchu, Taiwan 300, R.O.C.
Mail：shaowen.eic09g@nctu.edu.tw

## ABSTRACT

This paper presents a machine learning (ML) framework for detecting network attacks. The proposed ML-based network anomaly detection (NAD) comprises data-preprocessing, ML modeling, and post-processing. Random forest and XGBoost algorithms are the base models of the proposed ensemble learning procedure. Trained and tested by ZYELL – NCTU Network Anomaly Detection Challenge datasets, the experimental results show the proposed scheme can detect 4 types of network anomalies: Smurf-DOS attack, Ip sweep probing, Nmap probing, and port sweep probing. The tested scoring is 0.52, which is ranked as the third place of the challenge.

## 1. INTRODUCTION

Network attack has become a serious threat since billions of computers are connected worldwide with the Internet. Consequently, network anomaly detection (NAD) is an important technology to improve computer network security. Signature-based or rule-based NAD has been used for a long time to detect malicious attacks where the packet contents or the network flows having recognizable features. The 2003 Slammer Worm [1] is an example. Since rule-based approaches can hardly be applied to detect unseen events. Machine learning (ML) methods such as XGBoost [2], SVM [3], and Random Forest [4] have been proposed in NAD recently [5] [6]. ML-based NAD remains a big challenge due to the traffic data's two properties: large scale and class imbalance.

This paper investigates the feasibility and effectiveness of several ML-based NAD algorithms that treat the NAD as a multi-class classification problem. Based on random forest [7] and XGBoost [8] algorithms, we proposed a NAD framework to detect four malicious attacks, which are Smurf-DOS, IP sweep probing, Nmap probing, and port sweep probing. Tested by the data in ZYELL – NCTU Network Anomaly Detection Challenge (One of the Grand challenges in IEEE ICASSP 2021), the proposed approach achieves a 0.52 score. The rest of the paper is organized as follows. The fundamentals of NAD and the machine learning algorithms are briefly reviewed in Section 2. Section 3 presents our proposed NAD framework. Experimental results are given in Section 4. Section 5 concludes this work.

## 2. BACKGROUND

### 2.1. Network Anomaly Classification

Two main network attacks are concerned in this paper: Smurf-DOS and Probing. And there are three types of probing attacks mentioned: Ip sweep, Nmap, Port sweep.

This paper focuses on two main network attack categories, smurf-based distributed denial of service attack (Smurf-DOS) [9] and probing. Smurf-DOS is an amplification attack that uses unprotected intermediate networks to amplify the attack traffic. ICMP [10] packets are used and broadcasted with the spoofed source address, resulting in numerous echo responses and traffic sent to the victim spoofed source.

Probing can be abused to discover the vulnerabilities of the members in a target network. IP sweep probe, also known as ICMP sweep probe, sends ICMP echo requests to every IP address in a network. The attacker determines the IP address from the replying packets and the target-specific IP address to perform an attack. Nmap is short for Network Mapper that scans the IP address, open ports, and accessible services. [11] Hence it can also be used to probe vulnerabilities in the network. Port sweep is an attack by which the attacker tries to scan multiple hosts in a network.

### 2.2. Random Forest

Random forest is a supervised machine learning algorithm. The algorithm is an ensemble of multiple decision-tree classifiers to obtain a more accurate and stable prediction. To have a model with high accuracy, it is required that each classifier being independent and having a fair accuracy. Bootstrap Aggregating [12], also called Bagging, and Boosting [13], can be utilized when the dataset size is limited. By nature, a random forest classifier rarely overfits the model if there're enough trees in the forest.

### 2.3. XGBoost

XGBoost stands for "Extreme Gradient Boosting", where the term "Gradient Boosting" [14]. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. XGBoost uses decision tree ensembles of a set of classification and regression trees (CART) [15],

$$\hat{y}_i = \sum_{k=1}^{k} f_k(x_i), f_k \in F$$

K is the number of trees, f is the function in the functional space F, and F is the set of all possible CARTs. The objective function to be optimized is given by

$$\text{obj} = \sum_{i=1}^{n} l(y_i, \ \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

The algorithm utilized an additive training procedure by adding new functions into the model recursively. Fig. 1 illustrates the greedy algorithm [8].

```
Algorithm 1: Exact Greedy Algorithm for Split Finding
Input: $I$, instance set of current node
Input: $d$, feature dimension
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i$, $H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ to $m$ do
    $G_L \leftarrow 0$, $H_L \leftarrow 0$
    for $j$ in sorted($I$, by $\mathbf{x}_{jk}$) do
        $G_L \leftarrow G_L + g_j$, $H_L \leftarrow H_L + h_j$
        $G_R \leftarrow G - G_L$, $H_R \leftarrow H - H_L$
        $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
    end
end
Output: Split with max score
```

