IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 7, 2021, accepted May 26, 2021, date of publication June 7, 2021, date of current version June 16, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3087201

Hybrid Classification for High-Speed and High-Accuracy Network Intrusion Detection System

TAEHOON KIM AND WOOGUIL PAK
Department of Information and Communication Engineering, Yeungnam 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, 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 Cybercrime is growing at a rapid pace, and its techniques are becoming more sophisticated. In order to actively cope with such threats, new approaches based on machine learning and requiring less administrator intervention have been proposed, but there are still many technical difficulties in detecting security attacks in real time. To solve this problem, we propose a new machine learning-based real-time intrusion detection algorithm. Unlike the existing approaches, the one proposed can detect the presence of an attack every time a packet is received, enabling real-time detection. In addition, our algorithm effectively reduces the system load, which may significantly increase from real-time detection, compared to non-real-time detection. In the algorithm, the increase in the number of memory accesses can be minimized (to below 30 %) compared to conventional methods. Since the proposed method is pure software-based approach, it has excellent scalability and flexibility against various attacks. Therefore, the proposed method cannot support the high classification performance of the hardware-based method but also the high flexibility of the software-based method simultaneously, it can effectively detect and prevent various cyber-attacks.

INDEX TERMS Hybrid classifier, network attack, network intrusion detection, three level, real-time detection.

I. INTRODUCTION

Network speed is increasing day by day, and at the same time, the rate of cybercrimes is increasing rapidly [1]. To solve this problem, a lot of security equipment is used, such as a firewall that provides simple session control using a five tuple-based policy, an intrusion detection system (IDS) that provides non-real-time detection against various attacks, and an intrusion prevention system (IPS) that processes sessions to detect attacks in real time. In particular, variant network attacks and zero-day attacks are constantly increasing, and they make it impossible to safely manage networks with security equipment that constantly requires administrator intervention [2], [3]. Of course, in order to solve these problems, network security equipment has evolved from using fragmentary and simple information about packets or sessions to more sophisticated and advanced methods using complex information, like the overall behavioral characteristics of the traffic. Today, machine learning-based intrusion detection technology is a promising solution to the problems. When machine learning, which has recently received much attention, is used to detect network attacks, it not only greatly reduces the need for intervention by administrators, but it also effectively responds to variant attacks or zero-day attacks by automating the task of extracting and utilizing the behavior patterns of the attacks [4]–[8].

However, most machine learning-based security systems have difficulty supporting real-time packet processing because of the characteristics of machine learning algorithms and the characteristics of network data. In the case of 10 Gbps networks, the network security equipment must process a maximum of 14 million packets, or three million packets on average every second under strict delay conditions. Therefore, it is impossible to perform packet-by-packet processing in real time while satisfying delay conditions with existing slow machine learning algorithms. Because of this limitation,

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

today's machine learning-based security systems only monitor statistical characteristic values for each session, instead of detecting per-packet attacks, and they just determine whether a network attack exists after the session ends [3]–[8].

Eventually, due to these limitations, machine learning-based network security devices are mainly used for intrusion detection, which determines the existence of an attack, rather than for real-time attack detection. In this case, since such devices notice that an attack has occurred after the attack is completed, it is helpful in recovering from damage after the attack, but fundamentally, it is impossible to safely defend against, and protect the network from, such cyber-attacks.

Through a long study of such technical limitations and the urgency to defend against security threats, this study proposes a new approach to solve them. Similar to the existing machine learning algorithms, the proposed algorithm detects an attack by session level. However, unlike the conventional methods of classification at the end of each session, classification is performed every time a packet is received. Nevertheless, it shows almost the same system load, compared to the existing session-based approaches. Due to this unique characteristic, unlike the existing methods, the proposed algorithm can detect attacks in real time. Also, it has a fairly high attack-detection rate without performance degradation from real-time attack detection. In order to achieve this result, it has a special structure for optimized classification of each attack class, and a hybrid classifier composed of three classifiers is applied to minimize misclassifications.

The most important advantage of our approach is that it can improve the performance of machine learning-based intrusion detection systems in close to real time. Through this, it maintains the advantages of machine learning and, at the same time, processes large amounts of traffic without delay. Therefore, it is possible to safely protect networks and users from the cybercrime that is now increasing rapidly.

Also, the proposed approach is implemented as a software-only algorithm. In order to detect or prevent intrusions, it must be able to efficiently process a large amount of traffic. In the existing approaches, dedicated hardware assists in speeding up slow machine learning. These hardware-based solutions offer very high performance for specific operations. However, the need for additional hardware increases the cost of the equipment, compared to the software-based approach, and the supported hardware functions are limited, so it is difficult to provide new complex functions whenever new requirements or new attacks arise. As cyber-attack methods become more diverse (and new methods are emerging day by day), it is very difficult to provide flexibility and scalability to prevent the high dynamics of such attacks by using a hardware-based approach.

Therefore, the proposed approach, which operates as a software algorithm, is of great help in this regard. Although it is software-based, it can provide very high performance, compared to the existing session-based approaches, and high flexibility and high scalability provided only by the software-based method. Such merits are the most important characteristics needed to increase network security. Now, we will explain the proposed method in detail and demonstrate the improved performance through evaluation.

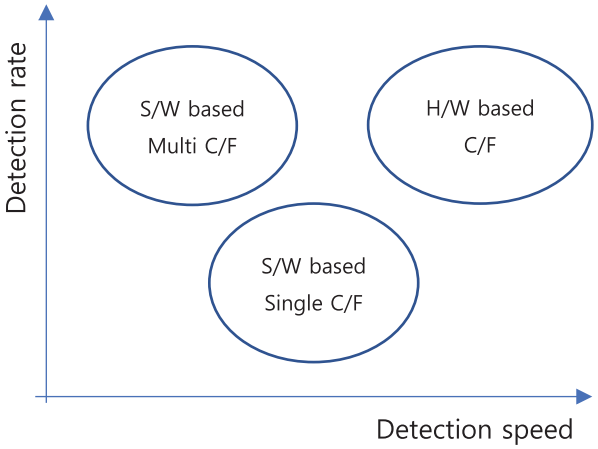


FIGURE 1. Classification of network intrusion–detection approaches.

The structure of the rest of the paper is as follows. In Section 2, we examine existing research in detail. In Section 3, the structure and characteristics of the proposed method are explained. In Section 4, we analyze how advantageous the proposed method is for real-time detection through a performance analysis. We conclude in Section 5.

II. EXISTING WORK

Network intrusion-detection technology based on machine learning can be largely divided into the software-based approach and the hardware-based approach. In addition, the software-based approach can be further divided into single-algorithm approaches and multi-algorithm approaches, depending on the number of algorithms used internally. For each approach, a comparison according to classification accuracy and speed is shown in Fig. 1.

A. SOFTWARE-BASED NIDS USING A SINGLE CLASSIFIER

NIDSs using software-based, single classification manage and monitor all packets received for each network session [7], [9]–[13]. After each session ends, features for a machine learning model are created using the monitored data from the session, which are then learned and classified using a single classification algorithm. A single classifier was used in early machine learning-based NIDSs, and a number of studies are still being conducted. The biggest advantage of this approach is that it does not detect an attack at the packet level, but at the session level, which is a logical grouping, so it can greatly reduce the overhead from processing traffic. In particular, the single classifier is of great help in improving normal traffic performance. The session length with normal traffic is much longer than that of most attack traffic. Therefore, when session-level processing is applied to normal traffic instead of packet-level, classification does not occur frequently, and thus, system overhead significantly decreases. In addition,

the single classifier does not use data for each packet, and creates a fixed number of features, regardless of the number of packets in the session, so memory usage is also very low. In particular, since a small number of features (generally, less than 100) is processed by a single algorithm, the learning and classification speed is very fast, so it can even be applied to large networks. However, it is difficult to provide high detection accuracy for various attack classes with a single algorithm. In addition, since a feature is created after the session is ended, if an attack is detected, the probability that the attack has already ended is high.

