Chinese Journal of Electronics
vol. 33, no. 3, pp. 645–656, May 2024
https://doi.org/10.23919/cje.2022.00.360

RESEARCH ARTICLE

QARF: A Novel Malicious Traffic Detection Approach via Online Active Learning for Evolving Traffic Streams

Zequn NIU[1], Jingfeng XUE[1], Yong WANG[1], Tianwei LEI[1], Weijie HAN[2], and Xianwei GAO[1]

1. *School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China*
2. *School of Space Information, Space Engineering University, Beijing 101416, China*

Corresponding author: Yong WANG, Email: wangyong@bit.edu.cn
Manuscript Received October 24, 2022; Accepted February 14, 2023

Abstract — In practical abnormal traffic detection scenarios, traffic often appears as drift, imbalanced and rare labeled streams, and how to effectively identify malicious traffic in such complex situations has become a challenge for malicious traffic detection. Researchers have extensive studies on malicious traffic detection with single challenge, but the detection of complex traffic has not been widely noticed. Queried adaptive random forests (QARF) is proposed to detect traffic streams with concept drift, imbalance and lack of labeled instances. QARF is an online active learning based approach which combines adaptive random forests method and adaptive margin sampling strategy. QARF achieves querying a small number of instances from unlabeled traffic streams to obtain effective training. We conduct experiments using the NSL-KDD dataset to evaluate the performance of QARF. QARF is compared with other state-of-the-art methods. The experimental results show that QARF obtains 98.20% accuracy on the NSL-KDD dataset. QARF performs better than other state-of-the-art methods in comparisons.
Keywords — Active learning, Online learning, Malicious traffic, Concept drift, Data imbalance.
Citation — Zequn NIU, Jingfeng XUE, Yong WANG, *et al.*, "QARF: A Novel Malicious Traffic Detection Approach via Online Active Learning for Evolving Traffic Streams," *Chinese Journal of Electronics*, vol. 33, no. 3, pp. 645–656, 2024. doi: 10.23919/cje.2022.00.360.

I. Introduction

Network-based intrusion detection system (NIDS) analyzes traffic data in network to identify malicious behavior, which is widely used in organizations and companies to detect network attack. Traditional NIDS is mostly based on matching the fingerprint of attack or strict identification of normal behavior. However, in practical applications, fingerprint matching brings high overhead on the maintenance of fingerprint database and failure to identify unknown attacks, while strict normal behavior identification suffers from a high false alarm rate [1]. Compared with traditional intrusion detection techniques, machine learning based detection techniques perform well on detecting malicious traffic and have no need to maintain a large database, which makes machine learning techniques widely discussed and used in intrusion detection.

However, the performance of machine learning-based algorithms is significantly affected by the quality of the training data. An ideal training dataset contains a large number of balanced and labeled samples, but in practical applications, the traffic data is usually evolutionary streaming, imbalanced, and rare labeled. Specifically, machine learning-based network intrusion detection techniques are facing the following three challenges:

1) Evolving traffic streams are unable to be processed by offline learning based techniques. In network, traffic data continues to generate and change, e.g., feature distribution of traffic data is changed by unknown malware, which requires frequent updates of machine learning-based detection models to capture the changes. However, frequent retraining and redeployment will bring unacceptable cost and time overhead [2], [3].

2) Labeled traffic samples are rare while the acquisition of the ground truth of unlabeled samples is expensive, which leads to inadequate training of supervised learning based model. Continuous generation of network

Associate Editor: Prof. Jiwei HUANG, China University of Petroleum.

traffic makes raw traffic easy to acquire, but these traffic are unlabeled that cannot be used to train supervised learning based models, while the ground truth of unlabeled data is very expensive to obtain. One possible solution to this challenge is to employ semi-supervised learning technology, which expands labeled dataset size. However, in semi-supervised learning techniques, the scope of pseudo-labels is limited to existing labeled instances, causing unknown malicious traffic mislabeled [4].

3) The imbalance of traffic data leads to the degradation of machine learning based detection models. In traffic streams, benign flow is much more than malicious ones, which causes models bias towards the benign category, resulting in ignorance of uncommon attacks [5].

To complicate matters further, the above challenges are not individual to each other in malicious traffic detection task. In practical applications, the traffic streams may be drift, rare labeled and imbalanced, making solutions that aim to solve individual challenges no longer effective [6].

The specific objective of this study is to explore how to effectively detect malicious streaming traffic which is drift, rare labeled and imbalanced. In this study, a holistic approach, queried advanced random forests (QARF), is utilized, integrating adaptive random forests (ARF) and adaptive margin sampling strategy to establish an online active learning-based malicious traffic detection approach. ARF is an online supervised algorithm that works with evolving data streams, and the adaptive margin sampling strategy selects representative instances from a large amount of unlabeled traffic streams and queries the ground truth of the selected instances for ARF training. QARF achieves adaptive adjustment following the change of traffic streams, which maintains a good balance between the cost of querying and the performance of the detection classifier.

The contribution of this paper is as follows:

1) QARF, an online active learning based malicious traffic detection approach, is proposed, which can effectively identify malicious traffic data in streaming traffic that drift, rare labeled and imbalanced.

2) An adaptive sampling strategy is proposed, which can adaptively adjust the threshold of margin sampling strategy with following the change of traffic flow. Through the strategy, QARF achieves saving the cost of querying in representative instances selection.

3) We conducted experiments on the NSL-KDD dataset and verified the superiority of our approach.

The rest of this paper is organized as follows: Section II discusses the related work, Section III presents preparatory knowledge, Section IV details the design methodology, Section V evaluates the performance of the proposed approach, and Section VI concludes the paper.

II. Related Work

Malicious traffic is any suspicious content or connection created or received over the network which has the threat of causing a security incident [7]. In the field of cyberspace security, the detection of malicious traffic is usually implemented by intrusion detection systems [8]. Intrusion detection techniques were proposed by Denning *et al.* [9] and have been widely studied. Traditional intrusion detection techniques are classified into anomaly detection and misuse detection [10], [11]. Misuse-based detection determines the attack behavior by matching attack signatures, and anomaly-based detection records the normal behavior pattern of the system and regards behavior as an intrusion when the behavior is found to be out of the normal behavior pattern characteristics. The two traditional intrusion detection techniques have different defects. Misuse-based detection can only detect the attacks that can be matched by the signatures in the attack signature database, which makes high cost on maintaining fingerprint database and powerless in the face of new attacks, while anomaly-based detection has the problem of a high false alarm rate.

With the widespread application of machine learning technology, malicious traffic detection methods based on machine learning techniques have become the focus of related research and been widely used [12]. Machine learning-based malicious traffic detection models classify network behavior to detect attack. Compared with the traditional intrusion detection techniques, the machine learning-based detection techniques are free from the reliance on fingerprints, which greatly improves the efficiency. Meanwhile, the machine learning based detection techniques are better at detecting unknown attacks.

