Received February 11, 2021, accepted March 5, 2021, date of publication March 17, 2021, date of current version March 30, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3066620

Real-Time Network Intrusion Prevention System Based on Hybrid Machine Learning

WOOSEOK SEO[1] AND WOOGUIL PAK[2]
[1]Wookyoung Information Technology, Daegu 41519, South Korea
[2]Department of Information and Communication Engineering, Yengnam University, Gyeongsan 38541, South Korea

Corresponding author: Wooguil Pak (wooguilpak@yu.ac.kr)

This work was supported in part by the National Research Foundation of Korea(NRF) Grant by the Korean Government through the Ministry of Science and ICT (MSIT) under Grant NRF-2019R1F1A1062320, in part by the 2020 Yeungnam University Research Grant, and in part by the Information Technology Research Center (ITRC) Support Program supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP) under Grant IITP-2020-2016-0-00313.

ABSTRACT Recent advancements in network technology and associated services have led to a rapid increase in the amount of data traffic. However, the detrimental effects caused by cyber-attacks have also significantly increased. Network attacks are evolving in various forms. Two primary approaches exist for addressing such threats: signature-based detection and anomaly detection. Although the aforementioned approaches can be effective, they also have certain drawbacks. Signature-based detection is vulnerable to variant attacks, while anomaly detection cannot be used for real-time data traffic. For resolving such issues, this paper proposes a two-level classifier that can simultaneously achieve high performance and real-time classification. It employs level 1 and 2 classifiers internally. The level 1 classifier initially performs real-time detection with moderate accuracy for incoming data traffic. If the data cannot be classified with high probability by the classifier, the classification is delayed until the traffic flow terminates. The level 2 classifier then collects the statistical features of the traffic flow for performing precise classification. Compared to existing techniques, the proposed two-level classification method can achieve superior performance in terms of accuracy and detection time.

INDEX TERMS Intrusion prevention system, intrusion detection system, machine learning, real-time, two-level classifier.

I. INTRODUCTION

Recently, network technology and related equipment have been evolving at a fast rate, and accordingly, the performance and total traffic volume of networks are rapidly increasing. The damage caused by cyber-attacks, however, is also increasing, not merely because the number of cyber-attacks is increasing, but because they have become sophisticated and their variants have been created. Nowadays, it is almost impossible to defend a network completely from malicious hackers [1]. A zero-day attack, which exploits the newly discovered vulnerability of network systems before its solution developed, is one of the most serious cyber-crimes. The network inevitably becomes defenseless until a solution is developed and applied to the system [2]. According to previous work, a zero-day attack lasts for 312 days on average and can last up to 30 months. Moreover, the total number of zero-day attacks can increase up to five times after a new vulnerability has been disclosed [3].

Today, network security systems supporting real-time attack prevention mainly use signature-based methods to detect certain patterns among incoming packets in a way similar to virus scanners [4]–[6]. Real-time detection requires in-line processing algorithms which can detect attacks at the line rate. Although we can increase the detection speed using distributed systems, they can suffer from expensive synchronization overhead, so it is not fundamental solution. Thus, fast detection algorithm is inevitable to implement real-time detection systems.

The signature-based approach has a high accuracy and speed for detecting prior known attacks. However, it is almost impossible to detect unknown attacks such as zero-day attacks, and it is vulnerable to variant attacks that bypass signature-based detection using code obfuscation or encryption. It also has a high burden of keeping the signature database up to date.

The associate editor coordinating the review of this manuscript and approving it for publication was Yassine Maleh.

In contrast to the signature-based approach, the anomaly detection approach observes the statistical characteristics of each flow of network traffic and detects it as an attack if it differs from normal behavior by exceeding the normal range of statistics. Because it needs not a signature database, it does not have any overhead for maintaining a database, and it can be very robust against zero-day or variant attacks. Various machine learning techniques to classify attacks from normal flows are widely adopted to modern network intrusion detection systems (NIDSs) based on anomaly detection.

However, it is very difficult to find fast and accurate machine learning algorithms to process incoming network traffics in real time. In addition, anomaly detection algorithms are designed to classify traffic based on the characteristics of each complete flow, for example, the number of packets transmitted and received during the flow, the number of packets lost, etc., rather than classifying based on every packet. Thus, it only starts determining whether each flow is malicious when the flow has been terminated. As a result, per-flow classification cannot detect attacks until the flow finishes and cannot defend the current network and users under attack.

In this paper, we propose a novel approach to resolve the problems of existing network intrusion detection and defense technologies. It is based on machine learning techniques, but it can be differentiated from other machine learning-based approaches in the following aspects.

A. REAL-TIME ATTACK DETECTION

There are a few previous works that use non-machine-learning-based methods to detect network attacks in real-time. However, they just provide several Gbps throughput, much lower than 100 Gbps. The proposed approach can support real-time attack detection at up to 100 Gbps traffic rates even though it uses a machine learning technique. As mentioned earlier, machine learning became popular in NIDS due to its ability to detect unknown attacks. However, machine learning-based algorithms are too slow to handle many Gbps of traffic, therefore they cannot be deployed in high throughput networks. To solve this problem, we suggest two-levels of classifiers: one for per-packet, the other for per-flow detection.

B. HIGH ATTACK DETECTION ACCURACY

Because network attacks have widely varying types and behaviors, it cannot be effective to adopt a single approach to detect all kinds of network attacks. Therefore, the proposed approach uses a two-level attack detection. We can detect some attacks by inspecting just a few packets, but for others such as distributed denial of service (DDoS), we need to observe the network-wide behavior of flows to detect the attack. Hence, our approach not only analyzes individual packets but also uses in-flow and inter-flow statistics flows to increase the accuracy of detecting attacks. We use two different classifiers simultaneously, achieving very high detection accuracy compared to existing work.

This paper is composed of five parts. In Section II, we briefly introduce related work and results. In Section III, we propose a new intrusion prevention system (IPS) with real-time attack detection and high accuracy. In Section IV, we analyze the performance of the proposed approach and show the results of comparing some selected recent works. Finally, we conclude our paper in Section V.

	Single classifier	Multiple classifier
Packet based	[4][7]	[5]
Session based	[9][12, 13] [15-17][24-29]	[8][30, 31]

FIGURE 1. Classification of ML based NIDS according to classifier number and feature type.

II. EXISTING WORK

Machine learning-based network intrusion detection systems (NIDSs) have been continuously developed. The early NIDS employed a simple structure with a single machine learning algorithm. However, there was a limit to detecting various network attacks accurately with a single machine learning algorithm, and therefore, NIDS research using a combination of various machine learning algorithms has been actively underway.

The machine-learning-based IDS is divided into a packet-based method that uses packet data and a session-based method that uses session data to train the model, based on the type of data used. The packet-based method converts raw packet data into machine learning features and then performs learning and classification using conventional machine learning algorithms such as convolutional neural networks (CNN). It does not need to extract features before training, so it simplifies training procedures without manual intervention.

The session-based method creates data to generate features of the session inside the IDS when processing packets of a new session, and it updates these session data whenever it processes the packets of the session. After processing the last packet of the session, it generates features for the session by processing the final updated data, and then, these features are used in machine learning. The biggest advantage of this method is that instead of using a large number of packets of a session, it uses a small number of statistical values for the session; thus, the number of features used for training and classifying is very small. As this is very effective for fast training and fast classification, it can handle large network traffics.