The inability to perform real-time intrusion detection is definitely a big drawback, because classification by packet is impossible due to the low classification performance of the software-based approach. However, the software-based approach runs on conventional hardware platforms, so it can easily increase memory capacity, making it easy to utilize large models and large datasets without memory constraints. It also has the advantage of being able to implement and install various types of machine learning algorithms according to the changing requirements.

B. SOFTWARE-BASED NIDS USING A MULTI-CLASSIFIER

This approach performs learning and classification using the features from the session, but uses several classification algorithms at the same time. It mainly uses an ensemble algorithm or a multi-level classifier [3], [14], [15]. The ensemble algorithm applies several classifiers at the same time, combines the results, and comprehensively derives a final classification result. Detection accuracy can be improved by using several algorithms for various classes. On the other hand, a multi-level classifier runs the first-level classifier, and then chooses one for the next level and executes it based on the result. The most common combinations are one unsupervised learning algorithm and a supervised one, or two supervised learning algorithms. For example, one study performed partitioning using k-Nearest Neighbors (kNN) and applied a Decision Tree (DT) to each partition [16]. The main reason for using various classification algorithms is to accurately detect various attack classes. This is because it is very difficult to accurately classify multiple-attack traffic with very different characteristics by using only a single classifier.

The multi-classifier approach applied to session-level classification achieves high detection accuracy, but traffic-processing speed deteriorates to a very low level due to the slow machine learning speeds, which become worse with multiple classifiers. In particular, as deep learning algorithms have been widely applied in recent years, their performance has become even slower. As with the approach using a single classifier, instead of creating a feature by using a packet, it creates a feature using the monitored data from the session, so it is impossible to defend against an attack, and the approach aims only at detecting an attack, like the software-based approach using a single classifier. However, due to the slow speed, it is difficult to detect an attack immediately, even after a session has just terminated, in a network that has a large amount of traffic, and there may be a long delay in detecting the attack.

C. HARDWARE-BASED NIDS

A hardware-based NIDS uses a network interface card (NIC) equipped with dedicated hardware for traffic processing [17]–[19]. This approach significantly relieves performance degradation due to overhead from data transmissions between the existing NIC and the CPU, and from the low classification speed of machine learning [20].

Dedicated hardware, generally referred to as a smart NIC, goes beyond simply receiving packets and transmitting them to the CPU, and performs even basic NIDS functions in the NIC itself. For instance, Marvell's LiquidIO III provides functions like simple pattern matching, encryption/decryption, and deep packet inspection (DPI) by a multicore processor and several coprocessors installed on the NIC [18]. In addition, it can support simple matching of an access control list (ACL) right up to complex attack detection using machine learning models. Recently, smart NICs equipped with hardware specialized for machine learning have also been released. For example, Xilinx's Alveo U250 is equipped with two 100GbE and provides a very high performance through FPGA, achieving more than three times the machine-learning inference performance, compared to the existing CPU/GPU-based software approach [19]. By detecting attacks on the NIC, the load on the CPU can be greatly reduced, and only traffic that is difficult to handle on the NIC is handled by the CPU to perform more sophisticated and intelligent processing, providing very high classification accuracy and very fast performance at the same time. Therefore, it is possible to support not only session-based classification but also packet-based classification, which is a huge advantage in that an attack can be detected in real time, which cannot be achieved by pure software-based approaches.

On the other hand, in the hardware-based approach, resources cannot be expanded as required. For example, it cannot support machine learning models with a size larger than the installed memory capacity. Generally, resources are fixed and not upgradable. In addition, it is limited in the implementation of various machine learning algorithms, because the supported hardware functions may not be varied enough. Also, to support many Ethernet ports, multiple smart NICs must be used. In this case, cards must be co-operated and synchronized, and as a result, are disallowed from operating as independent systems. In other words, data synchronization occurs between NICs or between each NIC and the CPU, which may cause implementation complexity and performance degradation. Also, if malicious users exploit such low flexibility and poor scalability, it becomes a critical disadvantage that allows them to easily find a weakness to bypass detection or neutralize the security equipment.

In the end, there is a desperate need for a new approach that is able to combine the flexibility and scalability of a software-based NIDS and the speed of a hardware-based

NIDS. However, as far as we know, no existing studies have achieved this goal.

III. PROPOSED ALGORITHM

In this section, we present a new structure and method for real-time attack detection in an NIDS using a software-based multi-classifier. The proposed algorithm applies the existing session-based classification for exact detection, and additionally uses packet-based classification for fast detection. When packet-based classification is applied, it is difficult for the NIDS to execute packet processing without a delay due to the low classification speed of the software-based classifier. In order to solve this problem, the proposed approach adopts a hybrid classifier based on a threshold for each class, and this allows the NIDS to detect attacks in real time. In general, the proposed algorithm classifies all packets received by a packet classifier, and if the score for each class is higher than a specific threshold, the packet and the session which the packet belongs to are classified and processed as the classification result. On the other hand, if the score is not higher than the threshold for all packets in the session, then session-based hybrid classification is executed. Session-based hybrid classifiers are very rarely used because most of the traffic is classified by a packet classifier. The advantage of this structure is that the packet classifier significantly reduces the computational overhead the session classifier has to bear by initially processing sessions that can accurately detect a class in advance.

When the session cannot be accurately classified until it ends, a hybrid classifier applying a session-based classifier and a packet/session-based classifier at the same time is adopted to achieve more accurate detection. This classifier primarily performs the same operation as the existing session-based classifiers. When the session ends, monitored data for the session are created, features for the session are created, and the session is classified using them. However, what differentiates the hybrid classifier from the existing session-based classifiers is how it handles sessions that cannot be accurately classified, even by the session-based classifier. Existing classifiers cannot handle such sessions, so they simply select the class with the highest score. On the other hand, in the proposed algorithm, session-based classification is performed primarily, and if it does not exceed a specific threshold for each class, the result of the packet-based classifier and the result of the session-based classifier are combined as new features for more accurate classification. Then, classification is performed through a secondary packet/session classifier. Now, the operation of each step in the classifier will be described in detail.

A. PACKET-BASED EARLY DETECTORS (PED)

The PED is a first-stage classifier to perform fast attack detection, and it classifies every packet when it is received in order to detect attack traffic. When classifying a packet, features for classification are created through the accumulated session information for the corresponding session the received packet belongs to. For example, when the first packet is received, session information is extracted from the packet, and saved into a session table, Then, features are created by processing the session information. When the second packet is received, the session information is updated by adding the second packet's information and saved into a session table. After that, features are created from the updated session information. When the k-th packet is received, session information is updated in the same way by adding the k-th packet's information to the accumulated session information from the first to the (k-1)-th packet. The k-th features are created from the updated session information. An example of creating features is seen in Fig. 2a, which shows session information that is updated every time a packet is received. This information is used to create features for packet classification. Fig. 2b shows the calculation of some selected features using session information.