However, the complex state of traffic is a challenge to machine learning based detection methods. The ideal training data for machine learning based models is offline, balanced and labeled, but the traffic data is usually streaming, drift, imbalanced and rare labeled in practical applications. Researchers have focused on how to detect suboptimal traffic effectively. To detect evolving traffic streams, researchers employ online learning techniques to achieve adaptive adjustment. Murugaraj *et al.* [13] proposed a hybrid online-offline system, in which online model keeps learning from traffic streams and offline model selects samples for learning. Mahmodi *et al.* [14] proposed a new drift aware adaptive method for detecting attack in streams by employing linear-order algorithms and Gaussian-order algorithms with a slide window to capture the drift in streams. Bhatia *et al.* [15] proposed MemStream which uses auto-encoder techniques to extract features and memory module to save the trend of streams to adapt to the drift of the streaming traffic. Jain *et al.* [16] combined MapReduce technology and K-means sliding window clustering technique to handle concept drift in massive amounts of traffic. To detect traffic with rare labeled samples, researchers employ active learning technology to query the ground truth of unlabeled samples for retraining model. Considering that active learning techniques are effective techniques to avoid keeping redundant training instances, Deka *et al.* [17] proposed an abnormal detection method based on parallel active learning to detect DDoS (distributed denial of service) in traffic. Zhang *et al.* [4] combined semi-supervised learning and active learning techniques to extract unlabeled samples with rich information to query the ground truth, and then uses the queried samples to retrain the model. To detect imbalanced traffic, Al-Yaseen *et al.* [18] proposed a hybrid IDS combining support vector machine, extreme learning machine and K-means clustering algorithm to improve the detection rate of uncommon attacks. Ahmim *et al.* [19] proposed HCPTC-IDS, an IDS system based on predict probabilities of decision trees. The HCPTC-IDS system consists of two layers, the first layer is a decision tree, and the second layer is an end classifier that combines the different probabilities of the first layer to make predictions. Wang *et al.* [20] proposed DA-Transfer based on deep transfer learning, to improve the performance of small-sample classification models in anomaly detection. Lin *et al.* [21] proposed a multi-level feature fusion model (MFFusion) with employing deep learning techniques and adaptive balanced training method to achieve automatic feature extraction and effective training when the training samples are imbalanced. Chapaneri *et al.* [22] proposed Wasserstein GAN improved deep regret (WGAN-IDR) to generate augmented samples to extend the size of labeled dataset, aiming to train the detection model with a balanced training set.

The above approaches are proposed to response to challenges in traffic streams, but each approach can only response one single challenge. However, in practical applications, traffic streams do not come with only one challenge, but with multiple challenges at the same time. Therefore, the purpose of this research is to investigate how to detect malicious traffic effectively when the traffic streams are drift, rare labeled and imbalanced.

III. Adaptive Random Forests

Adaptive random forests [23] (ARF) is a stream-adapted version of the random forests algorithm. Original random forests algorithm is an offline learning algorithm and can only be trained on offline dataset, which conflicts with data streams. For stream learning, ARF uses VFDT [24] (very fast decision tree) tree as the base classifier to build the forest model, but differs from VFDT in feature selection, sampling method, and pruning strategy of trees to avoid overfitting. In the feature selection, basic classifier in ARF uses a subset rather than full set of features that used in VFDT. On the sampling strategy, rather than using full dataset in VFDT, ARF uses online Bagging approach [25] with Poisson ($\lambda=6$) to assign a unique training subset to each basic classifier. On the pruning strategy, the trees of ARF will not be pruned, but grown completely. Unique subsets, different feature subsets and full growth of the trees widen the difference between basic classifiers in ARF and thus avoid overfitting during ARF training.

Concept drift in streaming data is a challenge that online learning-based models need to face. To combat concept drift, ARF employs ADWIN [26] (ADaptive WINdowing) and makes some improvements on ADWIN to work as the concept drift detector to monitor data streams. Different from ADWIN, ARF does not replace the working base classifier with a new initial based classifier once the concept drift is detected, instead, a two-stage setup of drift warning and drift detection is performed. If a drift warning occurs, ARF creates a new tree in the background as a background tree. The new background tree is trained together with the working base tree. When a drift detection threshold is triggered, the working base tree which triggers the drift detection threshold will be replaced with the background tree. In this way, ARF ensures that the base classifiers are adaptively adjusted when detecting evolving data streams.

IV. The Proposed Method

1. Overview

QARF is proposed to detect streaming traffic data, therefore the system of QARF processes newly-arrived instances one by one. QARF consists of ARF and a query and filtering component (QFC) which includes the adaptive sampling strategy, and the overall architecture of QARF is shown in Figure 1. In QARF, ARF works as a basic classifier to identify newly-arrived instance, and QFC determines whether the newly-arrived instance need to be queried for the ground truth. If an instance is queried and labeled by experts, both the ARF and the QFC will be updated through learning the queried instance.

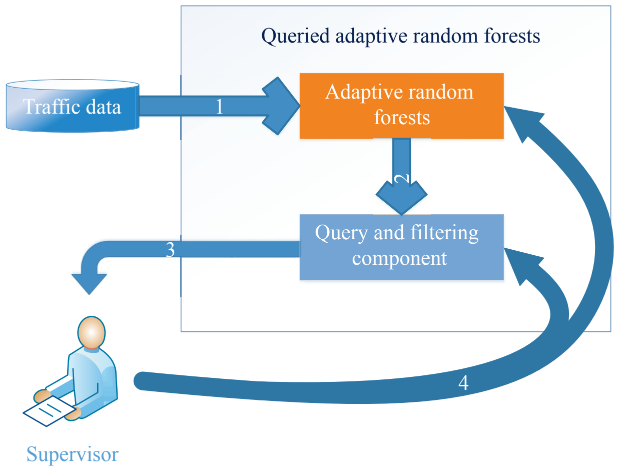

Queried adaptive random forests
Traffic data
1
Adaptive random forests
2
Query and filtering component
3
4
Supervisor

Figure 1 Overview of queried adaptive random forests to malicious traffic detection in a traffic stream.

Combined with the overall architecture of QARF in Figure 1, the main process of QARF is as follows:

1) A newly-arrived instance is inputted into ARF.

2) ARF outputs the prediction to QFC.

3) QFC determines whether the instance is valuable to query. If valuable, QFC sends the instance to supervisor to query for the ground truth.

4) The labeled instance is sent from the supervisor

to ARF and QFC for updating. Specifically, ARF learns the labeled instance and updates the parameters in the model. Meanwhile, QFC compares the ground truth of the instance with the predicted label made by ARF, and then updates the threshold value in QFC which is used to determine whether instances are valuable.