**Figure 1. Pseudocode of split finding by using exact greedy algorithm [8]**

## 3.THE PROPOSED NAD FRAMEWORK

### 3.1. Data Pipeline

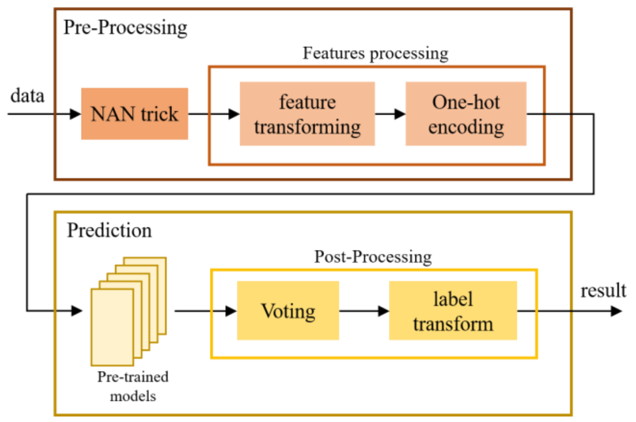


**Figure 2. The framework for prediction structure**

Figure 2 shows the data pipeline of the proposed NAD framework. Pre-processing comprises data cleaning and feature processing. In data cleaning, the missing or NAN values are replaced by zero. Feature processing converts the categorical attributes such as "time", "src", "dst" into an integer. One-hot encoding is applied to the feature "app". The prediction model contains pre-trained models and post-processing (optional). The pre-trained models are XGBoost and Random decision forest classifiers. There can be single or multiple models. Post-processing combines the multiple models by voting. For example, if they're a series of numbers such as 0, 0, 0, 1, 2, 3, 4 from seven ensemble models, class-0 will be the final result after voting. LabelEncoder [16] is applied to convert prediction result into label format.

### 3.2. Training Procedure

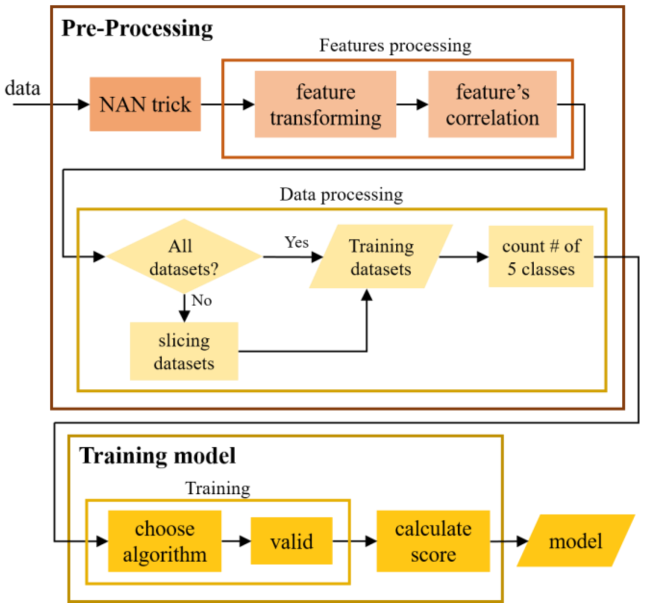


**Figure 3. The framework for training structure**

Figure 3 shows the model training procedure. It consists of two parts, pre-processing and training model. Same data cleaning and pre-processing are applied. The feature correlation, shown in Figure 4, explains the relationship between multiple variables and dataset attributes. The datasets are imbalanced. Most of the data are of normal samples, and the anomaly cases are rare events. Therefore, we also compute the feature correlation with the sub-sample part. According to Figure 4, most features have low correlation, so we choose features randomly or keep all the features to train our model. Two approaches can be used to solve the data imbalance problem, cross-validation, and sub-sampling. The second one is to divide the dataset fairly and reconstruct it into multiple csv files. Put them to train the model with different parameters; then, we will get different ensemble models. For example, we divide five classes of labels into five parts fairly and reconstruct them into five csv files. Training them partly, then we will get five ensemble models. It is more time-efficient, and we can increase the diversity of models.

