Flow based Sequential Grouping System for Malicious Traffic Detection

Jee-Tae Park[1], Ui-Jun Baek[1], Min-Seong Lee[1], Young-Hoon Goo[2], Sung-Ho Lee[3] and Myung-Sup Kim[1*]
[1]Dept. Of Computer and Information Science
Korea University, Korea
[E-mail: {pjj5846, pb1069, min0764, tmskim}@korea.ac.kr]
[2]Advanced KREONET Center, Korea Institute of Science and Technology Information
Daejeon, Korea
[E-mail: gyh0808@kisti.re.kr]
[3]AhnLab, Korea
[E-mail: sungho.lee@ahnlab.com]
*Corresponding author: Myung-Sup Kim

Received December 7, 2020; revised June 13, 2021; revised July 18, 2021; accepted September 16, 2021; published October 31, 2021

Abstract

With the rapid development of science and technology, several high-performance networks have emerged with various new applications. Consequently, financially or socially motivated attacks on specific networks have also steadily become more complicated and sophisticated. To reduce the damage caused by such attacks, administration of network traffic flow in real-time and precise analysis of past attack traffic have become imperative. Although various traffic analysis methods have been studied recently, they continue to suffer from performance limitations and are generally too complicated to apply in existing systems. To address this problem, we propose a method to calculate the correlation between the malicious and normal flows and classify attack traffics based on the corresponding correlation values. In order to evaluate the performance of the proposed method, we conducted several experiments using examples of real malicious traffic and normal traffic. The evaluation was performed with respect to three metrics: recall, precision, and f-measure. The experimental results verified high performance of the proposed method with respect to first two metrics.

Keywords: Traffic Classification, Flow Correlation Index, Malicious Traffic Detection, Flow Information

This research was supported by a Korea University Grant and supported by the Technology Innovation Program grant funded By the Ministry of Trade, Industry & Energy (MOTIE, Korea) and the Korea Evaluation Institute of Industrial Technology (KEIT) (No. 20008902, Development of SaaS SW Management Platform based on 5Channel Discovery technology for IT Cost Saving)

1. Introduction

With the rapid development of science and technology over the recent years, the variety of commonly-occurring internet traffic and types of application have also increased, consequently expanding network environments. Recently, this phenomenon has been further accelerated by developments such as 5G mobile telecommunication and edge computing.

However, the frequency of malicious traffic has also increased simultaneously, and attack patterns have been diversified. This diversification has expanded the range of possible damages, including personal and confidential information leaks and incapacitation of certain corporate services, alongside the risk of financial damages. For instance, the quantity of emails containing malicious code disguised as important communication sent from public institutions to individuals have risen significantly. If the victim clicks the link included in such an email, the embedded malicious code is automatically executed on the victim's computer and personal information is stolen. Additionally, instances of services of important financial or administrative institutions of the country being halted by attacks through malicious traffic have also become more frequent [7-9].

To reduce the damage caused by and to prevent such malicious attacks, network administrators are required to establish effective network management and security policies, and accurate detection and analysis of malicious behavior based on these policies has become imperative. Traffic classification is a field that has been studied for a long time and is the most basic field of research on malicious traffic detection and analysis [1-5]. There are various methods of traffic classification, such as payload signature-based methods, which use patterns of traffic flows, and deep learning-based classification methods, which operate by learning the traffic features.

The payload signature-based methods exhibit the best performance in terms of accuracy and completeness. Various studies have been conducted on payload signature-based methods in traffic classification. However, these suffer from the problems of high computational duration and cost during the processing of a significant amount of traffic in real-time [10-14]. In particular, addressing this problem is critical to the maintenance of a high-speed network and managing a high volume of traffic data in accordance with the recent computational demands.

To address the aforementioned problems in payload signature-based traffic classification methods, a deep learning-based traffic classification method has been studied. Besides learning the payload of traffic flow corresponding to a specific application, such a method is also capable of learning the characteristics of flow, such as flow size or the number of associated packets. Based on the particular features of incoming traffic, deep learning classifies it as malicious flow or normal flow.