Packet_count += 1
Flow_duration = Current_time - Flow_start_time
Sum_of_IAT += Current_time - Last_time_stamp
Sum_of_IAT_sqr += (Current_time - Last_time_stamp)^2
Last_time_stamp = Current_time
…

(a) Session information update for each received packet

Flow_IAT_mean = Sum_of_IAT/Packet_count
Flow_IAT_stddev=$\sqrt{\text{Sum\_of\_IAT\_sqrt/Packet\_count- Flow\_IAT\_mean}^2}$
…

(b) Feature generation from session information

FIGURE 2. Generation of accumulated packet-based features, where IAT stands for inter-arrival time.

Since not all the features used in the existing NIDS literature can generate the above cumulative features, in this study, only features that can be generated using these cumulative features are selected and used. Fortunately, it is possible to create a large number of features that have been used in existing studies on the suggested method. For example, most of the features used in the CICIDS2017 dataset can be created based on the above accumulated packet-based features [21].

In order to determine by using the first-stage classification result whether the session is under attack, that first-stage result must have remarkably high reliability. Basically, in machine learning-based classification, reliability is expressed as a score, so it is reasonable to believe the result (and process the session according to it) only when it is above a certain score. The question is how to determine the minimum score in order to trust the result of first-stage classification. Basically, there are attacks of various types and characteristics, so using only one minimum score for all attacks can be dangerous. After all, it is necessary to set the optimal minimum score for each class. However, since the minimum score for one class may affect the classification results of other classes, the score must be decided carefully

by considering all classes. When analyzing the results of first-stage classification for packets belonging to the same session, we can find many cases where a false detection is obtained at the beginning, and accurate detection is obtained after a few more received packets. In this case, if the minimum score for the class is not set high enough, the session is processed as the wrong class. In the end, when determining the minimum score for each class, the results of other classes must be considered, so determining the optimal minimum score for each class becomes an exceedingly difficult problem.

Instead of finding an optimal solution, we build an approximated solution in the proposed approach. It applies the gradient ascent method to the randomly determined initial value $(\theta_1^0, \theta_2^0, \cdots, \theta_f^0)$ of each class threshold, which repeats adjustments (one step at a time) in the direction of one dimension, where the value of the F1-score increases the most, and where θ_n^m is the threshold value of the n-th class in the m-th step.

That is, let $S(\theta_1^k, \theta_2^k, \cdots, \theta_f^k)$ be the total F1-score value obtained through classification using threshold values at the k-th step. Now when we change the threshold for the h-th score by $\Delta\theta$, the new total F1-score becomes $S_h = S(\theta_1^k, \theta_2^k, \cdots, \theta_h^k + \Delta\theta, \cdots, \theta_f^k)$. At this time, if $h^* = \text{argmax}_h S_h$, the new threshold for each class is $(\theta_1^{k+1}, \theta_2^{k+1}, \cdots, \theta_f^{k+1}) \leftarrow (\theta_1^k, \theta_2^k, \cdots, \theta_{h^*}^k + \Delta\theta, \cdots, \theta_f^k)$. Here, the reason for changing $\Delta\theta$ for only one class at a time is to prevent a large increase in the computational amount when the number of classes is large. Generally, we should consider 2^f directions at each step because the S function cannot be differentiated. Thus, it impractically takes too long time for each step.

This process ends when we can no longer adjust $(\theta_1^k, \theta_2^k, \cdots, \theta_f^k)$ to increase the F1-score. Although this method cannot find the global maximum value, it can effectively reduce the amount of calculations; so by repeating this whole procedure many times, we can obtain an approximate value that is quite close to the correct answer.

B. HYBRID LAZY DETECTOR (HLD)

The HLD is largely composed of a session-based classifier, called HLD-S, and a packet/session-based classifier, called HLD-PS. The HLD classifies sessions that do not obtain a classification score above the minimum score in the PED. In more detail, first, when the session ends, the HLD creates features for the entire session, and classifies it using those features. Similar to the PED, the HLD has a threshold for classification scores of each class. Therefore, if the classification score obtained by HLD-S is greater than the threshold value for the class, the session is processed based on the classification result. On the other hand, when it is less than the threshold of the class, we do not trust the result of the HLD-S, and we perform classification again in the HLD-PS. The HLD-PS is a classifier for making comprehensive decisions using the results of both the PED and the HLD-S. If the final score from the HLD-S does not exceed the corresponding threshold, it means that the results of both the PED and the HLD-S are not sufficiently reliable. Therefore, it is desirable to consider the results of both the PED and the HLD-S to solve this problem. Hence, the result is classified in the HLD-PS by using the score of each class, which is the result of the two classifiers, as a feature, so the HLD-PS has $2f$ features, where f is the total number of classes. The HLD-PS has three labels (0, 1, and 2), where 0 means that the session should be processed according to the PED classification result; 1 means the session should be processed according to the HLD-S classification result; and 2 means the results of the PED and the HLD-S are both not correct. Since intervention by the administrator is required for label 3, logging is performed or the session is processed according to a predetermined policy.

C. CLASSIFIER LEARNING

In order to train each classifier, it is necessary to know the threshold for each class in each classifier. However, a trained classifier is needed to determine the optimal threshold for each class. In the end, the problems of learning a model and finding a threshold value are correlated, so solving them at the same time requires too many computations. As a resolution, this study proposes the following method. First, the PED and the HLD-S are trained using the entire training dataset. In addition, threshold values for each class are set so that the PED and the HLD-S have the highest detection rates. In this study, F1-score is assumed to be used to measure the detection rate, for easy explanation. However, this can be changed according to the optimization purpose of each system. In order to easily calculate the performance of the PED and the HLD-S based on the threshold value, the classification for each class is calculated in advance according to threshold values that are increased at a certain step size. Then, by using this result, it is possible to determine the optimal threshold for each class by applying the gradient ascent method without a large amount of calculation.

After the threshold for each class of the PED is determined, the threshold for each class of the HLD-S is calculated in the same way based on pre-calculating the classification result and the gradient ascent method. At this time, the HLD-PS is trained using only unclassified data in the PED from among the entire training dataset. Therefore, the size of the dataset used to train the HLD-PS becomes very small, compared to the entire training dataset. That results in more accurate training of the HLD-PS by excluding unnecessary data. The method and steps for training are in Algorithm 1.

D. THE OVERALL STRUCTURE OF THE ENTIRE CLASSIFIER

In order to construct an entire classifier, we need to specify classification algorithms for the PED and the HLD. In order to select the classifier of each level, the following method is used in this study. First, since speed is more important than accuracy in the PED, we choose between DT and random forest (RF) for the PED [22]. Also, since accuracy is more important than speed for HLD classifiers, it is necessary to

Algorithm 1: Train Classifier

Input: PED, HLD, dataset1 (for training)
Output: Trained PED and HLD, and two threshold vectors $\theta^{(1)}$ and $\theta^{(2)}$ for PED and HLD-S
Function *Find_Threshold(CF1, CF2, Dataset)*:
  Train CF1 and CF2 with Dataset.
  Randomly choose $\theta_0 = \left(\theta_1^0, \theta_2^0, \cdots, \theta_f^0\right)$ of CF1.
  $k \leftarrow 0$
  while *true* **do**
    $\theta_{k+1} \leftarrow \left(\theta_1^k, \theta_2^k, \cdots, \theta_{h^*}^k + \Delta\theta, \cdots, \theta_f^k\right)$ using gradient ascent method
    if $S(\theta_k) > S(\theta_{k+1})$ *for CF1 & CF2.* **then**
      break
    $++k$
  return θ_k

$\theta^{(1)}$ = Find_Threshold(PED, HLD-S, dataset1)
Make a sub-dataset, i.e., dataset2, which is composed of data classified by HLD using PED with $\theta^{(1)}$.
$\theta^{(2)}$ = Find_Threshold(HLD-S, HLD-PS, dataset2)
return PED, HLD, $\theta^{(1)}$, and $\theta^{(2)}$

individually select between RF and ADT to find the most accurate algorithm combination. The reason for selecting a classifier from among the DT, RF, and ADT is ease of implementation, because all three classifiers are implemented based on the DT. Other algorithms can be also selected, depending on the characteristics of the application field or the characteristics of the dataset.