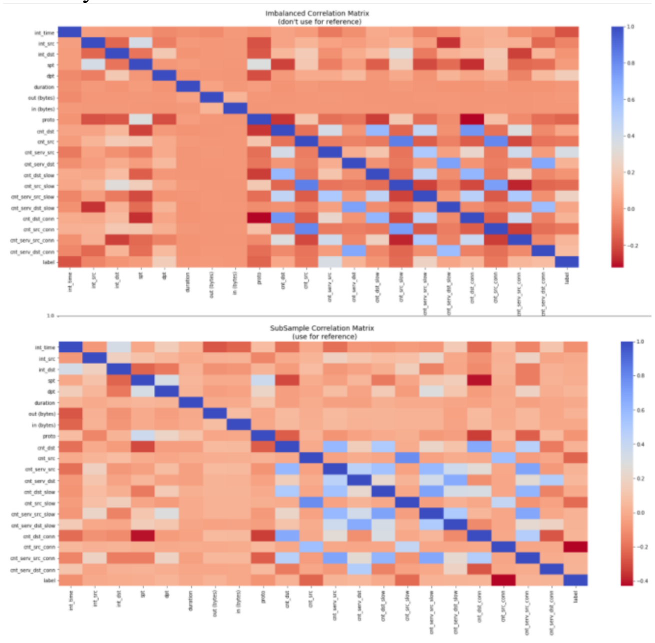


**Figure 4. The upper figure shows the correlation of imbalanced datasets. The other figure shows the correlation of subsamples.**

$$\text{Evaluation criteria} = \alpha \times \left(1 - \frac{\log(total\ cost)}{\log(\max\ cost)}\right) + (1-\alpha) \times (macro\ F_{\beta} score)$$

where α = 0.3, β = 2.

Total cost = cost value calculated by the provided cost matrix

Max cost = max cost value × number of total entities

macro = macro averaging

$$\text{F\_β score} = (1+\beta^2) \times \frac{Precision \times Recall}{\beta^2 \times Precision + Recall}$$

| | Normal | DOS-smurf | Probing-IP sweep | Probing-Nmap | Probing-Port sweep |
|---|---|---|---|---|---|
| Normal | 0 | 2 | 1 | 1 | 1 |
| DOS-smurf | 2 | 0 | 1 | 1 | 1 |
| Probing-IP sweep | 1 | 2 | 0 | 1 | 1 |
| Probing-Nmap | 1 | 2 | 1 | 0 | 1 |
| Probing-Port sweep | 1 | 2 | 1 | 1 | 0 |

**Table 1. cost matrix for example provided by official**

Training and score calculating are carried out in the training model part. XGBoost and Random decision forest are the base models. In this paper, the accuracy of the pre-trained model's validation data is referred to as score_1, and our score calculated by the official website, shown in Table 1, is called score_2 in this paper.

## 4. EXPERIMENTAL RESULTS

In this section, we will use "1203_firewall.csv", "1210_firewall" and "1216_firewall.csv" provided by the official challenge websites training datasets to test our framework. We will also merge those three files into one csv file as validation datasets to calculate a score. We use the officially released "0123_firewall.csv", "0124_firewall.csv", "0125_firewall.csv", and "0126_firewall.csv" for evaluating criteria after labeling.

### 4.1. Experimental results on testing datasets

| Submit version | Implementation | Score on validation datasets | Score by Official |
|---|---|---|---|
| Version 1 | • Model Algorithm：Random Forest *(whole datasets training)*<br>• Numbers of ensemble models：one<br>• Voting based：none use | 0.90663665 | 0.4976868602 |
| Version 2 | • Model Algorithm ：XGBoost *(whole datasets training)*<br>• Numbers of ensemble models：one<br>• Voting based：none use | 0.950664663 | 0.5219859738 |
| Version 3 | • Model Algorithm ：Random Forest and XGBoost *(training by sub-sample)*<br>• Numbers of ensemble models：seven *(six XGBoost, one Random Forest)*<br>• Voting based：mode | 0.008117 | 0.02621276 |
| Version 4 | • Model Algorithm ：Random Forest and XGBoost *(whole datasets training)*<br>• Numbers of ensemble models：seven *(four XGBoost, three Random Forest)*<br>• Voting based：mode | 0.671112 | 0.4658674915 |
| Version 5 | • Model Algorithm ：XGBoost *(whole datasets training)*<br>• Numbers of ensemble models：five *(five XGBoost)*<br>• Voting based ：mode | 0.67112053 | 0.4676763738 |