The deep learning-based traffic classification method is capable of addressing the problems faced by payload signature-based traffic classification methods, such as encrypted traffic classification or high computational cost and duration, and exhibits high performance in terms of accuracy. However, it remains highly dependent on the training data. If wrong data is included in the training set or the amount of training data is inadequate, it becomes difficult to extract satisfactory performance from the method [17-19]. In the field of network security, it is difficult to obtain raw data corresponding to malicious traffic and the selection of proper learning features to enable correct classification of malicious traffic is a challenging problem. Even with well-performing deep learning models and adequate raw data, the selection of inappropriate learning features could yield poor results [20-23].

Therefore, a significant amount of research has been conducted to ascertain a method of selecting proper learning features. In addition to the two aforementioned methods, new ones such as deep packet inspection (DPI)-based methods are also being developed [6, 24].

In this paper, we propose a method to detect malicious traffic by comparing the statistical characteristics of network flows. We calculate various features and information of a network flow and define the flow correlation index (FCI). By comparing the flow correlation indices of malicious traffic and normal traffic, we define a threshold for each flow and classify malicious flow based on these thresholds.

The proposed method is similar to the deep learning-based traffic classification method as its operation is based on the study of past malicious traffic. However, the proposed method calculates the flow correlation index according to clearly defined features, disposing the necessity of selecting appropriate learning features. Moreover, as the flow correlation index is calculated based on the characteristics of the flow, it makes it possible to clearly classify malicious flows which comprise a mixture of malicious traffic and normal traffic.

The rest of this paper is organized as follows. In Section 2, we will discuss the related work and, describe the detailed algorithm in Section 3. The experiments that have been conducted to evaluate the proposed method using actual malicious traffic are presented in Section 4. Finally, we conclude the paper and outline future research directions in Section 5.

2. Related Works

As mentioned in Section 1, traffic classification has been a well-studied topic over a significant duration because of its importance with respect to efficient network management and network security. Especially in the field of network security, several solutions exist that protect a system from intrusion, such as firewalls, anti-virus software, and authentication systems. However, the protection and prevention of intrusion depends on the successful detection of malicious traffic. Thus, precise identification of malicious traffic and its subsequent analysis is crucial in the defense against malicious attacks. Several methods of traffic classification exist, including port-based, signature-based and deep learning-based ones. The most widely studied methods are payload signature-based ones and deep learning-based ones.

Signature-based classification methods can, in turn, be sub-classified into statistical information-based, header information-based, and payload-based methods. The statistical information-based approach uses statistic information such as flow size, sequence, and vectors, but takes a long time to generate a signature and suffers from low accuracy.

The header information-based approach uses flow header information such as IP addresses and port numbers. However, they do not use specific data to generate signatures, and are, therefore, not suitable for general use, because the data used to generate the signature is liable to change.

The payload signature-based approach uses an automatic signature generation system to automatically extract the payload signatures of the target flows. A payload signature is a unique pattern corresponding to a particular application. This implies that network traffic generated in the same application has the same payload pattern, and such traffic can be classified by comparing its signature to patterns corresponding to the same application. Malicious traffic shares a common payload pattern just like normal network traffic and, thus, can be detected based on it. Although signature based-method exhibit high detection rate, accuracy, and coverage, they also suffer from several limitations.

Firstly, significant computational duration and cost is required to generate a traffic's payload signature. This is because signatures can be defined only by verifying all substrings that are common among the payload contents. In addition, the process of finding a common substring in the payload is complicated, which burdens the user. To solve this problem, a significant amount of research has been conducted to automatically generate payload signatures such as LASER (LCS-based Application Signature ExtRaction) [13, 14].

LASER automatically generates an application signature, in the form of a sequence of substrings, by using a modified version of the LCS (longest common subsequence) algorithm [5]. The inputs of the LCS algorithm are two distinct byte streams of packet payloads and the extracted signature of the traffic. This method finds a common sequence of strings by using a backtracking matrix and comparing pairs of strings. Although payload signatures can be generated automatically using this method, it takes a long time to do so, like in the case of existing problems [10-12].