2. Query and filtering component

Query and filtering component (QFC) is proposed to select representative instances from traffic streams for querying when labeling budget is limited. Data in traffic streams is large but unlabeled, while labeling instances by experts is accurate but costly, which means labeling all raw instances is not feasible. Selecting representative instances from traffic streams to label and train the detection model rather than labeling the whole traffic stream is acceptable on both the requirements for model training and the spending of label querying. However, ARF is an algorithm for classification and does not have the ability to identify whether instances are representative or not, thus QFC is proposed in this paper by employing margin sampling method to select representative instances out from traffic streams for querying.

Margin sampling is one of query strategies in active learning technology [27]. Margin sampling selects instances that can be easily determined to be in two categories, i.e., instances with small differences between the maximum posteriori probability and the second most posteriori probability. The calculation of margin(x) is shown in (1), where $\hat{y}_1$ and $\hat{y}_2$ represent the category with the maximum posteriori probability and the category with the second maximum posteriori probability in the classification, respectively. The instances selected by margin sampling are provided to experts for labeling and then used to train ARF to improve the performance of the basic classifier.

$$\text{margin}(x) = \arg\min_{x}(P(\hat{y}_1|x) - P(\hat{y}_2|x)) \tag{1}$$

In QFC, representative instances are filtered by comparing their margin values with the threshold of QFC. Margin sampling provides the formula for calculating the margin value of instances, but does not provide how to determine the appropriate threshold to select uncertain instances, which is the focus of QFC. Because the evolving traffic flow is always in dynamic change, fixed threshold is not appropriate in QFC. The threshold should change adaptively to minimize the expert query cost in condition of effective training of ARF, e.g., if concept drift occurs in traffic streams, the threshold should be increased to let the classifier learn more queried instances, otherwise the threshold should be decreased to let only the most uncertain instances labeled which aims to reduce the expert query cost. To achieve the advanced adjustment of the threshold, sliding window structure is adopted in QFC to cache information about the latest instances and update the threshold value.

Figure 2 describes the procession in QFC and the pseudo-code of QARF is shown in Algorithm 1. First, supposing that data A_n is the data block captured in traffic stream by QARF at moment n (step 1, Figure 2). ARF identifies A_n and outputs prediction including the predict label and the predicted probability for each label (step 2, Figure 2), and then margin value of A_n is calculated in formula (1) (step 3, Figure 2). After calculation, margin of A_n is used to compared with threshold in QFC (step 4, Figure 2), and if margin value of A_n is lower than the threshold in QFC, A_n will be regarded as representative instances and sent to the expert to query for the ground truth (steps 5 and 6, Figure 2). After labeled by experts, A_n is used to update ARF and QFC (steps 7 and 8, Figure 2).

Algorithm 1 Queried adaptive random forests algorithm

Input: A: traffic data stream; n: number of instances; T: margin sampling threshold in QFC; win_f: window to cache margin of incorrect predictions; win_t: window to cache margin of correct predictions; win_{rec}: window to cache the performance of basic classifier in the classification; grad: gradient of the window weight; w: weight of sliding windows.

Output: y_n: the predict label of A_n; y_p: the posteriori probability of A_n.

```
While HasNext($A$) do:
  get the next instance $A_n$ = next($A$);
  get the predict label $y_n$ = ARF.predict($A_n$);
  get posteriori probability $y_p$ =ARF.predict_proba($A_n$);
  calculate margin $k$ = Margin($y_p$);
  select representative instances If ($k < T$) then:
    query for the ground truth $y$ = Label_query($A_n$);
    the predict label of $A_n$ is compared with the queried
    ground truth If NotSame($y$, $y_p$) then:
    $win_f$.First_In_First_Out($k$);
    $win_{rec}$.First_In_First_Out(1);
    Else:
    $win_t$.First_In_First_Out($k$);
    $win_{rec}$.First_In_First_Out(0);
    evaluate the performance of ARF
    $cr = \log(win_{rec}.\text{count}(0)/win_{rec}.\text{count}(1))+1$;
    update the weight of sliding windows
    $w = w + \text{grad} \times cr$;
    update the margin sampling threshold
    $T = w \times (win_t.\text{mean}() - win_t.\text{std}())$
        $+ (1 - w) \times (win_f.\text{mean}() + win_f.\text{std}())$;
End function
```

In QFC, every time an instance queried, threshold will be updated according to the performance of ARF in the classification. First, the predict label which is given by ARF is compared with the ground truth given by expert to determine whether ARF makes a correct prediction for the instance (step 8, Figure 2). Three sliding windows, win_t, win_f, and win_{rec}, are used to store the information of the comparison. Among the sliding windows, win_t is used to cache margin of correct predictions and win_f is for incorrect predictions, meaning if the instance is correctly classified by ARF, the margin of

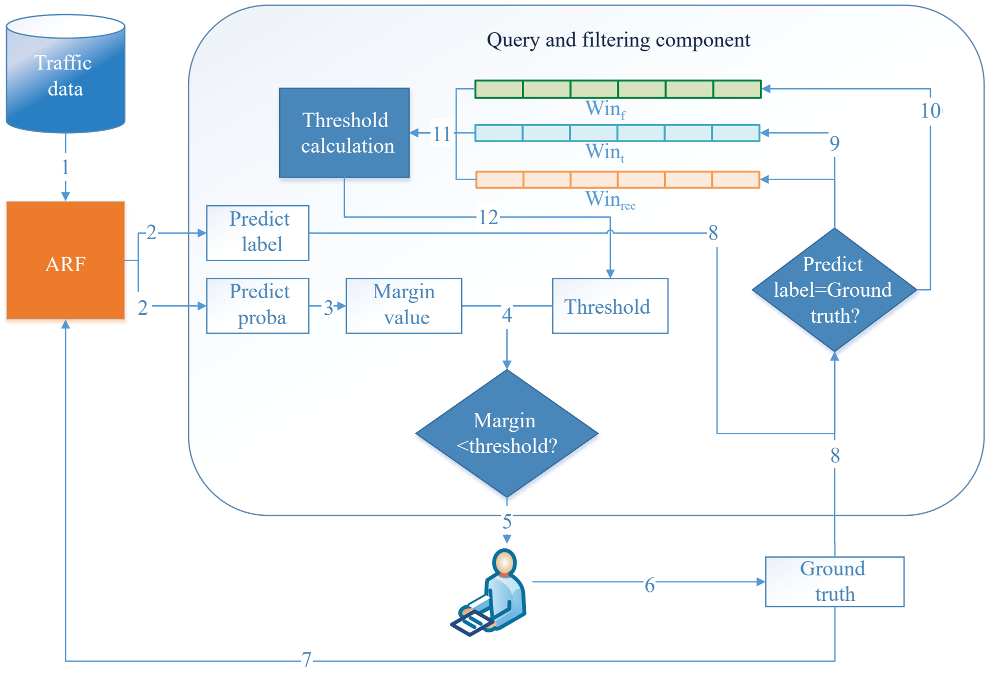


Figure 2 Schematic representation of complete query and filtering component procedure.

the instance is cached in win_t, otherwise the margin is cached in win_f. Another sliding window win_{rec} is used to cache the performance of ARF in the classification, meaning if the instance is correct predicted by ARF, the value "1" is cached into win_{rec}, otherwise "0" is input into win_{rec} (step 9 and 10, Figure 2). To capture the recent changes of traffic streams, the three sliding windows work as queues with FIFO (first-in-first-out) strategy to store the information of latest instances, meaning when the windows are filled up, the information of the earliest instance will be deleted and the new one is cached.