Because machine learning-based NIDS can be classified into single and multiple classifier-based NIDS according to the number of algorithms used, machine learning-based NIDS can be classified into four types, as seen in Fig. 1. We examine the main research on each method.

A. PACKET-BASED SINGLE-MACHINE LEARNING ALGORITHM METHOD

This method learns and classifies packet data through a single machine learning algorithm. It has the advantage of detecting malicious code in packet payload data.

However, because individual packets are analyzed independently to determine whether an attack has occurred, the conventional signature-based NIDS belongs to this method [4]. Therefore, this method has the advantage of being able to detect an attack when it occurs in real-time. However, it is vulnerable to zero-day attacks, variant attacks, and bypass using packet fragmentation to avoid detection. Recently, an attack detection method that collects multiple packets of a session rather than a single packet has been proposed to overcome these weaknesses.

B. PACKET-BASED MULTIPLE-MACHINE LEARNING ALGORITHM METHOD

This method detects attacks using multiple machine learning algorithms rather than a single algorithm for packet-based data. Hence, it can perform training and classification more effectively than a single algorithm. However, as with the packet-based single machine learning algorithm method, a large number of features (over thousands) should be generated from packets for the training machine learning algorithm. Therefore, it has the disadvantage of being difficult to use in large networks because of the very slow training and classification speed [5].

C. SESSION-BASED SINGLE-MACHINE LEARNING ALGORITHM METHOD

Instead of using packets, this method extracts features for a session and applies them to a single algorithm for training and classification [24]–[29]. It is one of the most common studies in the early machine learning-based literature. As it does not use packet data and generates a fixed number of features regardless of the number of packets or packet size belonging to a session, the memory usage is very low. In particular, as it processes a small number of features (e.g., less than a hundred) using a single algorithm, the training and classification speed can be very fast. Thus, it is applicable to large networks with heavy traffic. However, it is difficult to provide high detection rate for various attack types using a single algorithm. Further, as features are generated after the session ends, an attack has most likely been already completed when it is detected. This is a critical limitation of this category.

D. SESSION-BASED MULTIPLE-MACHINE LEARNING ALGORITHM METHOD

This method performs training and classification by using features for a session while simultaneously using various classification algorithms. Among this category, well-known types are ensemble and multi-layered methods [30], [31]. The ensemble method simultaneously applies several algorithms and integrates the results. It can improve the detection performance by using several algorithms for various classes. The multi-layered method executes the next algorithm based on the result after executing a specific algorithm. In most cases, it makes use of unsupervised learning and supervised learning together. For example, it can perform partitioning using k-nearest neighbor (kNN) and applying a decision tree (DT) to each partition. The session-based multiple machine learning algorithm method has a very high classification performance. However, in reality, it is difficult to support real-time attack detection because it is impossible to process network traffic in real-time because of the very high computational cost caused by multiple machine learning algorithms. Further, because the overall implementation cost is high, it is difficult to apply to a real network security system.

Various approaches have been adopted to increase the detection accuracy and speed. However, as of now, there is very limited research on real-time intrusion prevention systems that can detect and defend attacks in real-time; thus, there is an urgent need for research on this topic.

III. MACHINE-LEARNING-BASED REAL-TIME IDS

As discussed in Section II, existing NIDSs are struggling to increase both detection speed and accuracy simultaneously, but no practical solutions are available. As a solution to this problem, we need a different strategy than the signature-based or anomaly-based approaches. For achieving accurate attack detection, using anomaly-based machine learning is inevitable, due to the limitations of using pre-configured signature databases, e.g., the low detection accuracy of zero-day attacks. Hence, we should aggressively adopt machine learning techniques in our NIDS. However, machine learning-based approaches are considerably slower than signature-based ones. As a result, real-time detection becomes almost impossible with machine-learning-based NIDS, making it difficult to defend against various attacks.

A. MOTIVATION

Basically, classification accuracy and classification time tend to be proportional [32]. For example, DT can be classified quickly by using a single tree, but classification performance may be degraded by instability and dependency on a particular set of features. On the other hand, random forest (RF) classifies more accurately than DT for most cases, but the classifying speed is much slower that DT. Thus, if we take advantage of the fast but less accurate classifier and the slow but more accurate classifier, we can develop a fast and accurate classifier. Let us consider the case of trying to classify the incoming traffic using the fast classifier at first. In this case, we need to check and evaluate the reliability of the result for each classification. If it has very high reliability, i.e., high score, it would be good to believe the result and process the traffic according to it. If the reliability is low, it is necessary to ignore the result and to run the slow but more accurate classifier. In this case, the most important design factor is that the fast classifier should handle as much traffic as possible, enabling the slow classifier to handle the remaining traffic without queueing. If this condition cannot be met, the speed of the slow classifier becomes a serious bottleneck of fast classification.

In order to satisfy such a condition, it implies that the distribution of traffic according to the difficulty of classification must be proportional. Fortunately, even a simple DT can classify normal and attack traffic with a fairly high accuracy. Therefore, it is reasonable to assume that the amount of traffic with high classification difficulty is relatively small.

For high speed classification, the speed of the classification algorithm itself is the most critical factor. However, feature extraction also consumes much time. Hence, we need to design the first classifier so that required features can be extracted easily without any special processing overhead. In addition, the number of features important because the number of features increases affects the classification speed, therefore, feature selection is necessary.

On the other hand, the second classifier needs to obtain important features even though it takes a much time for accurate classification. it will be essential to use session features completely describing the entire session, which is acquired by waiting until the session finishes.

Now, based on this conclusion, we will explain how the proposed approach was designed to provide real-time detection and accurate detection simultaneously.

B. TWO-LEVEL CLASSIFIER

The proposed approach is a two-level scheme using two different classifiers simultaneously. To provide both high speed and accuracy in attack detection, the proposed scheme operates as follows: the level 1 classifier handles receiving packets at line speed, only if it can precisely classify them as normal or attack. The level 2 classifier handles any remaining packets which were not classified as attack in level 1, and classifies them with a slow but exact classifying algorithm because level 2 classifier has no constraint or burden of real-time processing.

The level 1 classifier adopts a classification algorithm optimized for classification speed, even though it sacrifices accuracy. This classifier plays a vital role in supporting line speed packet processing and in reducing the burden of the level 2 classifier. Thus, the level 1 classifier should process as many incoming packets as it can, but it also should not process them when it cannot do so with high accuracy. It sensibly determines if the packet is classified in level 1 or postponed to the level 2 classifier.

The level 2 classifier will handle packets unprocessed in the level 1 classifier because they could not be accurately classified. By handling only a small portion of the entire traffic, the burden of real-time processing is considerably less, which allows sophisticated and time-consuming classification for accurate detection. To achieve this goal, the level 2 classifier uses the statistical features of each flow for the whole lifespan instead of packet data. Although detection is delayed until the flow finishes, it is possible to effectively detect attacks that cannot be detected by analyzing only some of the packet data.

In addition to the unique two-level structure, the proposed approach trains classifiers, as shown in Fig. 2, to improve the performance of the level 2 classifier. It trains the level 1 classifier using the entire training dataset, as in other existing machine-learning-based classifiers. To train the level 2 classifier, we have two options: First, it can use the entire training dataset in the same way as the level 1 classifier. Second, it is possible to train the level 2 classifier using only data that the level 1 classifier cannot classify with high accuracy, because the level 2 classifier does not need to process the traffic that could be determined accurately by the level 1 classifier. This training can improve the detection accuracy of level 2 classifiers by reducing unnecessary training data.