Deep learning-based classification methods classify the traffic by learning its characteristics. Recently, with the development of artificial intelligence (AI) technology, such methods have been heavily studied, and its high performance and ease of learning have been established. In addition, as various algorithms, such as convolutional neural network (CNN) and recurrent neural network (RNN) in deep learning, can be applied to classify malicious traffic. Since deep learning is being studied in many fields, classification method based deep learning has a high possibility of development [25, 26].

However, deep learning-based classification methods suffer from the limitation of having to ensure the quality of several factors such as learning data, deep-learning model, algorithm, and proper feature selection in order to ensure high performance. Among these factors, the selection of proper features is the most important, and several studies have been conducted to ascertain a method for this purpose [22]. Although several such methods exist, most of them are limited in usability, and are difficult to use universally because the features can be defined differently depending on the data.

3. Proposed Method

In this section, we explain the basic concept, system structure, and detailed algorithm of the method proposed. The structure of proposed method has been depicted in **Fig. 1**. The structure of the FCI system consists of Training and Testing.

3.1 Structure of the FCI System

Training is the process of determining the criteria based on which malicious flows are detected by calculating the characteristics of pre-classified malicious and normal traffic. This criteria value for classification is defined as a threshold and the set of thresholds is defined as a guideline.

First, pre-classified malicious and normal traffic are separately fed into the system as inputs. The format of the traffic is taken to be pcap, which is preprocessed into a file in the fwp format. After preprocessing, Seed is generated by the seed generation module with inputs from malicious fwp files. The Seed is a text file that contains 5-tuples of malicious traffic. This is a fundamental concept in FCI Systems. We calculate the flow correlation index of the flow corresponding of a particular seed information and a given target network flow. After preprocessing and generation of seeds, guidelines are generated by the guideline generation module based on inputs from the Seed and preprocessed traffic. As mentioned previously, the guideline includes the thresholds that act as the criteria for classification. After the operation of the guideline generation module, the optimization process is performed, and as a result, an optimized guideline is created.

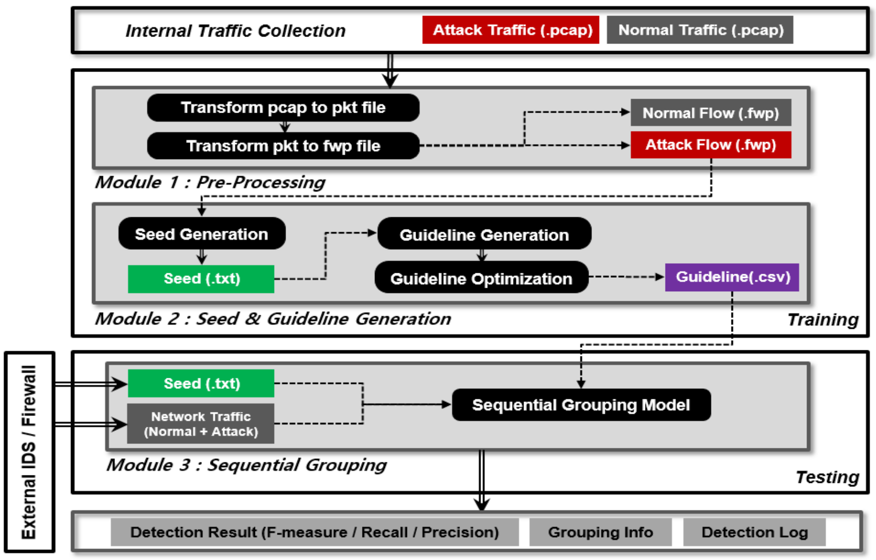

Internal Traffic Collection
Attack Traffic (.pcap)
Normal Traffic (.pcap)
Transform pcap to pkt file
Transform pkt to fwp file
Normal Flow (.fwp)
Attack Flow (.fwp)
Module 1 : Pre-Processing
Seed Generation
Seed (.txt)
Guideline Generation
Guideline Optimization
Guideline(.csv)
Module 2 : Seed & Guideline Generation
Training
External IDS / Firewall
Seed (.txt)
Network Traffic (Normal + Attack)
Sequential Grouping Model
Module 3 : Sequential Grouping
Testing
Detection Result (F-measure / Recall / Precision)
Grouping Info
Detection Log