Fig. 3 shows the overall procedure of the proposed algorithm. The process of the proposed classifier is largely divided into packet processing using the PED each time a packet is received, and session processing using the HLD whenever a session ends. As shown in Fig. 3, the PED searches for the session to which the received packet belongs in the session table that stores session information. It then adds information on the current packet, and creates a feature vector using the updated session information to perform classification. Therefore, the PED must be able to access the session table at high speed. In general, sessions are determined by five tuples, so a high-speed search can be supported using an open hash data structure. In addition, the results with the highest scores from among the classifications by the PED are added together in the session information. This is later used as a feature in the HLD-PS. As shown in Fig. 3, the HLD runs every time a session ends, and performs classification using the features for the terminated session and the PED's classification results, if necessary.

IV. PERFORMANCE EVALUATION

In order to compare and analyze the performance of the proposed approach, the latest competing algorithms and several

TABLE 1. Datasets for performance evaluation.

Dataset name	Characteristic	Value
ISCXIDS2012	Dataset size	160K
	Total number of classes	5
	Ratio of training to testing datasets	7:3
CICIDS2017	Dataset size	1M
	Total number of classes	11
	Ratio of training to testing datasets	7:3

TABLE 2. Combination of classification algorithms used in the proposed method.

Dataset	Machine-learning algorithms (for PED, HLD-S, and HLD-PS)
ISCXIDS2012	RF, RF, ADT
CICIDS2017	RF, ADT, ADT

datasets were used to compare classification speed, system load, and classification accuracy for each class. The proposed algorithm internally uses three classification algorithms that should be selected according to the characteristics of each dataset. The characteristics of the dataset used in the experiment are shown in Table 1, and the chosen combinations of the classification algorithms (based on the highest F1-score selected through pre-experiments) are shown in Table 2.

The algorithms selected for comparison are RF, Adaboosted DT [23] (ADT), SMOTE+RF [24], DTNB [25], TSE [26] including Rotational Forest, Extreme Learning Machine (ELM), Deep Neural Network (DNN), Gradient Boosting Tree (GBT), and Support Vector Machine (SVM). Various characteristics can be compared, selecting from simple algorithms to the latest sophisticated algorithms. In addition, we also present comparison results with an algorithm similar to the proposed approach, but using a global threshold rather than a threshold for each class. Since all of the existing algorithms are for session-based classification, comparing them with a global threshold-based algorithm is helpful in analyzing how much the proposed approach improves performance, compared to the simple integration of packet and session classifiers. The detailed configuration for each algorithm is shown in Table 3.

The most important factor in performance evaluation is the speed in detecting attacks. For real-time detection, the speed must be significantly faster than the existing method. In addition, system load is also a very important factor. If the system load is too high, compared to the existing method, expensive hardware is required, or implementation may even be impossible, so it is a serious disadvantage. Lastly, detection accuracy is the most basic performance for an IDS. Although the detection speed is fast, network security equipment with low detection accuracy is impractical. Therefore, in this section, performance evaluation results and analysis are presented

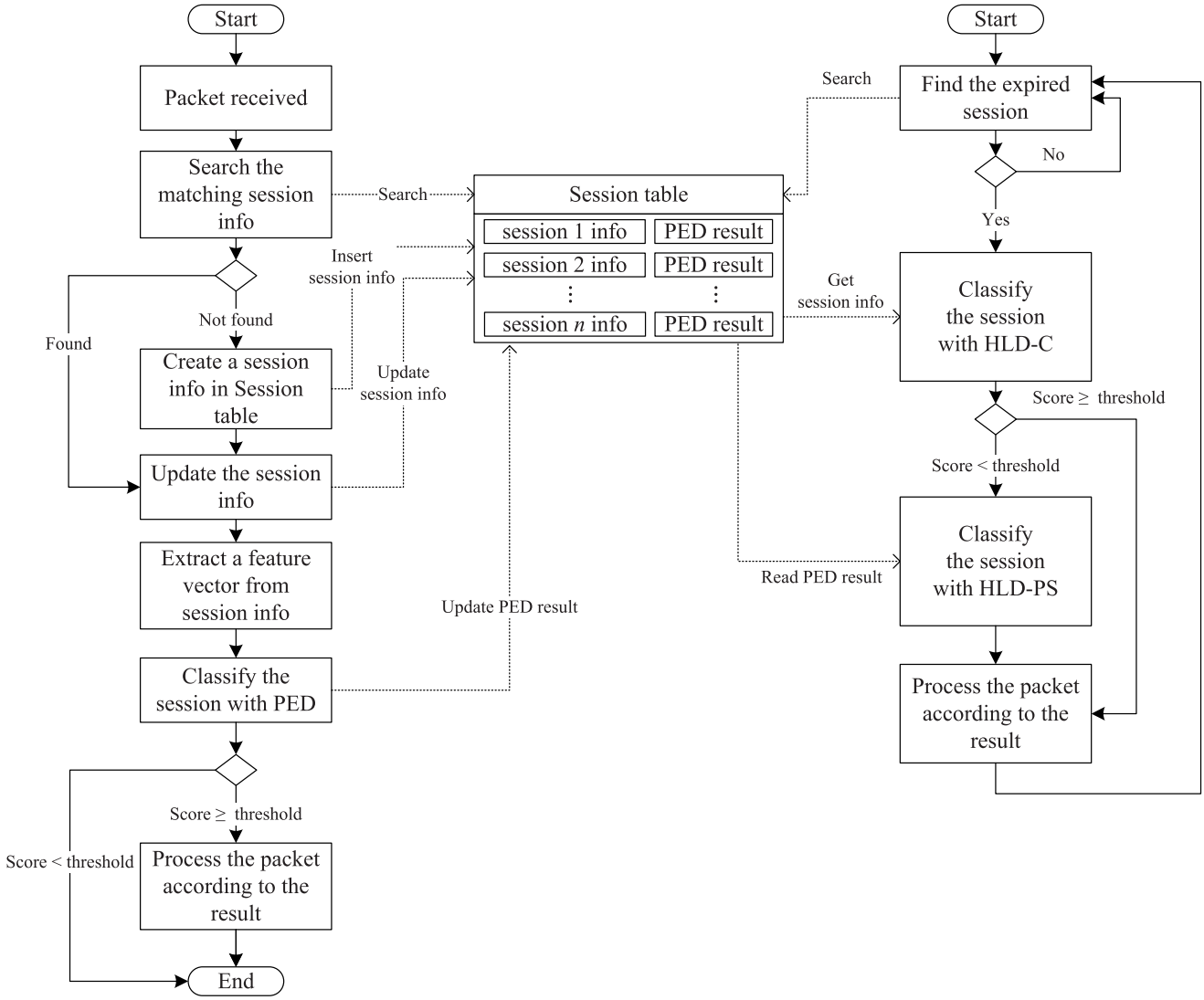


FIGURE 3. Session table and procedure for the entire classification process.

in the order of detection speed, system load, and detection accuracy.