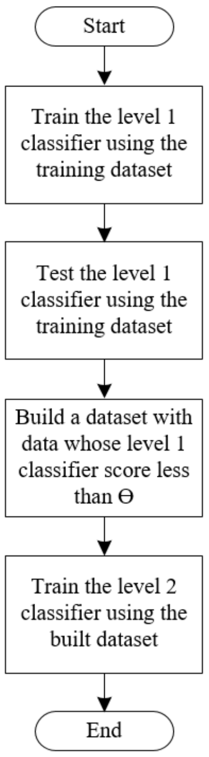
Start
Train the level 1 classifier using the training dataset
Test the level 1 classifier using the training dataset
Build a dataset with data whose level 1 classifier score less than Θ
Train the level 2 classifier using the built dataset
End

FIGURE 2. Procedure of the proposed two-level classifier.

Based on the second option, the proposed approach creates a new dataset to train the level 2 classifier in the following way. First, it trains the level 1 classifier using the entire dataset, and then the classifier classifies each data entry of the training dataset. If the score of the classification result exceeds the predefined threshold value called minimum level 1 classification score ($\ominus$), the data entry is excluded from the training dataset of the level 2 classifier. Therefore, the level 2 classifier is trained using the dataset consisting only of the remaining data entries whose score is less than $\ominus$.

After training both the level 1 and 2 classifiers is complete, it is ready to classify the actual traffic. Fig. 1 shows how the proposed approach classifies traffic using two-level classifiers. As described above, the level 1 classifier must process traffic at high speed. To enable this, the level 1 classifier should be designed to use features that can be generated simply and quickly. The proposed approach builds features from only the first data packet of each flow to determine whether the flow is attack or normal. If the packet is classified as an attack, with a score higher than $\ominus$, by the primary classifier, the packet is discarded, and the flow is blocked. Otherwise, the packet is forwarded, and the flow is allowed

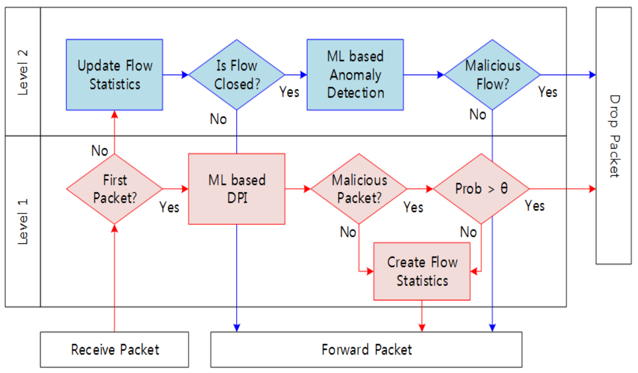

FIGURE 3. Two-level classification procedure of the proposed classifier.

but monitored. For monitoring the flow, statistics of the flow are generated on receiving the first packet and updated whenever it receives a packet belonging to the flow until the flow is terminated. When the flow ends, the level 2 classifier uses the monitoring data as features for machine learning, and the flow is evaluated to detect an attack. Features for the second classifier will be described later. The overall procedure of the proposed algorithm is shown in Fig. 3.

C. LEVEL 1 PACKET-BASED CLASSIFIER

As described above, the proposed approach uses only the first data packet of each flow to determine whether the flow is an attack or a normal flow. The flow is basically composed of multiple packets; therefore, we can significantly increase the number of flows processed in one second when using only the first data packet of each flow compared to the existing work that processes all packets of the flow. However, the detection accuracy of our approach is lower due to the lack of information.

Several works analyzing packet data instead of flow statistics already exist. For example, HAST-IDS[1] uses all packet data and converts each byte of data through one-hot encoding [6]. It is known to achieve a very high classification accuracy, but it can suffer from very large features generated by one-hot encoding. For example, it will have 25,600 features for a 100-byte packet because one-hot encoding generates 256 features for each byte. Such a considerable feature size is a serious burden to achieving high classification speed.

For fast classification, our proposed algorithm uses each byte value as a feature without one-hot encoding. This approach inevitably causes reduced classification accuracy, but such feature generation significantly reduces the total number of features used for training. The proposed approach compensates for the decreased performance with the level 2 classifier. However, it can be insufficient for handling 100-Gbps line speeds. Therefore, we perform feature selection to choose only some features to reduce the size of the feature set further. Feature selection helps both classification speed and accuracy because it eliminates unnecessary or less important features. In addition, source and destination IP addresses in the IP header are excluded from the feature set. If an IP address is included in the set, the classifier can determine an attack using specific server and host IP addresses, so the classifier has high dependency on some specific flow. Excluding IP addresses from features helps the classifier to get better trained without severe skewness. Fig. 4 shows the difference in feature generation between one-hot encoding based on existing work and the proposed approach.

[1]Hierarchical spatial-temporal IDS

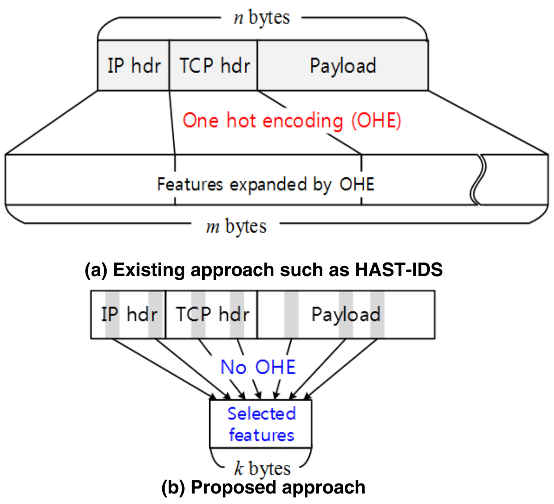

FIGURE 4. Comparison of feature selection for existing and proposed approaches. Selected data are noted in grey.

D. LEVEL 2 FLOW-BASED CLASSIFIER

The level 1 classifier tries to detect attacks based on the first data packet of each flow. However, for some network attacks such as denial-of-service, it will be better to examine intra-/inter-flow statistical information rather than a packet to detect attacks exactly [10]. To achieve this end, the proposed level 2 classifier generates and uses flow-specific features in addition to packet-based features for detecting malicious flows. The list of all flow features is shown in Table 14 in the Appendix. Some features can be generated from packets, but others, which are noted in grey, can be generated only when the flow ends. In the list, we have 46 features in total, and one-hot encoding is performed for some features, i.e., protocol, session state, and service, contrary to our level 1 classifier, resulting in 231 features used for machine learning. Compared to the level 1 classifier, the number of features used in the level 2 classifier is greater, and feature generation takes a long time waiting until the flow being monitored finishes. This makes the classification speed of the level 2 classifier quite low for supporting 100 Gbps. Therefore, we need to design our IDS such that the level 1 classifier should handle most of the traffic and leave an indeterminable and very small portion of the traffic to the level 2 classifier.

For training the level 2 classifier, we can use the same training dataset used for the level 1 classifier. In this case, the dataset includes data that was classified with a score larger than $\ominus$ by the level 1 classifier. Because such data never reach the level 2 classifier, they are not useful for the

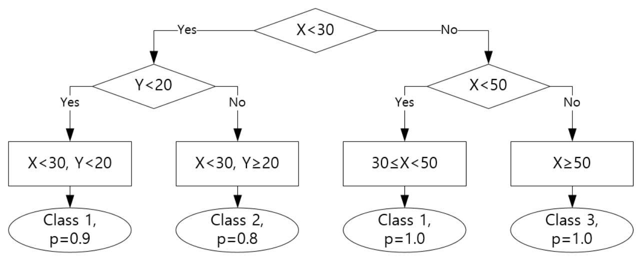

FIGURE 5. Example of DT with two features, X and Y, where each feature is 8 bit-field.