Fig. 1. Entire Structure of Sequential Grouping Model

Testing is the process of classifying input traffic as malicious or normal flow based on the guideline created during Training. It distinguishes malicious flow from network traffic based on seed information obtained via an external IDS (Intrusion Detection System) or firewall and the guideline obtained via Training. Malicious flows are extracted and classified from the target network traffic in the sequential grouping model module and the final output is derived. The final output includes detection results, classified information (grouping information), and the detection log. The detection results consist of recall, precision, and f-measure. The detailed algorithms and module descriptions are described as follows.

3.2 Flow Correlation Index (FCI)

The most basic concept pertaining to the proposed method is the flow correlation index. Any network traffic consists of a certain number of packets, which carry certain information such as source IP, destination IP, etc. A network flow is defined as the collection of packets that share the identical 5-tuples of information (i.e. source IP, source port, destination IP, destination port, protocol). A session is defined as a bidirectional flow in network traffic [2].

As the characteristics of flows occurring in the same session tend to be similar, we numerically calculate the characteristics of the two flows in a session and define their similarity to be the flow correlation index. Flow correlation index is, thus, composed of two indices: the similarity index and the connectivity index. The similarity index is a value that encodes the similarity between the statistical characteristics of the two flows, and the connectivity index is a value that encodes the similarity between the header information of the two flows.

The features of the similarity index have been described in **Table 1**. We use packet inter arrival time (PIT) and packet size distribution (PSD) of the two flows to calculate the corresponding similarity index. PIT5 and PSD5 denote the features of the first 5 packets except the TCP 3-way handshake.

Table 1. Explanation of the Similarity Features

Feature	Explanation	Function	Range
PIT_Mean PIT5_Mean	PIT = Packet Inter Arrival Time of the Flow	$\alpha_{PIT}(f_x) = \frac{\sum_{i=1}^{m} T_i}{m}$ (m : The number of Packet in Flow f_x T_i : Inter Arrival Time)	0~1
	PIT 5 = PIT of first 5 packets		
PSD_Mean PSD5_Mean PSS5	PSD = Packet Size Distribution of the Flow	$\alpha_{PSD}(f_x) = \frac{\sum_{i=1}^{m} P_i}{m}$ (m : The number of Packet in Flow f_x P_i :Payload Length)	
	PSD 5 = PSD of First 5-packets		
	PSS 5 = Packet Size Sequence of First 5 Packets		

To keep the calculations objective, we used Min-Max Normalization to change the feature value over a wide distribution to a value between 0 and 1. The feature values corresponding to the two flows were calculated as a similarity index using the Euclidean Distance. Euclidean Distance is the formula for the distance between two points in n-dimensional space. Euclidean Distance can be used to express the similarity of flow characteristics by expressing each feature value between the two flows as multidimensional coordinates. The higher the similarity between the two flows, the smaller is the difference between the values of each feature. Therefore, the similarity index is defined as the difference calculated using the Euclidean distance as shown in Eq. (1).

$$\mathrm{SI}(f_x, f_y) = 1 - \sqrt{\sum_{i=1}^{5} \left(feature_i(f_x) - feature_i(f_y)\right)^2} \quad (1)$$

We use the 5-tuples of information corresponding to the flow as a feature to obtain the connectivity index. The features of the similarity index have been described in **Table 2**. In this table, ST denotes the start time of the flow, and calculates the similarity between the start times of the two flows. SIP and DIP calculate the connectivity of the source and destination IP addresses of the flow, respectively. The connectivity of IP Address is calculated by reflecting the same number of bits in the 32-bit address. Similarly, SPT and DPT denote the connectivity of the source and destination ports of the flow, respectively. The connectivity of the ports is calculated by reflecting the same number of bits among 16 bits. PROT takes the value 1 if the protocols of the two flows are the same and 0 if they are different. The connectivity index is calculated by combining the features with appropriate weights.