The average and standard deviation of sliding windows reflect the distribution of current traffic streams, thus the average and standard deviation of win_t and win_f are used to calculate threshold (step 11 and 12, Figure 2). The calculation of threshold is shown in (2), where $\overline{x}$ represents the average of elements in sliding windows, and s represents the standard deviation of sliding windows. In (2), w represents the weight of sliding windows, and works to adjust the weight between win_t and win_f according to the change of traffic streams.

$$\begin{aligned}\text{threshold} =& w \times (\overline{x}_{win_t} - s_{win_t}) \\ &+ (1-w) \times (\overline{x}_{win_f} + s_{win_f})) \end{aligned} \tag{2}$$

The weight w updates once an instance is inputted into QFC, and the update of w begins at calculating the change rate of threshold. The change rate of threshold is different at different moment, for example, when ARF has a high error rate in the recent time, it means the distribution of traffic streams has changed, and the threshold needs to be increased rapidly to let more uncertain instances queried and used to update ARF so that ARF can adapt to the traffic changes rapidly. On the contrary, when the error rate of ARF is low, it means the streaming traffic is smooth, and the threshold needs to be decreased gradually to reduce the number of labeled instances so that the cost of expert querying is reduced. win_{rec} is used to calculated the change rate and the calculation is shown in (3), where m represents the length of win_{rec}, win_{rec_m} represents Win_{rec} at the m element, and cr represents the change rate.

$$cr = \ln \frac{\sum_{i=0}^{n} win_{rec_m}}{m - \sum_{i=0}^{n} win_{rec_m}} + 1 \tag{3}$$

After the calculation of cr, w is calculated as shown in (4).

$$w = \begin{cases} w + \text{grad} \times cr, & t < 0 \\ w - \text{grad} \times cr, & t \geq 0 \end{cases} \tag{4}$$

In (4), grad represents the gradient of w, and t represents the flag whether ARF predicts the queried instance correctly. If ARF makes correct prediction, $t \geq 0$, otherwise $t < 0$.

V. Experiments and Analysis

1. Data sets

NSL-KDD [28] is used to evaluate the performance of QARF in this paper, because NSL-KDD is a benchmark dataset of intrusion detection research and also widely used in online learning research. NSL-KDD is an improved version of the well-known network intrusion traffic dataset KDD'99 [29]. In the training set of KDD'99, many records are redundant, which makes the classifier often biased towards more frequent records. At the same time, the duplicate records in the test set make the performance of machine learning-based model suffered, and the overall detection rate is relatively high due to the high detection rate of frequent records. NSL-KDD has a more reasonable data distribution compared with KDD'99, as NSL-KDD removes the redundant data of KDD'99. NSL-KDD can be used as a benchmark dataset to evaluate the performance of intrusion detection methods.

In NSL-KDD, there are 22 attacks in the training set, which can be classified into four attack categories: DoS, Probe, R2L and U2R. The details are shown in Table 1. Meanwhile, there are another 16 attacks in the test set, and the presence of the extra attacks allows NSL-KDD to evaluate the performance of the machine learning-based model in the face of new attacks.

However, even after eliminating the redundant data from KDD'99, the data distribution in NSL-KDD still leaves some problems because of the objective condition that the attack samples are difficult to collect. The numbers of various samples are shown in Table 2. In the training set, there are few samples of R2L and U2R, which often makes the model undertrained for these two attacks, resulting in a low detection rate. In practical applications, R2L and U2R are very dangerous attacks for the system.

Table 1 Attack types in NSL-KDD

Category	Types in train set	Additional types in test set
Normal	normal	–
DoS	neptune, back, land, pod, smurf, teardrop	Apache2, mailbomb, processtable
Probe	ipsweep, nmap, portsweep, satan	mscan, saint
R2L	warezmaster, warezclient, ftpwrite, guesspassword, imap, multihop, phf, spy	sendmail, snmpguess, snmpgerattack, named, xlock, xsnoop, worm
U2R	Rootkit, bufferoverflow, loadmodule, perl	httptunnel, ps, sqlattack, xterm

Table 2 The distributions of samples in NSL-KDD

Attack category	Train set (KDDTrain)	Test set (KDDTest)
Normal	67343	9711
DoS	45927	7458
Probe	11656	2421
R2L	995	2754
U2R	52	200
Total	125973	22544

2. Experimental environment

This experiment was carried out in windows 10 operating system on a computer configured with Intel i7-6700@3.40 GHz and 32G RAM. The implemented algorithm was written in Python 3.7, using the sklearn library, the tensorflow library and the river library [30].

3. Evaluation indicators

Pre-quential validation [31] is used to evaluate the performance of methods in this paper. Pre-quential validation is a method to evaluate online learning techniques, which is also named test-and-train validation. Through prequential validation, the new coming instances from the data stream will be used to test before used to update model. In terms of evaluation indicators, we evaluate the effectiveness of QARF using the Accuracy, Prediction, Recall and F1-score.

4. Experiment description and results

In terms of data settings, the feature distributions of the training set and test set of NSL-KDD are different, thus a streaming dataset is created by splicing the test set of NSL-KDD after the training set and the point where concept drift occurs is the connection point of the training set and the test set. Samples in the streaming dataset are inputted into the evaluated model one by one to simulate a drift traffic stream. Different numbers of instances are labeled for pre-training in different experiments, and the other instances are regarded as unlabeled traffic data for methods to identify. The preset value of w is 0.5, grad is 0.01 in QARF. Four ARF based algorithms are evaluated and compared:

1) ARF trained with initial labeled samples (Initial learning). In this scenario, the model is only trained in pre-training stage, and the other instances are used for evaluation.

2) ARF trained with all instances (Supervised learning). In this scenario, the whole streaming dataset are labeled and can be learned by the model. The performance of ARF in this scenario can be regarded as the upper limit of performance in the comparisons.

3) QARF. In this scenario, QARF is trained in pre-training stage and then select unlabeled instances to query in the process of detecting the unlabeled traffic stream.