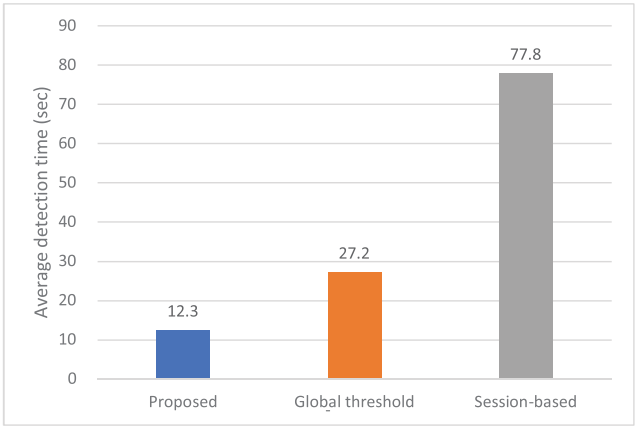
A. COMPARISON OF DETECTION TIMES

Existing session-based algorithms simply classify a session after the session ends, so attack detection is delayed until the session terminates. Therefore, in the session-based method, assume that the session end time is the same as the classification time. Fig. 4 shows the results of comparing classification times of the proposed algorithm, the global threshold-based classification algorithm, and an existing session-based classification algorithm for the two datasets (CICIDS2017 and ISCXIDS2012) [14]. As seen in the results, the proposed algorithm is six times faster than the existing session-based algorithm for the two datasets. In particular, it is more than twice as fast as the global threshold-based algorithm. This indicates that the threshold value optimized for each class accelerates detection speed for each class. It confirms that if the proposed algorithm is applied, attacks can be detected effectively and quickly, regardless of the dataset.

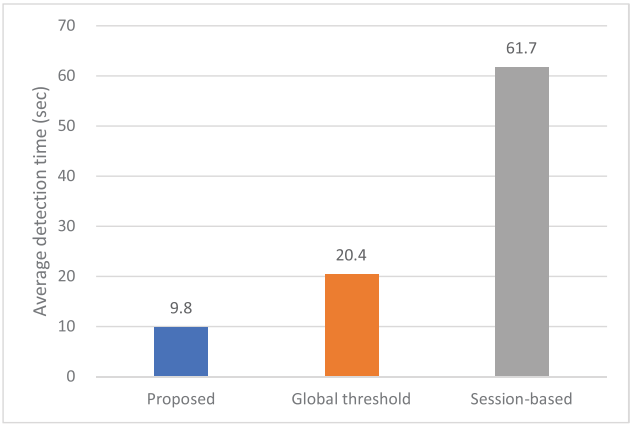
Fig. 5 shows the average time required to detect each class with an existing session-based approach, the proposed approach, and the global threshold-based approach for the CICIDS2017 and ISCXIDS2012 datasets, respectively. In Fig. 5a, we can see that the proposed approach can detect Benign, DDoS, Dos Hunk, and Dos Slowhttptest classes more than 20 times faster than the existing approaches. For the FTP-Patator class, the detection speed improved significantly, by more than 300 times. On the other hand, for classes such as Bot and PortScan, the speed improvement is very small, and the variation in performance improvement by class is very large. However, as explained in detail later, even the Bot class had a huge improvement in terms of system load. This is because the PortScan class has a very short

TABLE 3. Parameter settings for each algorithm.

Algorithm	Parameter	Value
Proposed	$\Delta\theta$	0.1%
Global threshold-based	$\Delta\theta$	0.1%
DTNB	max depth	Unlimited.
	min samples to split	2
	min samples leaf	1
	max features	8
	max leaf nodes	Unlimited
SMOTE+RF	k neighbors	5
TSE (Rotation Forest)	# of trees	10
	rotation algo	PCA
	bootstrap	True
	max depth	Unlimited
	min samples split	2
	min samples leaf	1
	max features	80
	max leaf nodes	Unlimited
RF	# of trees	100
	max depth	Unlimited
	min samples split	2
	min samples leaf	1
	max features	8
	max leaf nodes	Unlimited
ELM	unit	1024, 768, 512, 256, 128
	activation	ReLU
	density	0.5
DNN	unit	1024, 768, 512, 256, 128
	activation	ReLU
	density	0.5
ADT	# of trees	50
	learning rate	1
	algorithm	SAMME
GBT	loss	deviance
	learning rate	0.1
	# of estimators	100
	criterion	Friedman MSE
	min sample split	2
	min sample leaf	1
	max depth	3
	tolerance	1×10^{-4}
SVM	C	1
	kernel	RBF
	degree	3
	gamma	scale
	cache size	200
	decision function shape	OVR
	coefficient0	0
	shrinking	True



(a) CICIDS2017 dataset



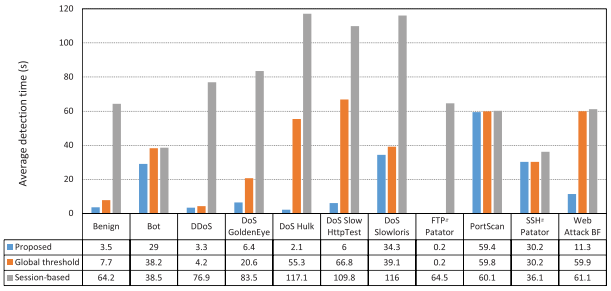
(b) ISCXIDS2012 dataset

FIGURE 4. Time required for detection when using the CICIDS2017 and ISCXIDSIDS2012 datasets.

session length, i.e., 2 on average, so there is little room for improvement.

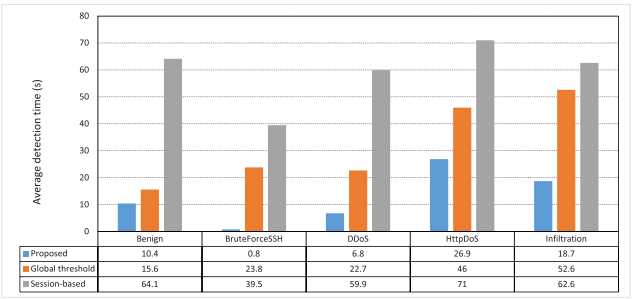
Fig. 5b shows the results from measuring the average detection time for each class in the ISCXIDS2012 dataset. For each class, the range of performance improvement stayed very high, compared to the session-based approach, ranging from 50 times to 2.6 times. Only Benign and DDoS are classes common to both datasets compared to CICIDS2017. However, the performance improvement with those two classes in ISCXIDS2012 was more than double. Therefore, it shows that the effect of the characteristics of the dataset on performance is very great. Nevertheless, we found that the proposed approach improved significantly detection speed for all classes in both datasets, compared to an existing session-based IDS. In addition, we can see that detection performance improved for all classes, even compared to the approach using the global threshold. In conclusion, the proposed approach is very effective in improving detection speed.