Table 2. Explanation of the Connectivity Features

Feature	Explanation	Function	Range
ST	Start Time	$\alpha_{st}(f_x, f_y) = 1 - \frac{dist(f_x, f_y)}{MAX\ Interval}$	0~1
SIP	Source IP Address	$\alpha_{IP}(f_x, f_y) = \left\{\frac{prefixlenAddr(f_x, f_y)}{32}\right\}^2$	0~1
DIP	Destination IP Address		
SPT	Source Port Number	$\alpha_{PT}(f_x, f_y) = \left\{\frac{prefixlenPort(f_x, f_y)}{16}\right\}^2$	0~1
DPT	Destination Port Number		
PROT	L4 Protocol	$\alpha_{PROT}(f_x, f_y) = \begin{cases} \text{mean}: f_x.\text{PROT} \neq f_y.\text{PROT} \\ 1: f_x.\text{PROT} = f_y.\text{PROT} \end{cases}$	Mean or 1

The weights are greater than 0, less than 1, and the sum of all the weights must be 1, because the weights represent the reflection ratio of each feature in the connectivity index. Smaller weight units correspond to higher levels of sophistication of the resultant threshold. However, setting the weight unit to be too small increases the duration of the process, and so proper definition of the weight unit is necessary. We defined the weight variance unit to be 0.01. Therefore, the connectivity index is defined as the sum of multiplied values between feature values and weights as shown in Eq. (2).

The similarity and connectivity indices are calculated as numbers lying between 0 and 1. If the values are close to 1, the two flows are deemed to be similar, and if the values are close to 0, they are deemed to be dissimilar. For example, if the flow correlation index between a malicious flow and a normal flow is calculated, the value is close to 0. In contrast, the flow correlation index between a malicious flow and another malicious flow is observed to be close to 1.

$$\mathrm{CI}(f_x, f_y) = \left(w_{ST} \times f_{ST}(f_x, f_y)\right) + \left(w_{IP} \times f_{IP}(f_x, f_y)\right) + \left(w_{Pt} \times f_{Pt}(f_x, f_y)\right) + \left(w_{PROT} \times f_{PROT}(f_x, f_y)\right) \ (\text{where}, \textstyle\sum_{i=1}^{4} w_i = 1) \quad (2)$$

3.3 Sequential Grouping Model

As mentioned previously, any network traffic consists of a large number of sessions and flows. Similarly, malicious traffic also consists of a large number of malicious flows. Our main goal is to classify normal and malicious flows precisely, and the basic aim of the proposed method is to identify malicious flows that are similar to known malicious flows by comparison. The target network traffic is generally commonly used network traffic that consists of normal and malicious flows, and a malicious flow used for the purpose of comparison is defined as the seed flow