4) ARF with random sampling strategy (Random sampling). In this scenario, ARF is trained in pre-training stage and the query unlabeled samples randomly in the process of detecting the unlabeled traffic stream.

Three experiments are designed to evaluate the performance of QARF.

In experiment 1, we explored the performance of each method with different numbers of instances learned in pre-training stage. In this experiment, we observed the accuracy rate and the number of queried instances of each method, which shows the performance and query cost overhead of each method.

In experiment 2, we observed the performance of each method in multi-classification, especially focusing on the accuracy on detecting R2L and U2R attacks to evaluate the performance of methods in the face of imbalanced traffic streams.

In experiment 3, we explored the performance of QARF in the face of concept drift traffic data. In this experiment, the accuracy of each method near the drift point is recorded to observe how the performance of each method changes when the methods process drift traffic streams.

Further, we compared QARF with other state-of-the-art abnormal traffic detection methods for concept drift traffic in terms of accuracy and query cost.

1) Experiment 1: Evaluation under different number of labeled instances in pre-training

In this experiment, QARF and other methods are evaluated to process traffic streams with different number of labeled instances used in pre-training. The number and ratio of instances for pre-training are shown in Table 3 and the results are illustrated in Figure 3. In addition, we record the metrics and query cost of the methods when the labeled instances are 100, 1000, 10000 and 118813 in Table 4, where 100, 1000, and 10000 represent the cases that rare labeled instances in traffic streams, and 118813 represents the cases that a large number of labeled instances in traffic streams.

As shown in Figure 3, accuracy of Supervised learning is 98.46%, which can be regarded as the upper limit of the performance of methods. When the label budget increases from 29703 to 118813 (30%–80%), the performance of methods is relatively stable, QARF has an accuracy of around 98%, Random sampling has an accuracy of around 97%, and Initial learning has an accuracy of around 95%. However, when the number of labeled instances for pre-training is less than 29703 (30%), the performance of both Initial learning and Random sampling shows a significant decline, while the performance of QARF is still close to Supervised learning. As shown in Table 4, when the instances for pre-training is from 100

Table 3 Number and ratio of labeled instances in rounds.

Rounds	Number	Ratio
1	100	0.07%
2	500	0.34%
3	1000	0.67%
4	5000	3.37%
5	10000	6.73%
6	14851	10%
7	29703	20%
8	44555	30%
9	59406	40%
10	74258	50%
11	89110	60%
12	103961	70%
13	118813	80%

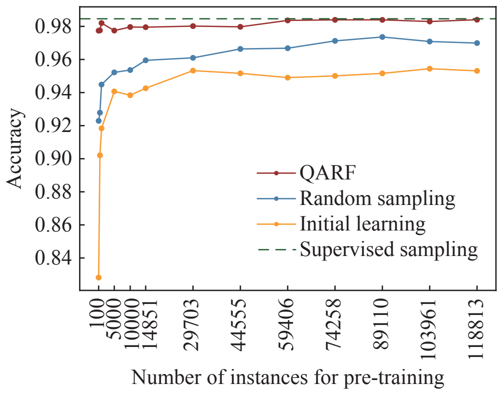


Figure 3 performance of methods under different numbers of labeled instances used in pre-training.

Table 4 Results of methods under learning different numbers of instances in pre-training.

Model	Instances for pre-training	Accuracy	Precision	Recall	F1-score	Query cost
QARF	100	97.74%	97.75%	97.74%	97.69%	2237
	1000	98.20%	98.18%	98.20%	98.15%	2365
	10000	97.96%	97.94%	97.96%	97.88%	2089
	118813	98.40%	98.38%	98.40%	98.37%	1285
Initial learning	100	82.82%	83.39%	82.82%	80.58%	0
	1000	91.84%	89.64%	91.84%	90.29%	0
	10000	93.84%	93.83%	93.84%	92.55%	0
	118813	95.31%	95.32%	95.31%	94.53%	0
Random sampling	100	92.29%	92.12%	92.29%	91.55%	2237
	1000	94.48%	94.58%	94.48%	93.88%	2365
	10000	95.36%	95.14%	95.36%	94.74%	2089
	118813	96.99%	96.84%	96.99%	96.68%	1285
Supervised learning	148517	98.46%	98.44%	98.46%	98.42%	0

to 10000, the accuracy of Initial learning is improved from 82.82% to 93.84%, the accuracy of Random sampling is improved from 92.29% to 95.36%, while QARF is stable at around 98%.

Further, we record the query cost of QARF in different training scenarios to evaluate the label budget of QARF, and the results are as follows in Figure 4.

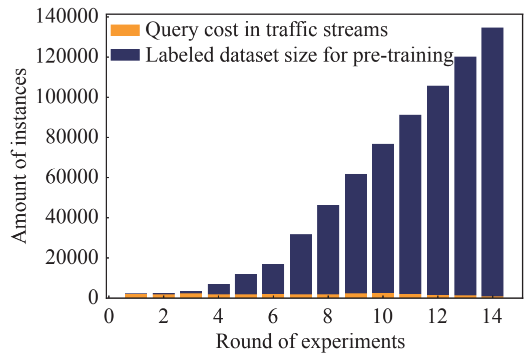


Figure 4 Instances queried and learned by QARF in different rounds of experiments.

As shown in Figure 4, the increase or decrease of the pre-training dataset size does not affect the query cost of QARF. No matter how many instances learned in pre-training, the unlabeled instances queried by QARF is around 2200. Only when the pre-training instances account for 80%, the number of query instances is reduced to 1285. Combining the performance and query cost of methods, QARF obtains high accuracy and low query cost under the condition of rare labeled instances for pre-training, which shows that QARF is more suitable for online malicious traffic detection when the labeled instances in data streams are rare and the label budget is limited.

2) Experiment 2: Comparison on multi-classification

When detection models identify malicious instances in imbalanced traffic, the overall accuracy cannot specifically reflect the performance of methods on rare attacks. Therefore, we record the multi-classification confusion matrix of models to observe the performance of methods on identifying specific attack types, especially the malicious attack types with few instances like R2L and U2R. We focus on the performance of the methods in the condition of insufficient labeled instances for pre-training, thus labeled instances for pre-training are 1000 in this experiment except Supervised learning which uses the whole traffic streams to train. The classification results are shown in the following Figure 5.