TABLE 1. Policy set obtained from DT example in Fig. 5.

Rule ID	Condition	Action
1	$0 \le x < 30,\ 0 \le y < 20$	Class 1, p=0.9
2	$0 \le x < 30,\ 20 \le y \le 255$	Class 2, p=0.8
3	$30 \le x < 50$	Class 1, p=1
4	$50 \le x \le 255$	Class 3, p=1

classifier. Therefore, we train the level 2 classifier using a sub-dataset that only includes the data whose scores from the level 1 classifier are less than $\ominus$.

E. PACKET ORIENTED FAST CLASSIFIER

As classifiers based on machine learning are general-purposed, they can be used in various environments, achieving a high classification accuracy with moderate classification speed. However, their speed is not enough to process incoming network traffic in real-time. On the other hand, packet classifiers have been developed for a long time to handle firewall policies and access control lists at high speed. Such classifiers can deliver more than 3 M packet classifications per second, which can support network traffic in the tens of Gbps. However, this kind of classifier cannot be used for our purpose because packet classifiers cannot support any learning process and it passively constructs search tables using a pre-defined policy set. Thus, we cannot directly use a packet classifier to boost classification speed.

To solve this problem, we propose a novel approach that builds DT at first and extracts a static policy set from training results. For example, Fig. 5 shows a DT after training whose leaf node has a condition for reaching the node, matching class ID, and matching probability.

The policy of the packet classifier consists of policy priority, matching condition, and action. We can get such information from all leaf nodes in the DT. Matching class ID and probability are saved as actions. We can ignore policy priority because the matching conditions of all leaf nodes of the DT are disjoint. Thus, the policy priority has no effect on the packet classifier. For this reason, we simply set the leaf node ID as the policy priority. By doing so, we can create a policy set for packet classifiers from the DT, as shown in Table 1. Although we use the packet classifier based on DT instead of DT, the searching results are the same as for DT. As a result, we can increase classification speed significantly without any loss of detection accuracy.

IV. PERFORMANCE EVALUATION

For analyzing the performance of the two-level real-time classification algorithm proposed in this research, we evaluated and compared the performance using various existing competitive algorithms.

A. EVALUATION ENVIRONMENT

We compared our algorithm with existing ones, such as DT, RF, and HAST-IDS [11], [12], [5]. The environment of the performance evaluation was as follows. A Jupyter server running on Intel Xeon E5-2640 v4 with two GeForce TITAN Xp graphics cards was used for evaluation. Scikit-learn and Weka libraries were used to measure classification performance and time except for HAST-IDS, which was implemented with CNN using the TensorFlow library. Only HAST-IDS uses GPU and CPU simultaneously, and other algorithms run on only CPU.

For comparison, we used sub-dataset created on January 22, 2015 from the UNSW-NB15 dataset [19]. We added the same class data from the dataset on a different date to the evaluation dataset to avoid the minority class problem by increasing the size of the class. We also removed some classes such as worms, backdoors, and analysis from the dataset because the total size of samples was too small even after balancing the dataset. Therefore, we evaluate the performance using seven classes, one for normal and the others for attack classes.

We also used CICIDS2017 dataset which include various DoS and DDoS attacks [30]. We removed Heatbleed and Infiltration among twelve classes because they have a too small data size.

For the level 1 classifier, we generated the features using only the first 100 bytes of the first data packet of each flow except for the source and the destination IPs. If a packet size was smaller than 108 bytes, we added some null values as padding. To build the features of the level 2 classifier, we used 46 features, which expanded to 231 features after one-hot encoding. We normalized all features to the range 0 to 1. We used the Pearson Correlation as a feature selection algorithm.

Each dataset includes 24,225 and 1,009,809 samples. It was divided into training and test datasets in the ratio of 6:4, so the size of total flows in test datasets were 9,692 and 302,966. Tables 2 and 3 show the size of data samples used for training and testing for each class of each dataset.

B. PARAMETER CONFIGURATION

Before the comparative experiment, we needed to find the best parameters for the proposed algorithm to achieve the highest accuracy and the fastest classification through pre-experiments. To find the optimal parameters, we measured the classification performance according to the combination of the feature size and classification algorithm for the level 1 classifier and the classification algorithm for the level 2 classifier. We also perform the measurement as $\ominus$, i.e., minimum level 1 classifier score increases.

TABLE 2. Dataset sizes of training and testing for UNSW-NB15.

Class	Train	Test	Total	Ratio
Normal	12,896	8,597	21,493	88.7
Exploits	300	201	501	2.1
Reconnaissance	126	85	211	0.9
DoS	192	127	319	1.3
Generic	513	344	857	3.5
Shell code	77	52	129	0.5
Fuzzers	429	286	715	3
Total	14,533	9,692	24,225	100

TABLE 3. Dataset sizes of training and testing for CICIDS2017.

Class	Train	Test	Total	Ratio
Normal	318,340	136,250	454,590	45
Bot	1,391	575	1,966	0
DDoS	89,506	38,514	12,8020	13
DoS GoldenEye	7,190	3,103	10,293	1
DoS Hulk	161,681	69,280	230,961	23
DoS Slowhttptest	3,816	1,613	5,429	1
DoS slowloris	4,049	1,744	5,793	1
FTP-Patator	5,528	2,410	7,938	1
PortScan	111,227	47,695	158,922	16
SSH-Patator	4,115	1,782	5,897	1
Total	706,843	302,966	1,009,809	100

For level 1 and 2 classifiers, the size of total features was increased by 10 from 10 to 100 and from 10 to 230, respectively.

There are many kinds of classification algorithms, but we considered only DT and RF for the classification algorithm of the proposed approach in our performance evaluation. DT is simple, but it has the advantage of fast classification speed. However, it can have low accuracy and over-fitting in classification. On the other hand, RF can achieve very high accuracy without a serious over-fitting issue. Because it internally uses multiple DTs, its classification speed is lower than DT's, but it is still fast compared to other classification algorithms, such as DNN and kNN. For these reasons, we selected the two algorithms as candidate algorithms for level 1 and 2 classifiers in our proposed algorithm. To maximize the strength of each algorithm, we apply the same algorithm to the level 1 and 2 classifiers.

It is very important to select a value of $\ominus$ that will ensure high classification performance. To determine the optimal $\ominus$, we measured the classification results as the score was increased from 0 to 1 by 0.1, and then we chose the $\ominus$ that achieved the best result. In addition to multi-class classification, we also conducted binary-class classification where all six attacks are regarded as one attack during training and classification to evaluate the classification performance under various situations.

Tables 4 and 5 show the chosen optimal parameter values for each combination of algorithms for each dataset. We can see that it achieves a higher classification accuracy when RF is used than when DT regardless of multi/binary class classification for UNSW-NB15. We also see that RF still shows the higher performance for CICIDS2017 even though the differences are marginal.

TABLE 4. Optimal parameters and corresponding classification performance according to the classification algorithm with multi and binary class classification for UNSW-NB15.