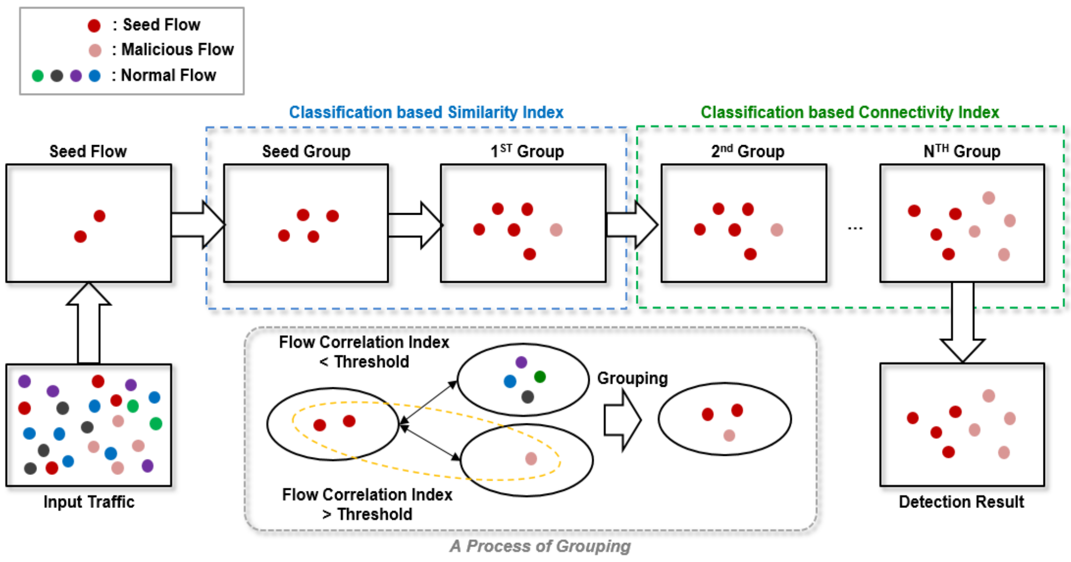


Fig. 2. An Example of Sequential Grouping Model

Grouping is the process of classifying similar flows based on the respective flow correlation indices and pre-defined thresholds between pairs of flows. The grouping process proceeds sequentially beginning with the seed flow, and the group classified during the first grouping operation is defined to be the seed group. Flows in the seed group consist of flows that are almost similar to seed flows. The flow correlation indices between one seed flow and target flows are repeatedly calculated, and flows with similar flow correlation indices are grouped together based on the defined threshold. In summary, in order to detect malicious flows in a target network traffic, the flow correlation index between the target network traffic and the seed flow is calculated, and the target flow is classified based on the value of the index and a predefined threshold. The overall process of the sequential grouping model has been depicted in **Fig. 2**.

The threshold is a criterion for grouping, and so the detection performance is significantly dependent on the threshold setting. This necessitates the use of a sophisticated threshold setting algorithm. If the flow correlation index between two flows is calculated to be greater than the threshold, the two flows are considered to be similar, and the grouping process is performed. If the threshold is set to too high a value, there are several malicious flows that correlation index with seed flow is lower than a threshold. In that case, there will be a large number of undetected malicious flows. In contrast, if the threshold is set too low, there are several normal flows that correlation index with seed flow is larger than a threshold. In that case, there will be a large number of false positive malicious flows. The process of the sequential grouping model based on the flow correlation index and threshold have been depicted in **Fig. 2**. The grouping process proceeds sequentially and is repeated until there are no more similar flows. The threshold setting algorithm will be explained in greater detail in Section 3.4.

3.4 Threshold Setting Algorithm

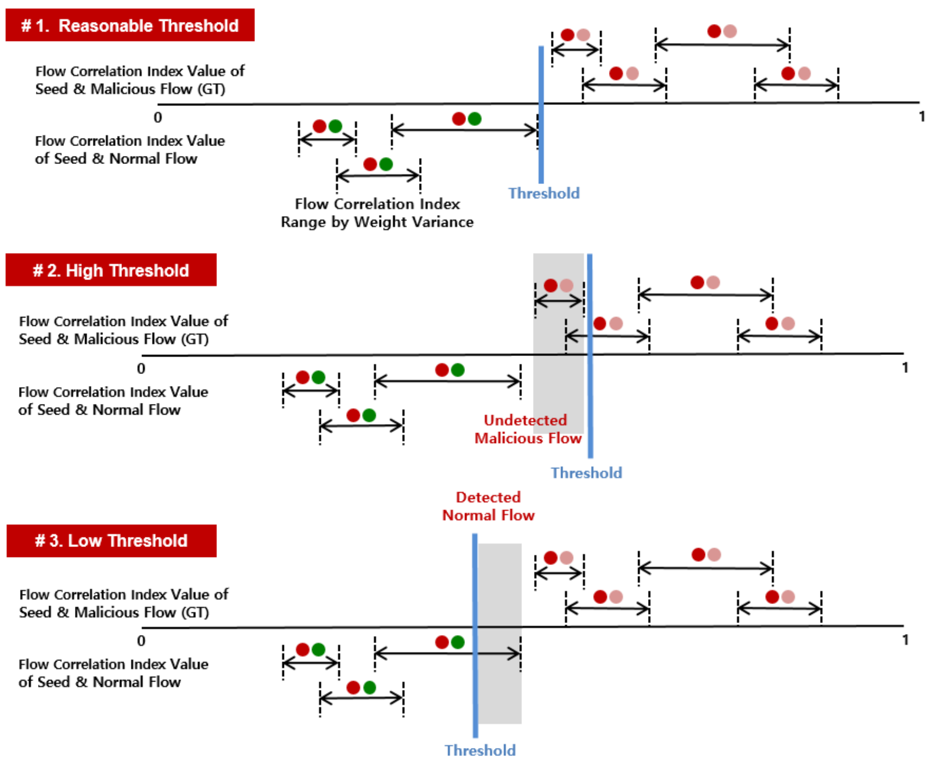
As depicted in **Fig. 2**, the first grouping of the filter into the seed group proceeds by calculating the similarity index, and the second grouping is performed by calculating the connectivity index. Grouping based on similarity and connectivity continues repeatedly. The flow correlation index of each pair of flows is compared with the thresholds for each sequence of grouping, and grouping proceeds in a similar manner for each sequence.