As can be seen in Figure 5, all of the four methods get accuracy of 99% on identifying common traffic instances like Normal. However, on identification of uncommon attacks like R2L and U2R, the identifying capabilities of the four methods are not as accurate as the detection of Normal category. Among them, QARF and Supervised learning can still effectively classify R2L and some U2R instances, while Random sampling and Initial learning are severely compromised and nearly impossible to distinguish these two attacks. Further, we record the instances of different categories that methods used in pre-training and queried in identifying traffic streams, which is shown in Table 5. As shown in Table 5, the proportion of instances learned by methods in pre-training is same, but in the identification of traffic streams, instances that queried for ground truth are significantly different. The distribution of instances in pre-training dataset is consistent with the distribution of the whole traffic stream, and pre-training dataset is the only data used to update the model of Initial learning. In the pre-training dataset, Normal and DDoS instances account for the majority, while the number of Probe, R2L, and U2R instances is relatively small, especially U2R instances do not appear in the pre-training stage, which corresponds to the low detection rate of Probe instances and the inability to detect R2L and U2R attacks in Initial learning. Unlike Initial learning, Random sampling queries instances for ground truth from traffic streams randomly, and the query cost of Random sampling keeps consistent with the cost of QARF. The distribution of instances queried by Random sampling conforms to the overall distribution of the whole streaming dataset, which makes the number and proportion of R2L and U2R instances are small, resulting in weakness of Random sampling in identifying R2L and U2R. QARF selects instances through QFC, which captures uncertain instances in traffic streams to query. Because of the uncertainty exhibited by R2L and U2R instances in identification, more U2R and R2L instances are captured by QARF, leading to the distribution of training data more balanced. The balanced distribution of training data helps QARF achieve the significantly improved accuracy in the detection of R2L and U2R and better performance compared with Initial learning and Random sampling. Supervised learning performs best, but it uses the whole traffic streams to train, which means all instances in traffic are queried, leading to the query cost unacceptable.

3) Experiment 3: Evaluation on the drifting data streams

To evaluate the performance of methods on detecting malicious traffic data in drifting data streams, we observe accuracy curve of methods during processing of the streaming dataset, as shown in Figure 6.

As shown in Figure 6 that when concept drift occurs (red barline), the accuracy of all four methods decreases. Among them, the decline rate of Supervised learning is the slowest, the decline rate of QARF is relatively faster but close to Supervised learning, while the performance of Random sampling and Initial learning has dropped significantly after concept drift occurs. For further analysis, we record the distribution of instances queried and learned by methods before and after the drift point, which is shown in Table 6.

As shown in Table 6, Supervised learning has used the whole streaming dataset as labeled instances for

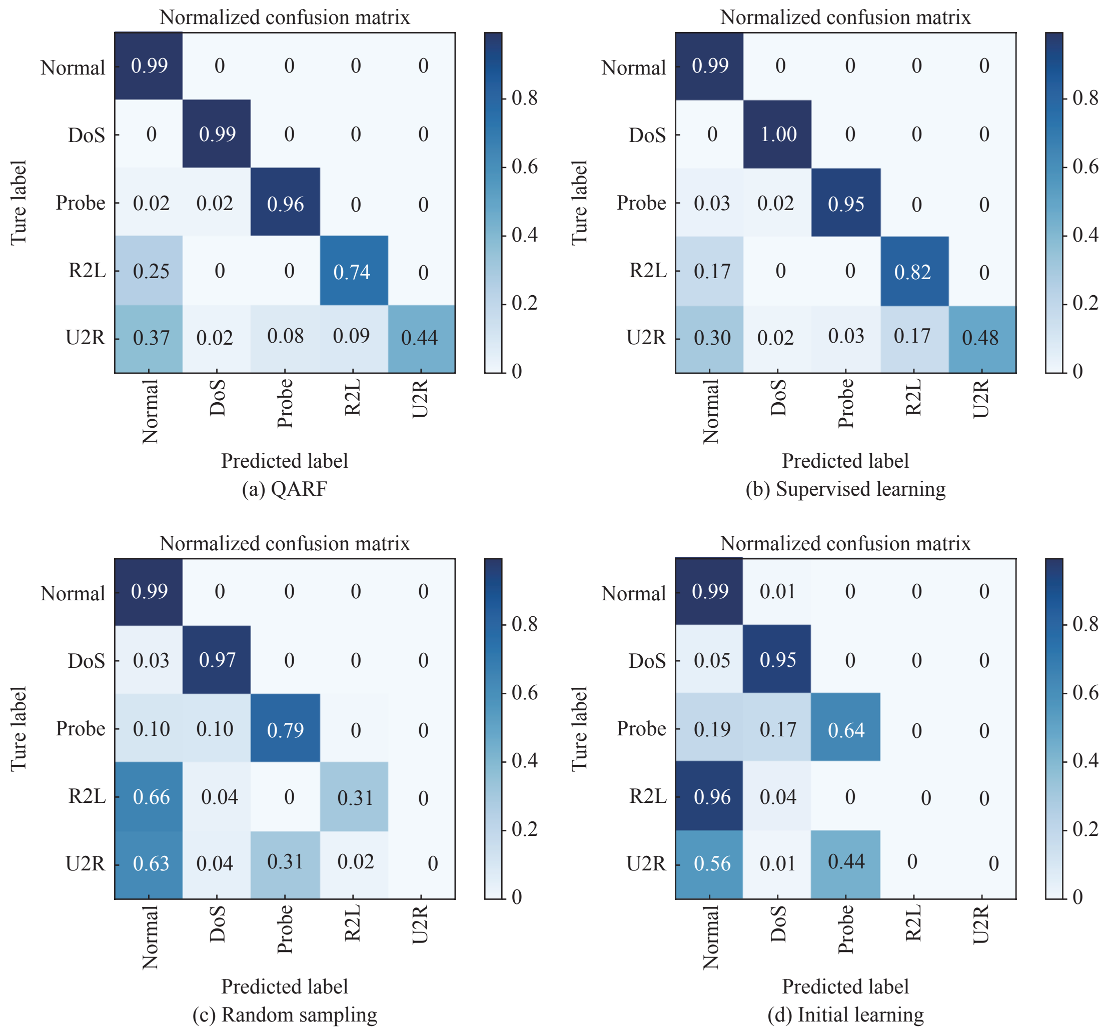


Figure 5 Normalized confusion matrix of methods under learning 1000 labeled samples in pre-training.

Table 5 Number of instances learned by methods in pre-training and label querying.

Instance	QARF		Initial learning		Random sampling		Supervised learning	
	Pre-training	Query cost	Pre-training	Query cost	Pre-training	Query cost	Pre-training	Query cost
Normal	515	165	515	0	515	692	515	76539
DDos	380	253	380	0	380	487	380	53005
Probe	91	549	91	0	91	145	91	13986
R2L	13	268	13	0	13	40	13	3736
U2R	0	50	0	0	0	2	0	252

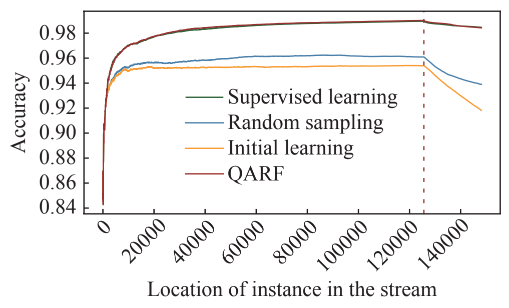


Figure 6 Accuracy of methods under learning 1000 labeled samples in pre-training.

training, including traffic data with concept drift, leading to the smoothest drop in performance. Compared to Random sampling, with the same label budget, QARF queries more samples after concept drift occurs, therefore QARF has lower attenuation on detecting drift traffic data than Random sampling and Initial learning.