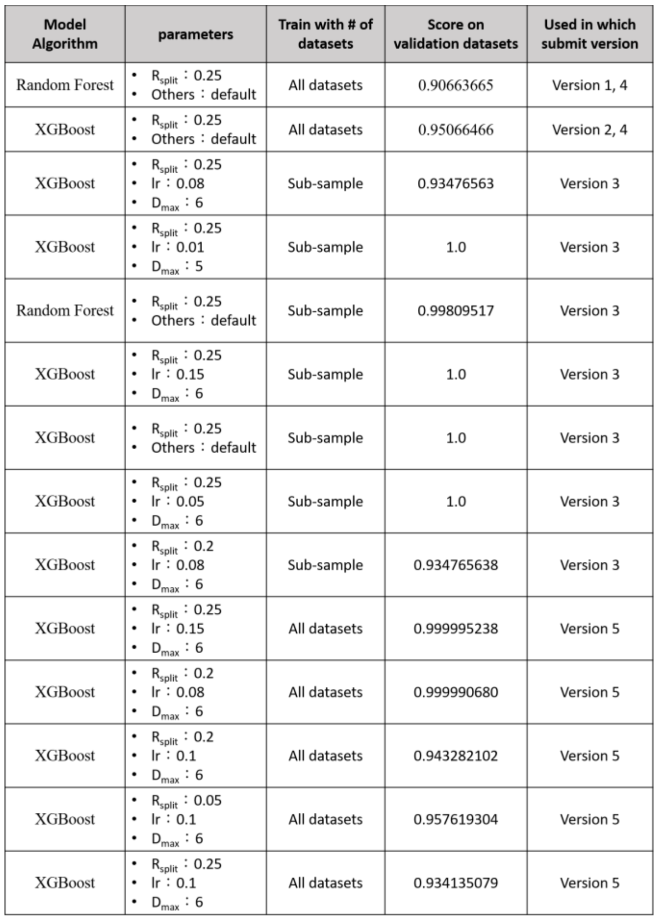
**Table 2. Results of five submit versions**

Table 2 shows the results of five submit versions. The implementation addresses these submissions' differences, including the number of ensembled models, the base model types, data sub-sample strategy, post-processing approach (voting), etc. The score in the "score on validation datasets" column is computed by our scoring function. The "Score by Official" column shows the official evaluation results. We find that there's a gap between these two columns. It might be due to the differences in the evaluation and the cost matrix.

Moreover, the results shown in table 2 indicate that version 3 has the worse grade. One of the reasons is that we only put part of all datasets to train the model. Since the datasets are quite imbalanced, the models could be overfitted if not sufficient samples of rare cases are used in training. In other words, the models trained with the whole datasets perform better than version 3. We can also find that the performances of version 4 and version 5 are not as good as version 1 and version 2. The results imply that the voting strategy can be further improved. For example, we could replace the voting with a weighted sum of the prediction probabilities.

## 4.2. Experimental results of training models

Table 3 shows the configurations of the pre-trained models that are used in prediction. The base models are "Random Forest" and "XGBoost". The parameter column shows the settings, such as the train-test split ratio ($R_{split}$) and the classifiers' hyperparameters. The learning rate ($lr$) is carefully chosen such that the step size decreases with training iterations to prevents overfitting. The default value is 0.3. The max depth ($D_{max}$) parameter means the maximum depth of the tree. Increasing this value will make the model more complex and more likely to overfit; its default is 6. We try two different datasets as training datasets; one is whole datasets, the other one is multiple sliced data. It should be noted that the validation datasets in the "Score in validation datasets" column is part of training data with multiply by split ratio, not the same mentioned in 4.1. More folds of cross-validation in different datasets can help to prevent overfitting, whether trained with the complete datasets or sub-samples.