Fig. 6 shows the number of packets to be received for detection, instead of the time taken in detection as shown in Fig. 4. In the proposed approach or the global threshold-based approach, the number of packets received until the classification was completed for a session indicates the number of classifications for the session. Since the classification operation consumes a lot of processing power and time, reducing the number of classifications per session will directly improve system performance. Therefore, how much the proposed approach reduces the number of classifications is very important. As shown in Fig. 6, the proposed approach requires 7 to 13 times fewer packets than the existing approach. We should note that it is not necessary to update session information for a received packet belonging to a session that has already been classified. On the other hand, in the session-based approach, session information must be updated every time a packet is received until the session is terminated, since it generates session-based features from the session information. Considering these characteristics, we found that the proposed approach requires only 2 to 2.4 packets (on average) for each



	Benign	Bot	DDoS	DoS GoldenEye	DoS Hulk	DoS Slow HttpTest	DoS Slowloris	FTP-Patator	PortScan	SSH-Patator	Web Attack BF
Proposed	3.5	29	3.3	6.4	2.1	6	34.3	0.2	59.4	30.2	11.3
Global threshold	7.7	38.2	4.2	20.6	55.3	66.8	39.1	0.2	59.8	30.2	59.9
Session-based	64.2	38.5	76.9	83.5	117.1	109.8	116	64.5	60.1	36.1	61.1

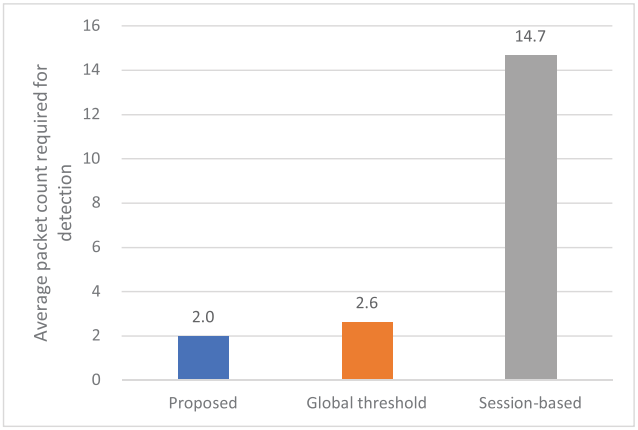
(a) CICIDS2017 dataset



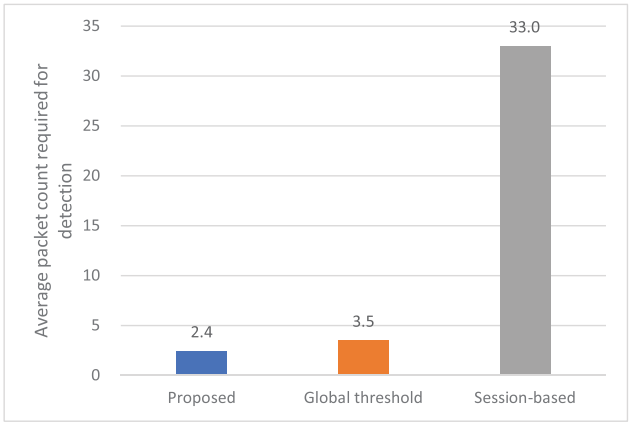
	Benign	BruteForceSSH	DDoS	HttpDoS	Infiltration
Proposed	10.4	0.8	6.8	26.9	18.7
Global threshold	15.6	23.8	22.7	46	52.6
Session-based	64.1	39.5	59.9	71	62.6

(b) ISCXIDS2012 dataset

FIGURE 5. Average detection time for each class when using the CICIDS2017 and ISCXIDSIDS2012 datasets.



(a) CICIDS2017 dataset



(b) ISCXIDS2012 dataset

FIGURE 6. Number of packets required for detection when using the CICIDS2017 and ISCXIDS2012 datasets.

TABLE 4. Comparison of the average number of memory accesses per session between the existing and the proposed approaches for each dataset.

Dataset	Proposed	Session-based	Ratio
CICIDS2017	2,708.7	2,889.2	1.28
ISCXIDS2012	5,365.8	4,655.0	1.15

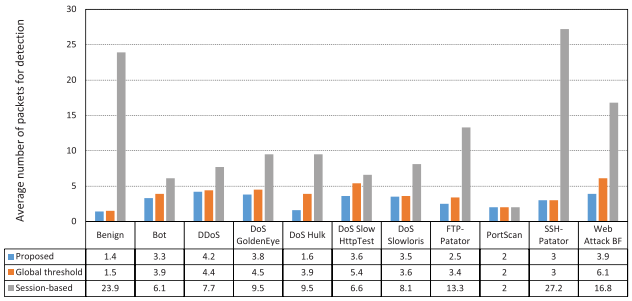
session, as shown in Fig. 6. This means that the proposed method requires at most 2.4 packet classifications (on average) per session, and also requires the same number of session information updates.

To maintain session information for feature sets in our proposed approach, 76 memory accesses are required for each packet received. In addition, the number of memory accesses required in RF was 1,772 and 2,147 (on average) for CICIDS2017 and ISCX2012, respectively. Based on this, the average number of memory accesses per session can be calculated. The results are shown in Table 4.

In Table 4, the number of memory accesses with the proposed approach increases surprisingly by just 28 % and 15 % for CICIDS2017 and ISCXIDS2012 datasets, respectively, compared to the existing session-based approach. The results imply that it is possible to implement an IDS supporting real-time attack detection using almost the same hardware as the existing session-based IDS, which only provides non-real-time attack detection.

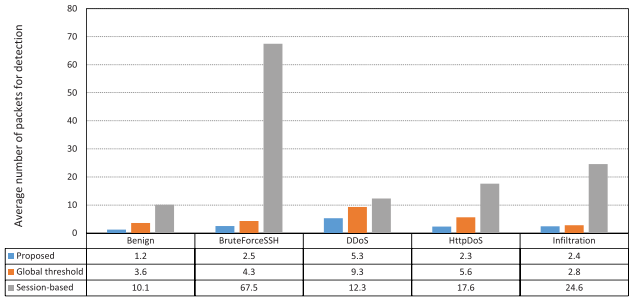
Fig. 7 shows the results from measuring the average number of packets required for packet detection of each class of CICIDS2017 and ISCXIDS2012 datasets. For all classes, the proposed approach requires fewer packets than the session-based and the global threshold-based approaches as shown in Fig. 7a. In addition, the average number for the session-based approach varied considerably (from two packets to 27.2 packets), depending on the class, whereas the proposed approach required from 1.4 to 4.2 packets, so the difference was very small. This means that only a small number of packets is needed, regardless of class types, so the proposed algorithm achieves fast detection for all classes. Compared with Fig. 5a, Bot had a very small inter-packet time, so the improvement in detection time seems small. However, in terms of the number of packets, we can see that the performance improvement almost doubled. On the other hand, PortScan had a very small total session length, so there was little margin for improvement in terms of detection time and system load.

Fig. 7b shows the average number of packets required for detection using the ISCXIDS2012 dataset. Similar to CICIDS2017, the proposed approach requires a very few packets, compared to the session-based or global threshold-based approaches. In particular, unlike with CICIDS2017, the length of each session was comparatively long, so we can see that the range in performance improvement also increased accordingly.