Figure 7 shows the distribution of instances learned and queried by QARF and change curve of the threshold value in QFC. The threshold shown in Figure 7 is 500 times magnified on the real threshold in order to show the changes of thresholds and the distribution of query instances in the same figure. The two barlines in Figure

Table 6 Distribution of instances learned by methods.

Method	Pre-training	Before drift	After drift
QARF	1000	1732	544
Supervised learning	1000	124973	22544
Random sampling	1000	1935	340
Initial learning	1000	0	0

7 represent the point where pre-training ends and the point where concept drift occurs, respectively. The two barlines split Figure 7 into three regions, which are pre-training region, smooth flow region, and drift flow region.

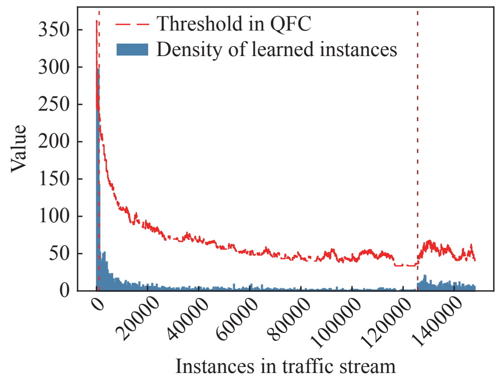


Figure 7 Density of learned samples and curve of threshold in QARF.

As shown in Figure 7, the distribution of the instances learned by QARF and the trend of threshold change curve overlap. From pre-training region to the smooth flow region, the traffic stream is stable, thus the threshold of QARF is constantly decreasing to reduce the query cost. When the threshold tends to converge, the query instances size also gradually converges, and it can be seen from Figure 7 that the query instances size has been in a low query level from the point where streaming data reaches 40000 to the end of smooth flow region. When concept drift occurs, as shown in drift flow region of Figure 7, the threshold increases rapidly, which leads to a rapid increase in the number of queried instances. Meanwhile, the threshold is relatively high in drift flow region, leading to more uncertain instances captured, thus QARF obtains more instances from the drift flow for training so that QARF can update faster to adapt to the drift traffic.

4) Comparison with other methods

In addition, QARF is compared with three other state-of-the-art methods. 1) OFE (online fusion of experts) proposed in Mahmodi *et al.* [14] is a drift aware adaptive method to capture social network-attack in traffic streams through employing online learning algorithms to identify drift in streams. 2) OALEnsemble (online active ensemble framework) proposed in Shan *et al.* [32] is a paired ensemble framework consisting of a stable classifier and a dynamic classifier to react concept drift to raw data streams with labeling small number of instances. In the framework, random sampling strategy and uncertainty strategy are combined to select and label instances for updating both stable classifier and dynamic classifier in online way. 3) IDDAL (intrusion detection with deep active learning) proposed in Ahmed *et al.* [33] is a deep active learning method to achieve intrusion detection in software-defined network, which combines pooling strategy and entropy uncertainty strategy to select instance for training from traffic data. QARF is compared with the above methods on accuracy and percentage of queried instances required for training, and the results are as follows in Table 7.

Table 7 Comparison of QARF and other methods.

Algorithms	Accuracy	Percentage of queried instances
QARF	98.20%	2.26%
OFE [14]	97.02%	100%
OALEnsemble [29]	96.90%	36.19%
IDDAL [30]	94.00%	60.00%

The results in Table 7 show that QARF is an effective approach, outperforming other methods in all indicators.

VI. Conclusion and Future Work

This research aims to effectively detect malicious traffic when the traffic streams are drift, imbalanced and rare labeled. The study contributes to our understanding of malicious traffic detection in complex environment. We proposed queried adaptive random forests (QARF), a malicious traffic detection method based on adaptive random forests (ARF) to achieves adaptive updating when detecting drift streaming traffic. In addition, by employing adaptive sampling strategy, QARF achieves effective training with limited label budget and sensitive to uncommon attacks. We implement a prototype of QARF and evaluate the performance of the model with NSL-KDD dataset. The experimental results show that QARF achieves an accuracy of 98.20% with low query cost for malicious traffic classification. In addition, QARF has better performance by comparing with other representative methods.

Although QARF still works stably after encountering traffic streams with concept drift, it has some performance degradation. QARF is less accurate when detecting traffic flows with concept drift than when detecting the traffic flows before concept drift occurs. However, it is important for malicious traffic detection systems to timely recover from the influence of drift traffic streams. For fast performance recovery, QARF needs to obtain a sufficient amount of representative traffic data to train. In this context, we will conduct further research on adding cluster analysis to adaptive sampling strategy so that more representative instances can be selected by considering the position of instances on the clusters.

Acknowledgements

This work was supported by the Major Scientific and Technological Innovation Projects of Shandong Province (Grant No. 2020CXGC010116) and the National Natural Science Foundation of China (Grant No. 62172042).

References

[1] H. Y. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Applied Sciences*, vol. 9, no. 20, article no. 4396, 2019.

[2] L. Yang, D. M. Manias, and A. Shami, "PWPAE: An ensemble framework for concept drift adaptation in IoT data streams," in *Proceedings of 2021 IEEE Global Communications Conference*, Madrid, Spain, pp.1–6, 2021.

[3] G. Andresini, F. Pendlebury, F. Pierazzi, *et al.*, "INSOMNIA: Towards concept-drift robustness in network intrusion detection," in *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, Virtual Event, pp.111–122, 2021.

[4] Y. Zhang, J. Niu, G. J. He, *et al.*, "Network intrusion detection based on active semi-supervised learning," in *Proceedings of the 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops*, Taipei, China, pp.129–135, 2021.

[5] H. L. Du, Y. Zhang, K. Gang, *et al.*, "Online ensemble learning algorithm for imbalanced data stream," *Applied Soft Computing*, vol. 107, article no. 107378, 2021.

[6] A. Chhabra, T. S. A. Nandyala, and P. Branco, "HEAL: Heterogeneous ensemble and active learning framework," in *Proceedings of the 34th Canadian Conference on Artificial Intelligence*, Vancouver, Canada, pp.1-6, 2021.

[7] C. A. M. S. Teles, C. R. G. V. Filho, and F. da Rocha Henriques, "A black-box framework for malicious traffic detection in ICT environments," in *Handbook of Research on Cyber Crime and Information Privacy*, M. M. Cruz-Cunha and N. R. Mateus-Coelho, Eds. IGI-Global, Hershey, PA, USA, pp.1–20, 2021.