Classification type	Multi class		Binary class	
Algorithm	Proposed (DT)	Proposed (RF)	Proposed (DT)	Proposed (RF)
# of Level 1 features	50	60	50	80
# of Level 2 features	40	230	90	200
Threshold	0.9	0.6	0.9	0.6
Accuracy	95.8	96.2	97.9	98.1
Precision	96.3	96.6	98.1	98.2
Recall	95.8	96.2	97.9	98.1
F1-score	95.9	96.3	98	98.2

TABLE 5. Optimal parameters and corresponding classification performance according to the classification algorithm with multi and binary class classification for CICIDS2017.

Classification type	Multi class		Binary class	
Algorithm	Proposed (DT)	Propose d (RF)	Proposed (DT)	Proposed (RF)
# of Level 1 features	20	70	20	10
# of Level 2 features	70	10	40	40
Threshold	1.0	60	1.0	1.0
Accuracy	99.80	99.82	99.98	99.98
Precision	99.81	99.76	99.98	99.98
Recall	99.79	99.88	99.98	99.98
F1-score	99.80	99.82	99.98	99.98

C. EVALUATION RESULTS AND ANALYSIS

For fair performance comparison, we found the optimal configuration for each algorithm. We used the optimal number of features obtained from each flow for the best F1-score of DT and RF. For HAST-I, we used 100- and 300-byte packet data, which showed the best classification speed and the highest F1-score, respectively. We use the notation HAST-I (N), which means N-byte packet data is used for HAST-I for the proposed algorithm.

1) COMPARISON OF THE PERFORMANCE FOR MULTI-CLASS CLASSIFICATION

Table 6 shows the result of comparing average classification performance in multi-class classification for UNSW-NB15. Our proposed algorithm using RF shows the highest performance in accuracy, precision, recall, and F1-score. Even the proposed algorithm using DT shows the second-highest performance of all metrics. HAST-I shows the worst, although the performance gap is less than one percent point. It reflects that the packet-based approach relying on some fixed-length packet data is not an efficient approach for detecting network intrusions.

TABLE 6. Average multi-class classification results of proposed and competitive algorithms for UNSW-NB15.

Algorithm	Accuracy	Precision	Recall	F1-score
Proposed(DT)	95.8	96.3	95.8	95.9
Proposed(RF)	96.2	96.6	96.2	96.3
HAST-I(100)	92.2	93.6	92.2	92.3
HAST-I(300)	92.6	93.9	92.6	92.6
DT	94.2	93.0	94.2	93.4
RF	94.2	95.2	94.2	94.1

TABLE 7. Average multi-class classification results of proposed and competitive algorithms for CICIDS2017.

Algorithm	Accuracy	Precision	Recall	F1-score
Proposed(DT)	99.98	99.81	99.79	99.80
Proposed(RF)	99.89	99.76	99.88	99.82
HAST-I(100)	99.85	99.57	99.82	99.69
HAST-I(300)	99.85	99.53	99.82	99.68
DT	99.94	99.67	99.64	99.65
RF	99.93	99.58	99.64	99.61

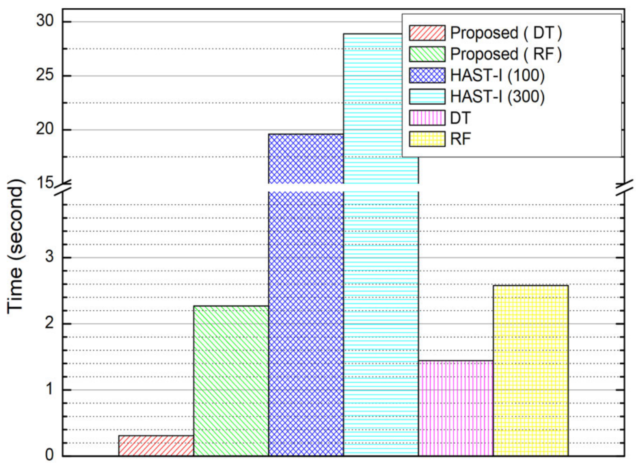

FIGURE 6. Total multi-class classification time taken to classify the entire testing dataset of UNSW-NB15.

Table 7 shows the result of comparing average classification performance for CICIDS2017. CICIDS2017 dataset is well-known for high classification result, so we can see that classification performances are very high regardless of algorithms. However, our proposed algorithm using RF shows the highest performance for precision, recall, and F1-score. HAST-I shows lower performance than ours but better than RF and DT, so it confirms that the proposed approach is efficient for detecting network intrusions.

Fig. 6 shows the total classification time for processing all 9,692 flows of UNSW-NB15. The proposed algorithm using DT shows unbeatable classification speed compared to other approaches. It reaches 60- and 90-times faster classification compared to CNN-based HAST-I (100) and HAST-I (300). It also shows 4.7 times faster classification compared to DT, which is the fastest competing approach.

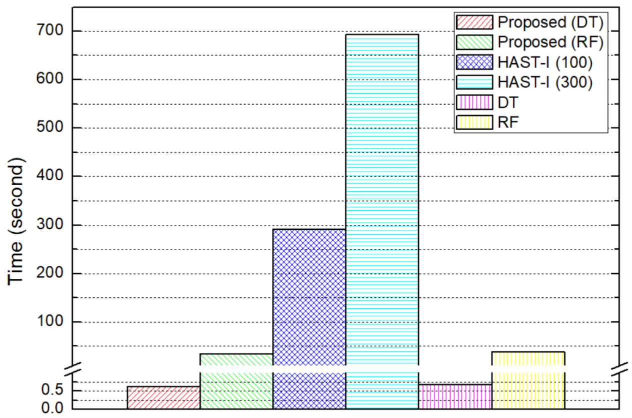

FIGURE 7. Total multi-class classification time taken to classify the entire testing dataset of CICIDS2017.

TABLE 8. Number and the rate of total flow processed by level 1 and level 2 classifiers for each class data when the proposed algorithm using RF is applied on multi-class classification for UNSW-NB15.

Class	Level 1 (%)	Level 2 (%)	Total
Normal	8,563 (99.6%)	34 (0.4%)	8,597
Exploits	117 (58.2%)	84 (41.8%)	201
Reconnaissance	67 (78.8%)	18 (21.2%)	85
DoS	70 (55.1%)	57 (44.9%)	127
Generic	331 (96.2%)	13 (3.8%)	344
Shellcode	21 (40.4%)	31 (59.6%)	52
Fuzzers	260 (90.9%)	26 (9.1%)	286
Total sum of attacks	866 (79.1%)	229 (20.9%)	1,095
Total	9,429 (97.3%)	263 (2.7%)	9,692

Through these experiments, we can see that only our approach can process tremendous incoming traffic in modern networks. The proposed algorithm using RF is about 8 times slower than using DT, but it is also the fastest except for existing DT.

Fig. 7 shows that the total classification time for processing all flows of CICIDS2017. As you can see, it shows the almost the same result compared to Fig. 6. Our approach with DT shows the highest speed and HAST-I still shows the lowest one even though HAST-I leverages GPU to boost the classification speed.

Table 8 depicts the numbers of data classified in the level 1 and 2 classifiers of the proposed algorithm with the optimal detection-rate parameters in multi-class classification for UNSW-NB15. 97.3% of incoming packets are classified in the level 1 classifier, and only 2.7% are classified in the level 2 classifier. Therefore, we can see that the total classification performance of the proposed algorithm is mostly determined by the level 1 classifier.

Table 9 shows the result for CICIDS2017. In the UNSW-NB15, the level 2 classifier classifies sessions more than the level 1 classifier for some classes such as shellcode. However, for CICIDS2017, the level 1 classifier absolutely classifies much more sessions than level 2 regardless of class types.