	Benign	Bot	DDoS	DoS GoldenEye	DoS Hulk	DoS Slow HttpTest	DoS Slowloris	FTP-Patator	PortScan	SSH-Patator	Web Attack BF
Proposed	1.4	3.3	4.2	3.8	1.6	3.6	3.5	2.5	2	3	3.9
Global threshold	1.5	3.9	4.4	4.5	3.9	5.4	3.6	3.4	2	3	6.1
Session-based	23.9	6.1	7.7	9.5	9.5	6.6	8.1	13.3	2	27.2	16.8

(a) CICIDS2017 dataset



	Benign	BruteForceSSH	DDoS	HttpDoS	Infiltration
Proposed	1.2	2.5	5.3	2.3	2.4
Global threshold	3.6	4.3	9.3	5.6	2.8
Session-based	10.1	67.5	12.3	17.6	24.6

(b) ISCXIDS2012 dataset

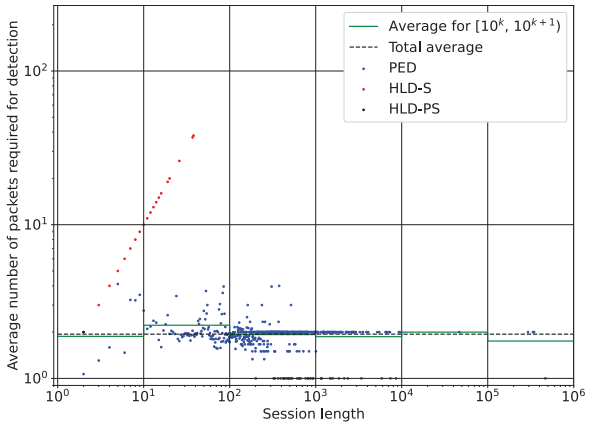
FIGURE 7. Average number of packets for detection of each class when using the CICIDS2017 and ISCXIDS2012 datasets.

B. SYSTEM LOAD FOR DETECTION

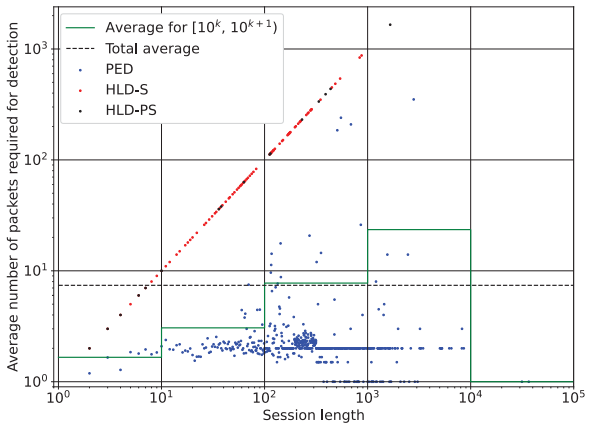
Fig. 8 shows the number of packets required for detection in each session for each dataset. In Fig. 8a, which is the results using CICIDS2017, we can see that the number of required packets remained almost constant as the length of the session increased. This means that even if the length of the session increases, the impact on the load of the system is very small, since it determined whether an attack existed or not after very few classifications. Therefore, we confirmed that the proposed approach can effectively maintain the system load, regardless of the length of the session. In Fig. 8a, the HLD is similar to the session-based approach, so the session length was the same as the number of packets required for detection. For this reason, the number of classifications in HLD increased as the session length increased. Nevertheless, since the overall average is kept low, it can be estimated that the lengths of the sessions processed in HLD were mostly very short, or there were very few long sessions. Fig. 8b shows the experimental results for ISCXIDS2012, which are slightly different from CICIDS2017. Since the number of sessions classified in HLD was larger than CICIDS2017, we can see that the average number of packets required for detection increased to some extent as the session length increased. Nevertheless, since the session length was relatively small, the overall average session length was still quite small, and thus, the overall system load was also very small.

C. COMPARISON OF DETECTION RATES

In order to compare the detection rates in detail, we measured and compared accuracy, precision, recall, and F1-score



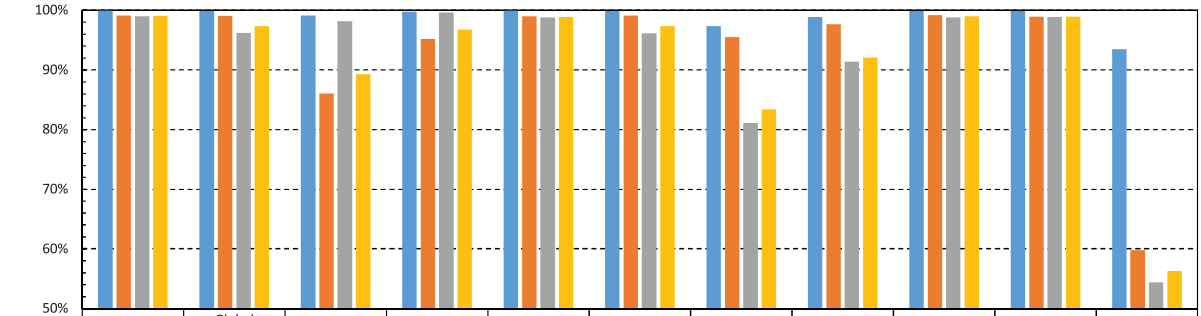
(a) CICIDS2017 dataset



(b) ISCXIDS2012 dataset

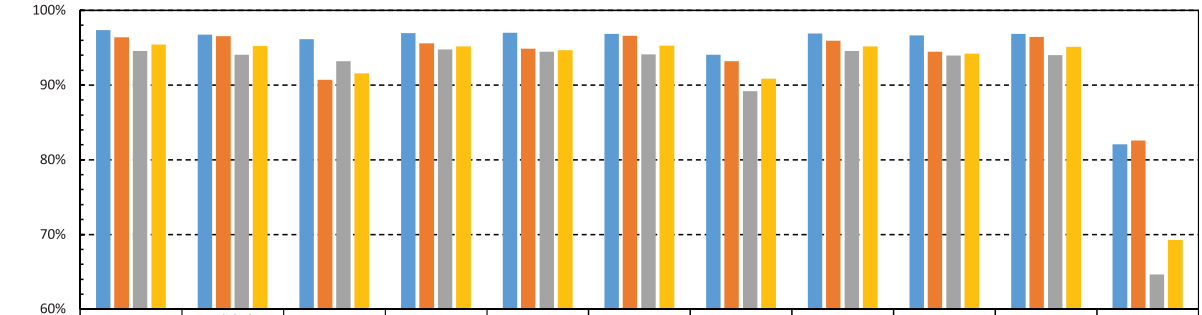
FIGURE 8. Length of each session versus the number of packets required for detection.

for each algorithm. Fig. 9 shows the experimental results for CICIDS2017 and ISCXIDS2012 datasets. As shown in Fig. 9a, the proposed approach has the highest results for almost all metrics, compared to the other approaches. Although our approach is mainly designed to detect attacks quickly (before the session terminates), its performance was rather high, compared to the existing session-based approaches. Among the compared algorithms, TSE shows the best performance, very similar to ours. ADT and GBT also show very high classification results, which are slightly worse than those of TSE. DTNB, ELM and SVM achieve lowest performance that at least 10 % lower f1-score than our proposed algorithm. TSE adopts three types of classifiers in the first stage and one additional classifier in the second stage, so this complex structure helps to achieve high accuracy, but the implementation complexity is very high and the classification speed is slow. However, our approach shows the best performance without such serious problems. In this respect, the proposed algorithm is superior compared to existing ones.