| Model Algorithm | parameters | Train with # of datasets | Score on validation datasets | Used in which submit version |
|---|---|---|---|---|
| Random Forest | • $R_{split}$：0.25 • Others：default | All datasets | 0.90663665 | Version 1, 4 |
| XGBoost | • $R_{split}$：0.25 • Others：default | All datasets | 0.95066466 | Version 2, 4 |
| XGBoost | • $R_{split}$：0.25 • lr：0.08 • $D_{max}$：6 | Sub-sample | 0.93476563 | Version 3 |
| XGBoost | • $R_{split}$：0.25 • lr：0.01 • $D_{max}$：5 | Sub-sample | 1.0 | Version 3 |
| Random Forest | • $R_{split}$：0.25 • Others：default | Sub-sample | 0.99809517 | Version 3 |
| XGBoost | • $R_{split}$：0.25 • lr：0.15 • $D_{max}$：6 | Sub-sample | 1.0 | Version 3 |
| XGBoost | • $R_{split}$：0.25 • Others：default | Sub-sample | 1.0 | Version 3 |
| XGBoost | • $R_{split}$：0.25 • lr：0.05 • $D_{max}$：6 | Sub-sample | 1.0 | Version 3 |
| XGBoost | • $R_{split}$：0.2 • lr：0.08 • $D_{max}$：6 | Sub-sample | 0.934765638 | Version 3 |
| XGBoost | • $R_{split}$：0.25 • lr：0.15 • $D_{max}$：6 | All datasets | 0.999995238 | Version 5 |
| XGBoost | • $R_{split}$：0.2 • lr：0.08 • $D_{max}$：6 | All datasets | 0.999990680 | Version 5 |
| XGBoost | • $R_{split}$：0.2 • lr：0.1 • $D_{max}$：6 | All datasets | 0.943282102 | Version 5 |
| XGBoost | • $R_{split}$：0.05 • lr：0.1 • $D_{max}$：6 | All datasets | 0.957619304 | Version 5 |
| XGBoost | • $R_{split}$：0.25 • lr：0.1 • $D_{max}$：6 | All datasets | 0.934135079 | Version 5 |

$R_{split}$：Split Ratio, lr：Learning Rate, $D_{max}$：max depth

**Table 3. Information of the pre-trained models that used in prediction.**

## 5. CONCLUSIONS

In this paper, we present an ML-based network anomaly detection (NAD). The results prove the feasibility of detecting 4 types of malicious attacks. Applying some pre-processing schemes such as data cleaning and "feature transforming" can improve the detection rate and accuracy. At last, our method is tested by the Grand challenge, and the final grade is about 0.5219859738. During post-processing in prediction part, we believe the performance can be furthered improve by replacing the voting strategy with cross entropy.

## 6. ACKNOWLEDGMENT

The authors would like to thank Mr. Hung-Kun Chang for his fruitful discussion.

## 7. REFERENCES

[1] D. e. a. Moore, "Inside the slammer worm," *IEEE Security & Privacy 1.4 ,* pp. 33-39., 2003.

[2] H. e. a. Jiang, "Network intrusion detection based on PSO-XGBoost model," *IEEE Access 8,* pp. 58392-58401., 2020.

[3] T. a. J. M. Shon, "A hybrid machine learning approach to network anomaly detection," *Information Sciences 177.18,* pp. 3799-3821, 2007.

[4] R. a. B. A. T. Primartha, "Anomaly detection using random forest: A performance revisited," *2017 International conference on data and software engineering (ICoDSE). IEEE*, 2017.

[5] T. B. O. a. M. C. Ahmed, "Machine learning approaches to network anomaly detection," *Proc. SysML,* 2007.

[6] N. e. a. Duffield, "Rule-based anomaly detection on IP flows," *IEEE INFOCOM 2009. IEEE,* 2009.

[7] Pal, Mahesh. "Random forest classifier for remote sensing classification." *International journal of remote sensing* 26.1 (2005): 217-222.

[8] Chen, Tianqi, and Carlos Guestrin, "XGBoost：A Scalable Tree Boosting System," *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining,* 2016.

[9] Kumar, Sanjeev. "Smurf-based distributed denial of service (ddos) attack amplification in internet." *Second International Conference on Internet Monitoring and Protection (ICIMP 2007)*. IEEE, 2007.

[10] Kaushik, Atul Kant, and R. C. Joshi. "Network forensic system for ICMP attacks." *International Journal of Computer Applications* 2.3 (2010): 14-21.

[11] SANS Institute Reading Room. ICMP attack illustrated. http://www.sans.org/reading_room/whitepapers/threats/ic mp_attacks_illustrated_477?show=477.php&cat=threats

[12] Bootstrap Aggregation https://corporatefinanceinstitute.com/resources/knowledge/other/bagging-bootstrap-aggregation/.

[13] Schapire, Robert E., "A brief introduction to boosting," *Ijcai. Vol. 99,* 1999 2010.

[14] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.," *Ann. Statist 29,* pp. 1189 - 1232, October 2001.

[15] Loh, Wei-Yin, "Classification and regression trees," *Wiley interdisciplinary reviews: data mining and knowledge discovery 1.1,* pp. 14-23., 2011.

[16] "scilit-learn," https://scikit-learn.org/stable/.