TABLE 9. Number and the rate of total flow processed by level 1 and level 2 classifiers for each class data when the proposed algorithm using RF is applied on multi-class classification for CICIDS2017.

Class	Level 1 (%)	Level 2 (%)	Total
Normal	181,525 (99.98%)	32 (0.02%)	181,557
Bot	752 (99.6%)	3 (0.4%)	755
DDoS	51,467 (99.96%%)	21 (0.04%)	51,488
DoS GoldenEye	4,111 (99.9%)	5 (0.1%)	4,116
DoS Hulk	92,197 (99.999%)	1 (0.001%)	92,198
DoS SlowHttpTest	2,136 (99.9%)	3 (0.1%)	2,139
DoS Slowloris	2,329 (99.96%)	1 (0.04%)	2,330
FTP-Patator	3,177 (99.97%)	1 (0.03%)	3,178
PortScan	63,772 (99.997%)	2 (0.003%)	63,774
SSH-Patator	2,366(99.87%)	3(0.13%)	2,369
Total sum of attacks	222,307 (99.98%)	40 (0.02%)	222,347
Total	403,832 (99.98%)	72 (0.02%)	403,904

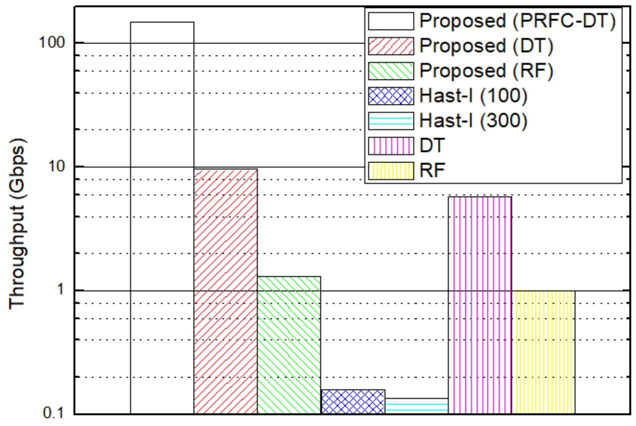

FIGURE 8. Total traffic classification performance on multi-class classification for UNSW-NB15.

By considering the total traffic size of flows for the entire test set, we can calculate the traffic processing rate in Gbps for each dataset and the results are shown in Figs. 8 and 9. In Fig. 8, the proposed algorithm using PRFC-DT represents classification results that are obtained by replacing the level 1 classifier DT with one of the existing fastest packet classification algorithms, PRFC [23] for UNSW-NB15. While the existing approaches show 157 Mbps to 5.8 Gbps throughput, the proposed algorithm shows 1.3 Gbps for using RF and 9.6 Gbps for using DT. Moreover, when the level 1 classifier is replaced with PRFC, the traffic processing performance was significantly improved to 149 Gbps without decreasing the attack detection accuracy.

Fig. 9 also shows the very similar results for UNSW-NB15. From this result, we can confirm that our approach is very effective to support real-time intrusion detection and high classification accuracy simultaneously.

2) COMPARISON OF THE PERFORMANCE FOR BINARY-CLASS CLASSIFICATION

Tables 10 and 11 list the result of comparing the binary class classification performance for each algorithm with UNSW-NB15 and CICIDS2017. As with multi-class classifications for UNSW-NB15, binary class classification results also show that the proposed approach yields the highest detection accuracy for all performance metrics regardless of dataset types. In particular, the proposed approach shows a significant increase in binary class classification performance compared to multi-class classification. However, competing algorithms such as HAST-I show a decreased detection accuracy regardless of packet size for UNSW-NB15. Interestingly, HAST-I becomes inferior to DT and RF for binary class classification of CICIDS2017 while HAST-I shows better performance than DT and RF for multi-class classification of CICIDS2017.

Figs. 10 and 11 shows comparisons of binary class classification speeds for UNSW-NB15 and CICIDS2017. From multiple classification rate comparisons, the proposed algorithm using DT is the fastest and HAST-I shows the slowest for all cases. Noting that proposed one does not rely on GPU but on

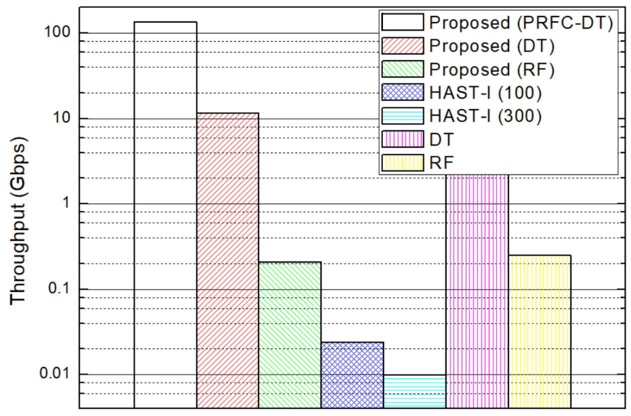

FIGURE 9. Total traffic classification performance on multi-class classification for CICIDS2017.

TABLE 10. Average binary class classification results of proposed and competitive algorithms for UNSW-NB15.

Algorithm	Accuracy	Precision	Recall	F1-score
Proposed (DT)	97.9	98.1	97.9	98.0
Proposed (RF)	98.1	98.2	98.1	98.2
HAST- I (100)	88.5	94.3	88.5	90.0
HAST- I (300)	89.5	94.6	89.5	90.8
DT	96.8	96.8	96.8	96.8
RF	96.6	96.8	96.6	96.7

TABLE 11. Average binary class classification results of proposed and competitive algorithms for CICIDS2017.

Algorithm	Accuracy	Precision	Recall	F1-score
Proposed (DT)	99.98	99.98	99.98	99.98
Proposed (RF)	99.98	99.98	99.98	99.98
HAST- I (100)	99.87	99.88	99.85	99.86
HAST- I (300)	99.86	99.87	99.85	99.86
DT	99.95	99.95	99.95	99.95
RF	99.95	99.95	99.95	99.95

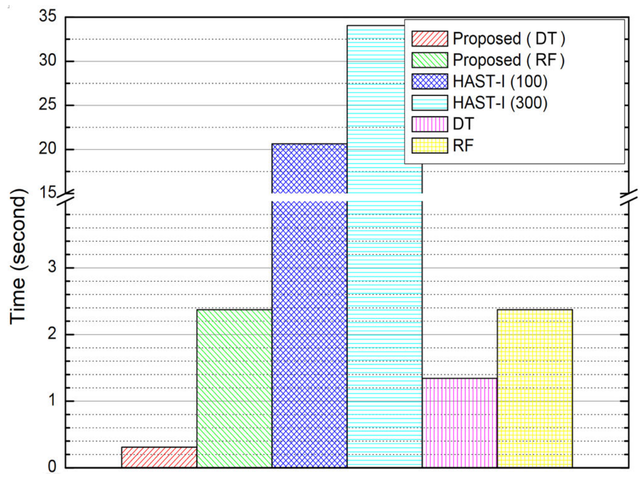

FIGURE 10. Total binary class classification time taken to classify the entire testing dataset for UNSW-NB15.