	Proposed	Global threshold	DTNB	SMOTE+RF	TSE	RF	ELM	DNN	ADT	GBT	SVM
Accuracy	99.90%	99.84%	99.10%	99.73%	99.90%	99.85%	97.32%	98.83%	99.87%	99.88%	93.44%
Precision	99.12%	99.07%	86.02%	95.20%	98.99%	99.11%	95.51%	97.65%	99.14%	98.94%	59.79%
Recall	98.96%	96.16%	98.13%	99.63%	98.76%	96.15%	81.06%	91.34%	98.77%	98.83%	54.36%
F1-score	99.04%	97.30%	89.28%	96.73%	98.87%	97.31%	83.35%	92.05%	98.95%	98.88%	56.27%

(a) CICIDS2017 dataset



	Proposed	Global threshold	DTNB	SMOTE+RF	TSE	RF	ELM	DNN	ADT	GBT	SVM
Accuracy	97.37%	96.75%	96.16%	96.94%	96.99%	96.84%	94.07%	96.88%	96.66%	96.85%	82.09%
Precision	96.38%	96.57%	90.71%	95.58%	94.85%	96.60%	93.18%	95.91%	94.44%	96.45%	82.58%
Recall	94.56%	94.06%	93.18%	94.79%	94.45%	94.13%	89.18%	94.55%	93.93%	94.00%	64.64%
F1-score	95.41%	95.24%	91.55%	95.16%	94.65%	95.29%	90.87%	95.17%	94.19%	95.14%	69.28%

(b) ISCXIDS2012 dataset

FIGURE 9. **Detection performance when using the CICIDS2017 and ISCXIDS2012 datasets.**

This means that the cooperative hybrid algorithms of PEM and HLD for the proposed approach significantly improve the performance, compared to the existing approach, in terms of accuracy and speed. In general, traditional ML-based IDS such as TSE are designed to target the high accuracy. However, PEM and HLD in our approach are each optimized for early detection time and classification accuracy. This unique structure allows our algorithms to outperform competing algorithms. Fig. 9b shows the results from experiments on ISCXIDS2012, and the proposed approach also showed the highest performance in most metrics, including accuracy, recall, and F1-score. Therefore, similar to CICIDS2017, high accuracy can be achieved with the sophisticated structure of PED and HLD in the proposed approach. Unlike previous experiments with the CICIDS2017 data set, SMOTE + RF shows the best classification results on the ISCXIDS2012 data set. The differences between SMOTE+RF and our approach are marginal, but the proposed algorithm outperforms SMOTE+RF in all metrics except for recall. DTNB and SVM show worst performance on ISCXIDS2012 either. Even using 11 algorithms and two datasets, our propose algorithm always show the best classification accuracy. We should note that some algorithms show almost similar classification performance but any other algorithms among compared ones cannot provide real-time classification. Only our proposed algorithm is designed to be able to detect intrusions on the fly with the highest detection accuracy. This experiment result reaffirms the advantage of our approach.

V. CONCLUSION

In this paper, we proposed an approach that can detect attacks in real time by solving the shortcomings in the existing session-based IDS (with detection most likely after an attack has been terminated). The proposed approach is composed of very elaborately designed packet- and session-based classifiers to improve classification accuracy and to support very high classification speed. Above all, while performing real-time detection, the number of memory accesses only increases by 15 % to 28 %, compared to existing session-based non-real-time IDSs. This means that a real-time IPS can be implemented on the existing session-based IDS platform without additional

high-performance hardware by extending our approach. Of course, there are several limitations to the proposed approach, which should be resolved before practical deployment. All classes have different processing speeds, and the detection speed may be slow depending on the specific class. In addition, it may be difficult to apply the proposed approach to an existing system, because implementation complexity and costs are higher than for existing IDSs composed of a simple classifier. Despite these shortcomings, the proposed approach has a very important strength in that it can provide real-time attack detection on an existing platform. Therefore, if these shortcomings are improved or relieved, the proposed approach is expected to be extended to real-time detection technology and be applicable against various attacks and in various environments. We hope our research will be of great help in keeping networks safe from increasing threats from malicious users.

REFERENCES

[1] D. Buil-Gil, F. Miró-Llinares, A. Moneva, S. Kemp, and N. Díaz-Castaño, "Cybercrime and shifts in opportunities during COVID-19: A preliminary analysis in the UK," *Eur. Soc.*, vol. 23, no. 1, pp. S47–S59, 2020.

[2] M. Roesch, "Snort: Lightweight intrusion detection for networks," in *Proc. USENIX LISA*, 1999, pp. 229–238.

[3] 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, 2018.

[4] C. Seelammal and K. V. Devi, "Computational intelligence in intrusion detection system for snort log using Hadoop," in *Proc. Int. Conf. Control, Instrum., Commun. Comput. Technol. (ICCICCT)*, Dec. 2016, pp. 642–647.

[5] 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)*, T. Yu, G. Danezis, and V. D. Gligor, Eds. Raleigh, NC, USA, Oct. 2012, pp. 833–844.

[6] 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, 2018.

[7] 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, 2018.

[8] M. Belouch and S. E. Hadaj, "Comparison of ensemble learning methods applied to network intrusion detection," in *Proc. 2nd Int. Conf. Internet Things, Data Cloud Comput. (ICC)*, New York, NY, USA, Mar. 2017, pp. 1–4.

[9] 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.

[10] 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, 2017.

[11] 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.

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

[13] Y. Otoum, D. Liu, and A. Nayak, "DL-IDS: A deep learning–based intrusion detection framework for securing IoT," *Trans. Emerg. Telecommun. Technol.*, p. e3803, 2019. [Online]. Available: https://onlinelibrary.wiley.com/action/showCitFormats?doi=10.1002%2Fett.3803, doi: 10.1002/ett.3803.

[14] 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.

[15] 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.

[16] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986.

[17] (2021). *Bluefield-2 DPU*. Accessed: Jan. 19, 2021. [Online]. Available: https://store.mellanox.com/categories/dpu/bluefield-2-dpu.html/

[18] (2021). *OCTEON TX2 LiquidIO III SmartNIC*. Accessed: Jan. 19, 2021. [Online]. Available: https://www.marvell.com /products/data-processing-units.html/

[19] (2021). *Alveo U250 Data Center Accelerator Card*. Accessed: Jan. 19, 2021. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/alveo/u250.html/

[20] S. Han, K. Jang, K. Park, and S. Moon, "Packetshader: A GPU-accelerated software router," in *Proc. ACM SIGCOMM Conf.*, New York, NY, USA, 2010, pp. 195–206.

[21] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy (ICISSP INSTICC)*, vol. 1. Setúbal, Portugal: SciTePress, 2018, pp. 108–116.

[22] V. Svetnik, A. Liaw, C. Tong, J. Culberson, R. Sheridan, and B. Feuston, "Random forest: A classification and regression tool for compound classification and QSAR modeling," *J. Chem. Inf. Comput. Sci.*, vol. 43, pp. 58–1947, Nov. 2003.

[23] Y. Coadou, "Boosted decision trees and applications," in *Proc. EPJ Web Conf.*, vol. 55, 2013, p. 02004.

[24] R. C. Bhagat and S. S. Patil, "Enhanced SMOTE algorithm for classification of imbalanced big-data using random forest," in *Proc. IEEE Int. Advance Comput. Conf. (IACC)*, Jun. 2015, pp. 403–408.

[25] M. Hall and E. Frank, "Combining naive Bayes and decision tables," in *Proc. 21th Int. Florida Artif. Intell. Res. Soc. Conf. (FLAIRS)*, Jan. 2008, pp. 318–319.

[26] B. A. Tama, M. Comuzzi, and K.-H. Rhee, "TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system," *IEEE Access*, vol. 7, pp. 94497–94507, 2019.

TAEHOON KIM was born in Daegu, South Korea, in 1995. He received the B.S. degree in information and communication engineering from Yeungnam University, in 2019, where he is currently pursuing the M.S. degree in information and communication engineering. His interests include artificial intelligence and network intrusion/prevention systems.

WOOGUIL PAK received the 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 was a Research Professor with the Junglee Research Institute for National Defence. In 2013, he was with Keimyung University, Daegu, South Korea. 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 1-Tb/s networks.