[8] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[9] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, 1987.

[10] A. Javaid, Q. Niyaz, W. Q. Sun, *et al.*, "A deep learning approach for network intrusion detection system, " in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies*, New York City, NY, USA, pp.21–26, 2015.

[11] J. Klein, S. Bhulai, M. Hoogendoorn, *et al.*, "Plusmine: Dynamic active learning with semi-supervised learning for automatic classification," in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, Melbourne, Australia, pp.146–153, 2021.

[12] H. P. Yao, D. Y. Fu, P. Y. Zhang, *et al.*, "MSML: A novel multilevel semi-supervised machine learning framework for intrusion detection system," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1949–1959, 2019.

[13] M. Odiathevar, W. K. G. Seah, and M. Frean, "A hybrid online offline system for network anomaly detection," in *Proceedings of 2019 28th International Conference on Computer Communication and Networks*, Valencia, Spain, pp.1–9, 2019.

[14] E. Mahmodi, H. S. Yazdi, and A. G. Bafghi, "A drift aware adaptive method based on minimum uncertainty for anomaly detection in social networking," *Expert Systems with Applications*, vol. 162, article no. 113881, 2020.

[15] S. Bhatia, A. Jain, P. Li, *et al.*, "MStream: Fast anomaly detection in multi-aspect streams," in *Proceedings of the Web Conference 2021*, Ljubljana, Slovenia, pp.3371–3382, 2021.

[16] M. Jain and G. Kaur, "Distributed anomaly detection using concept drift detection based hybrid ensemble techniques in streamed network data," *Cluster Computing*, vol. 24, no. 3, pp. 2099–2114, 2021.

[17] R. K. Deka, D. K. Bhattacharyya, and J. K. Kalita, "Active learning to detect DDoS attack using ranked features," *Computer Communications*, vol. 145, pp. 203–222, 2019.

[18] W. L. Al-Yaseen, Z. A. Othman, and M. Z. A. Nazri, "Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system," *Expert Systems with Applications*, vol. 67, pp. 296–303, 2017.

[19] A. Ahmim, L. Maglaras, M. A. Ferrag, *et al.*, "A novel hierarchical intrusion detection system based on decision tree and rules-based models," in *Proceedings of the 15th International Conference on Distributed Computing in Sensor Systems*, Santorini, Greece, pp.228–233, 2019.

[20] R. N. Wang, J. L. Fei, M. Zhao, *et al.*, "DA-transfer: A transfer method for malicious network traffic classification with small sample problem," *Electronics*, vol. 11, no. 21, article no. 3577, 2022.

[21] K. D. Lin, X. L. Xu, and F. Xiao, "MFFusion: A multi-level features fusion model for malicious traffic detection based on deep learning," *Computer Networks*, vol. 202, article no. 108658, 2022.

[22] R. Chapaneri and S. Shah, "Enhanced detection of imbalanced malicious network traffic with regularized Generative Adversarial Networks," *Journal of Network and Computer Applications*, vol. 202, article no. 103368, 2022.

[23] H. M. Gomes, A. Bifet, J. Read, *et al.*, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, no. 9-10, pp. 1469–1495, 2017.

[24] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, MA, USA, pp.71–80, 2000.

[25] N. C. Oza, "Online bagging and boosting," in *Proceedings of 2005 IEEE International Conference on Systems, Man and Cybernetics*, Waikoloa, HI, USA, pp.2340–2345, 2005.

[26] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing, " in *Proceedings of the 2007 SIAM International Conference on Data Mining*, Minneapolis, MA, USA, pp.443–448, 2007.

[27] A. Shahraki, M. Abbasi, A. Taherkordi, *et al.*, "Active learning for network traffic classification: A technical study," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 1, pp. 422–439, 2022.

[28] M. Tavallaee, E. Bagheri, W. Lu, *et al.*, "A detailed analysis of the KDD CUP 99 data set," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, ON, Canada, pp.1–6, 2009.

[29] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "Feature selection and classification in multiple class datasets: An application to KDD Cup 99 dataset," *Expert Systems with Applications*, vol. 38, no. 5, pp. 5947–5957,

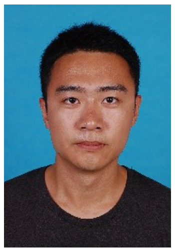
2011.

[30] J. Montiel, M. Halford, S. M. Mastelini, *et al.*, "River: Machine learning for streaming data in Python," *The Journal of Machine Learning Research*, vol. 22, no. 1, article no. 110, 2021.

[31] J. Gama, R. Sebastião, and P. P. Rodrigues, "Issues in evaluation of stream learning algorithms," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, France, pp.329–338, 2009.

[32] J. C. Shan, W. K. Liu, C. X. Chu, *et al.*, "Online active learning with drifted data streams using paired ensemble framework," *ITM Web of Conferences*, vol. 12, article no. 05016, 2017.

[33] U. Ahmed, J. C. W. Lin, and G. Srivastava, "A resource allocation deep active learning based on load balancer for network intrusion detection in SDN sensors," *Computer Communications*, vol. 184, pp. 56–63, 2022.

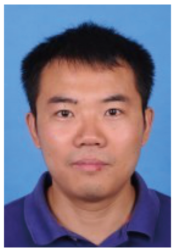


Zequn NIU was born in 1994. He received the B.E. degree in software engineering from Beijing Institute of Technology, Beijing, China. He is a Ph.D. candidate of Beijing Institute of Technology. His research interests include data mining and traffic analysis.
(Email: niuzq@ouchn.edu.cn)

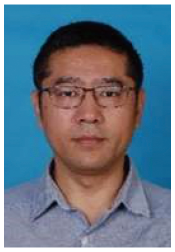
Jingfeng XUE was born in 1975. He is a Professor and Ph.D. Supervisor in Beijing Institute of Technology. His main research interests focus on network security, data security and software security.

Yong WANG was born in 1975. She is an Associate Professor of Beijing Institute of Technology. Her main research interests focus on cyber security and machine learning.
(Email: wangyong@bit.edu.cn)

Tianwei LEI was born in 1993. She received the M.E. degree in software engineering from Beijing Institute of Technology. She is a Ph.D. candidate of Beijing Institute of Technology. Her research interests include software fault and malware analysis.



Weijie HAN was born in 1980. He received the Ph.D. degree from Beijing Institute of Technology. He is currently a Lecture in Space Engineering University. His research interests include malware detection and APT detection.



Xianwei GAO was born in 1978. He received the Ph.D. degree from Beijing Institute of Technology. He has many years of experience in security operation and software engineering in a famous IT enterprise. His research interests mainly focus on artificial intelligence and cyber security.