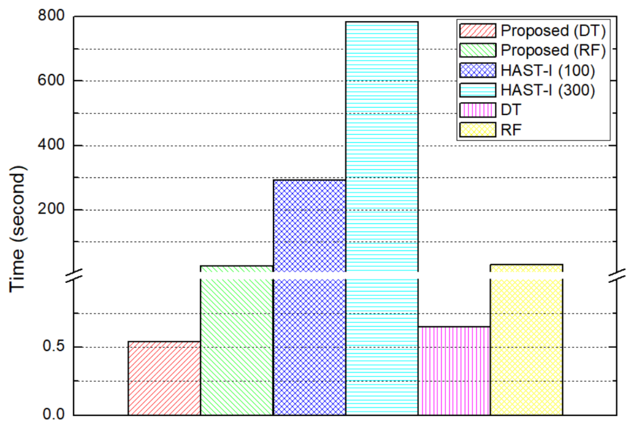

FIGURE 11. Total binary class classification time taken to classify the entire testing dataset for CICIDS2017.

TABLE 12. Number and rate of total flows processed by level 1 and level 2 classifiers for each class data when the proposed algorithm using RF is applied on binary class classification for UNSW-NB15.

Class	Level 1 (%)	Level 2 (%)	Total
Normal	8,582(99.8%)	15(0.2%)	8,597
Attack	1,067(97.4%)	28(2.6%)	1,095
Total	9,649(99.6%)	43(0.4%)	9,692

CPU but HAST-I requires GPU and CPU, the result show that our approach is very promising for real-time detection.

Tables 12 and 13 show that most flows are classified at the level 1 classifier in binary class classification similarly to multi-class classification for UNSW-NB15 and CICIDS2017. Interestingly, the ratio of attacks classified at the level 2 classifier is higher than that of multi-class cases. From this difference, it seems that minor class problem still exists in binary class classification. Such flows tend to have low scores in the level 1 classifier, therefore they are relatively more often classified at the level 2 classifier compared to normal traffic.

TABLE 13. Number and rate of total flows processed by level 1 and level 2 classifiers for each class data when the proposed algorithm using RF is applied on binary class classification for CICIDS2017.

Class	Level 1 (%)	Level 2 (%)	Total
Normal	100,117(45.0%)	122,230(55.0%)	222,347
Attack	130,591(71.9%)	50,966(28.1%)	181,557
Total	230,708(57.1%)	173,196(42.9%)	403,904

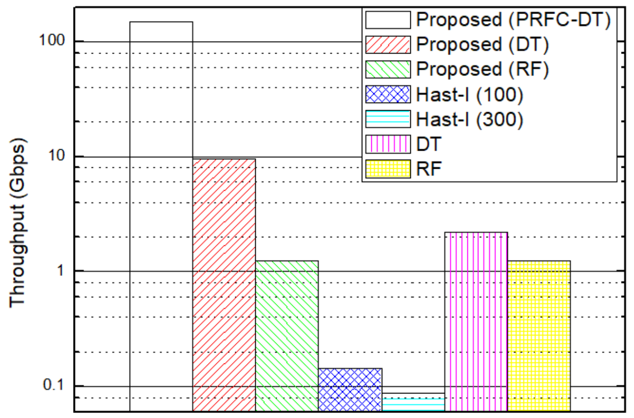

FIGURE 12. Total traffic classification performance on binary class classification for UNSW-NB15.

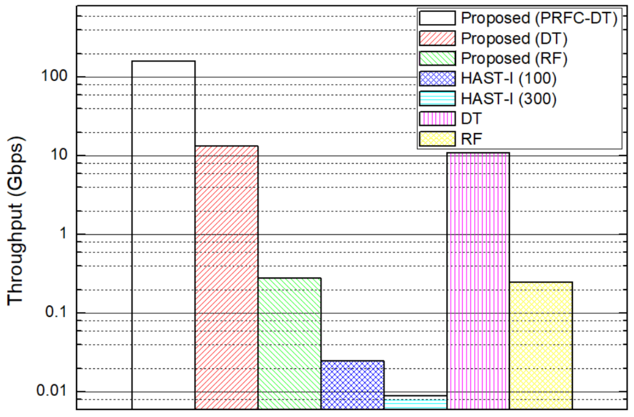

FIGURE 13. Total traffic classification performance on binary class classification for CICIDS2017.

Fig. 12 also shows the packet processing performances of each algorithm for UNSW-NB15. As with multiple classification, the proposed algorithm using DT shows 9.59 Gbps throughput. It shows very high performance compared to competing algorithms, which show at most 2.2 Gbps. Moreover, it can be further improved by replacing DT for the first level with PRFC, with the result that it achieves 149 Gbps throughput. Fig. 13 shows the packet processing throughput for CICIDS2017, which is quite similar to that for UNSW-NB15. From these results, we can conclude that our approach is very helpful to provide fast classification and high accuracy.

The proposed approach reaches the highest detection accuracy in multi class classification and binary class classification, showing enough processing speed to handle intrusion

TABLE 14. Total flow features before feature selection. Features which are generated only when the flow ends are noted in grey.

No.	Description	No.	Description
1	Source port number	24	The content size of the data transferred from the server's http service.
2	Destination port number	25	Source jitter
3	Transaction protocol	26	Destination jitter
4	State	27	Record start time
5	Total duration	28	Record last time
6	Src to dst transaction bytes	29	Source interpacket arrival time
7	Dst to src transaction bytes	30	Destination interpacket arrival time
8	Src to dst time to live	31	The sum of SYN_ACK and ACK_DATA
9	Dst to src time to live	32	Time between SYN and SYN_ACK
10	Src packets retransmitted or dropped	33	Time between SYN_ACK and the ACK
11	Dst packets retransmitted or dropped	34	SIP, SPORT equal to DIP, DPORT
12	Service	35	No. for each state according to specific range of values for source/destination time to live
13	Source bps	36	No. of flows has Get and Post method in http service
14	Destination bps	37	If the ftp session is accessed by user and password
15	Src to dst packet count	38	No of flows that has a command in ftp session.
16	Dst to src packet count	39	No. of connections that contain the same service and source address in 100 connections according to the last time
17	Src TCP window advertisement value	40	No. of connections that contain the same service and destination address in 100 connections according to the last time
18	Dst TCP window advertisement value	41	No. of connections of the same destination address in 100 connections according to the last time
19	Src TCP base SEQ number	42	No. of connections of the same source address in 100 connections according to the last time
20	Dst TCP base SEQ number	43	No. of connections of the same source address and the destination port in 100 connections according to the last time
21	Mean of the flow packet size transmitted by the src	44	No. of connections of the same destination address and the source port in 100 connections according to the last time
22	Mean of the flow packet size transmitted by the dst	45	No. of connections of the same source and the destination address in in 100 connections according to the last time
23	Depth into the connection of http request/response transaction	46	Label

detection in real time supporting enterprise-level and even backbone networks. The proposed algorithm using DT has a lower detection performance of 0.37% and 0.18% on F1-score compared to the proposed algorithm using RF, but it shows better processing performance than existing approaches for UNSW-NB15. Although the proposed algorithm using RF achieves the highest detection accuracy, the algorithm using DT surpasses any other competing algorithms in terms of detection speed. Therefore, from the experimental results, the proposed algorithm using DT or using PRFC and DT is a good choice if the target is a real-time IPS, and the proposed algorithm using RF is also a good choice if detection accuracy is more important.

V. CONCLUSION

As the total traffic volume of networks is rapidly increasing, cyber-attacks are also becoming more sophisticated and transforming into variants. Therefore, real-time IDSs are essential for protecting networks from such attacks. However, real-time detection cannot adopt elaborate and modern techniques due to the processing overhead, exposing weakness to zero-day attacks. We proposed a two-level intrusion detection approach supporting real-time processing with a high detection accuracy. It exploits packet- and flow-based classifiers to compensate for the performance and accuracy. The level 1 classifier extracts some selected features from the packet first to promote the fast classification, achieving real-time attack detection. The level 2 classifier only handles flows that were not classified by the level 1 classifier, therefore the traffic is small enough to be processed by a time-intensive machine-learning-based classifier. Such a unique structure of the two-level classifier can provide classification speed and accuracy simultaneously. We confirmed the effectiveness of this approach by extensive performance evaluation. We expect that it can be an effective solution to build real-time IPSs for overcoming the weaknesses of modern network security.

APPENDIX

See Table 14.

REFERENCES

[1] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," *Appl. Soft Comput.*, vol. 10, no. 1, pp. 1–35, Jan. 2010.

[2] M. Ektefa, S. Memar, F. Sidi, and L. S. Affendey, "Intrusion detection using data mining techniques," in *Proc. Int. Conf. Inf. Retr. Knowl. Manage. (CAMP)*, Mar. 2010, pp. 200–203.

[3] L. Bilge and T. Dumitras, "Before we knew it: An empirical study of zero-day attacks in the real world," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, Oct. 2012, pp. 833–844.

[4] M. Roesch, "Snort—Lightweight intrusion detection for networks," in *Proc. LISA 13th USENIX Conf. Syst.*, vol. 99, no. 1, Nov. 1999, pp. 229–238.

[5] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, Dec. 2018.

[6] C. Kruegel and T. Toth, "Using decision trees to improve signature-based intrusion detection," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*, Sep. 2003, pp. 173–191.

[7] R. Sedgewick, *Algorithms in C: Parts 1-4, Fundamentals, Data Structures, Sorting, and Searching*, 3rd ed. Boston, MA, USA: Addison-Wesley, 1997, p. 702.

[8] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection," *IEEE Access*, vol. 6, pp. 52843–52856, Sep. 2018.

[9] Q. Hu, M. R. Asghar, and N. Brownlee, "Evaluating network intrusion detection systems for high-speed networks," in *Proc. 27th Int. Telecommun. Netw. Appl. Conf. (ITNAC)*, Nov. 2017, pp. 1–6.

[10] M. V. Kotpalliwar and R. Wajgi, "Classification of attacks using support vector machine (SVM) on KDDCUP'99 IDS database," in *Proc. 5th Int. Conf. Commun. Syst. Netw. Technol.*, Apr. 2015, pp. 987–990.

[11] K. A. Jalil, M. H. Kamarudin, and M. N. Masrek, "Comparison of machine learning algorithms performance in detecting network intrusion," in *Proc. Int. Conf. Netw. Inf. Technol.*, Jun. 2010, pp. 221–226.

[12] F. Ertam, L. F. Kilincer, and O. Yaman, "Intrusion detection in computer networks via machine learning algorithms," in *Proc. Int. Artif. Intell. Data Process. Symp. (IDAP)*, Sep. 2017, pp. 1–4.

[13] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, Oct. 2016, pp. 1–6.

[14] H. Chauhan, V. Kumar, S. Pundir, and E. S. Pilli, "A comparative study of classification techniques for intrusion detection," in *Proc. Int. Symp. Comput. Bus. Intell.*, Aug. 2013, pp. 40–43.

[15] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Trans. Syst., Man, Cybern., C, Appl. Rev.*, vol. 38, no. 5, pp. 649–659, Sep. 2008.

[16] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[17] N. Wattanapongsakorn, S. Srakaew, E. Wonghirunsombat, C. Sribavonmongkol, T. Junhom, P. Jongsubsook, and C. Charnsripinyo, "A practical network-based intrusion detection and prevention system," in *Proc. IEEE 11th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Jun. 2012, pp. 209–214.

[18] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, L. Wang, G. Wang, and J. Cai, "Recent advances in convolutional neural networks," Dec. 2015, *arXiv:1512.0710*. [Online]. Available: https://arxiv.org/abs/1512.0710

[19] *The UNSW-NB15 Dataset Description*. (Nov. 14, 2018). [Online]. Available: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

[20] P. Gupta and N. McKeown, "Algorithms for packet classification," *IEEE Network*, vol. 15, no. 2, pp. 24–32, Mar. 2001.

[21] S. Singh, F. Baboescu, G. Varghese, and J. Wang, "Packet classification using multidimensional cutting," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun. (SIGCOMM)*, Aug. 2003, pp. 213–224.

[22] D. E. Taylor, "Survey and taxonomy of packet classification techniques," *ACM Comput. Surveys*, vol. 37, no. 3, pp. 238–275, Sep. 2005.

[23] W. Pak and Y.-J. Choi, "High performance and high scalable packet classification algorithm for network security systems," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 1, pp. 37–49, Feb. 2017.

[24] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Rahim, "Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection," *IEEE Access*, vol. 6, pp. 33789–33795, May 2018.

[25] S. Sahu and B. M. Mehtre, "Network intrusion detection system using J48 decision tree," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Aug. 2015, pp. 2023–2026.

[26] Y. Cheong, K. Park, H. Kim, J. Kim, and S. Hyun, "Machine learning based intrusion detection systems for class imbalanced datasets," *J. Korea Inst. Inf. Secur. Cryptol.*, vol. 27, no. 6, pp. 1385–1395, Dec. 2017.

[27] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, Nov. 2017.

[28] K. Park, Y. Song, and Y.-G. Cheong, "Classification of attack types for intrusion detection systems using a machine learning algorithm," in *Proc. IEEE 4th Int. Conf. Big Data Comput. Service Appl. (BigDataService)*, Mar. 2018, pp. 282–286.

[29] W.-H. Lin, H.-C. Lin, P. Wang, B.-H. Wu, and J.-Y. Tsai, "Using convolutional neural networks to network intrusion detection for cyber threats," in *Proc. IEEE Int. Conf. Appl. Syst. Invention (ICASI)*, Apr. 2018, pp. 1107–1110.

[30] S. Soheily-Khah, P.-F. Marteau, and N. Bechet, "Intrusion detection in network systems through hybrid supervised and unsupervised machine learning process: A case study on the ISCX dataset," in *Proc. 1st Int. Conf. Data Intell. Secur. (ICDIS)*, Apr. 2018, pp. 219–226.

[31] Y. Yuan, L. Huo, and D. Hogrefe, "Two layers multi-class detection method for network intrusion detection system," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2017, pp. 767–772.

[32] T.-S. Lim, W.-Y. Loh, and Y.-S. Shih, "A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms," *Mach. Learn.*, vol. 40, no. 3, pp. 203–228, 2000, doi: 10.1023/A:1007608224229.

WOOSEOK SEO was born in Gumi, South Korea, in 1994. He received the B.S. and M.S. degrees in computer engineering from the Keimyung University, in 2017 and 2019, respectively. He is currently working as a Software Engineer at Wookyoung Information Technology, Daegu. His research interests include biometric authentication and network intrusion/prevention systems.

WOOGUIL PAK received B.S. and M.S. degrees in electrical engineering and the Ph.D. degree in electrical engineering and computer science from Seoul National University, in 1999, 2001, and 2009, respectively. In 2010, he joined the Jangwee Research Institute for National Defence, as a Research Professor, and the Keimyung University, Daegu, South Korea, in 2013. Since 2019, he has been an Associate Professor with Yeungnam University, Kyeongsan, South Korea. His current research interests include network and system security, blockchain, and real-time network intrusion prevention based on machine learning for over 1Tbps networks.