The threshold varies depending on the particular grouping sequence, necessitating its specification for each sequence. Therefore, when the flow correlation index is calculated for each grouping sequence, the weight value and threshold are defined as guidelines. Thus, for each sequence, grouping is performed based on the weight and threshold defined in the guideline. As depicted in **Fig. 3**, three separate cases arise while setting the threshold based on the flow correlation index.

In the first case, the value of the connectivity index varies depending on the weight combinations. As in the first case, proper thresholds can minimize the number of false positives and false negatives. Via proper threshold setting, all malicious flows can be accurately detected. However, if the threshold set too high, several malicious flows will remain undetected. In this case, although the detection accuracy is 100%, the detection rate is low. In contrast, if the threshold set too low, although all malicious flows will be detected, normal flows can also be falsely detected as malicious. In this case, the detection rate is 100%; however, the detection accuracy is low.

As with the first case, appropriate threshold setting criteria are required. We define the largest flow correlation index value between the seed flow and the normal flow as the threshold to maximize detection rate and accuracy. The threshold setting algorithm stores the threshold and weight combination in the guideline. The same process is repeated until the grouping no longer occurs.

.



# 1. Reasonable Threshold
Flow Correlation Index Value of Seed & Malicious Flow (GT)
Flow Correlation Index Value of Seed & Normal Flow
0
1
Flow Correlation Index Range by Weight Variance
Threshold
# 2. High Threshold
Flow Correlation Index Value of Seed & Malicious Flow (GT)
Flow Correlation Index Value of Seed & Normal Flow
0
1
Undetected Malicious Flow
Threshold
Detected Normal Flow
# 3. Low Threshold
Flow Correlation Index Value of Seed & Malicious Flow (GT)
Flow Correlation Index Value of Seed & Normal Flow
0
1
Threshold

Fig. 3. An Example of Threshold Setting Algorithm

However, as the behavior of malicious traffic is more complicated and sophisticated, there exist several types of malicious traffic which are highly similar to normal traffic. In this case, the flow correlation index between the seed flow and a malicious flow can be lower than the maximum value of flow correlation index between the seed flow and the normal flows. If the flow correlation index between the seed flow and a malicious flow is lower than flow correlation index between the seed flow and a normal flow, undetected malicious flows will occur as depicted in the second case presented in **Fig. 3**.

To address this problem, we apply the threshold balancing algorithm to adjust the connectivity index of normal flows. An example of applying threshold balancing has been depicted in **Fig. 4**. Threshold balancing is the process of readjustment of weights when the maximum flow correlation index with normal flows is less than the minimum flow correlation index with malicious flows.

As mentioned in Section 3.2, individual connectivity features are calculated and multiplied with their respective weights to obtain the connectivity index and define the threshold. In order to obtain a reasonable threshold, we can adjust the weights corresponding to specific connectivity features. For example, the start times and ports of malicious flows can be similar to those of normal flow. In this case, the corresponding feature values are higher than those for several normal flows, leading to the detection of the associated malicious flow as a normal flow. To address this shortcoming, the weights are adjusted by assigning lower weights to the start time and port features and higher weights to other features. The detailed threshold balancing algorithm has been described in Algorithm 1.

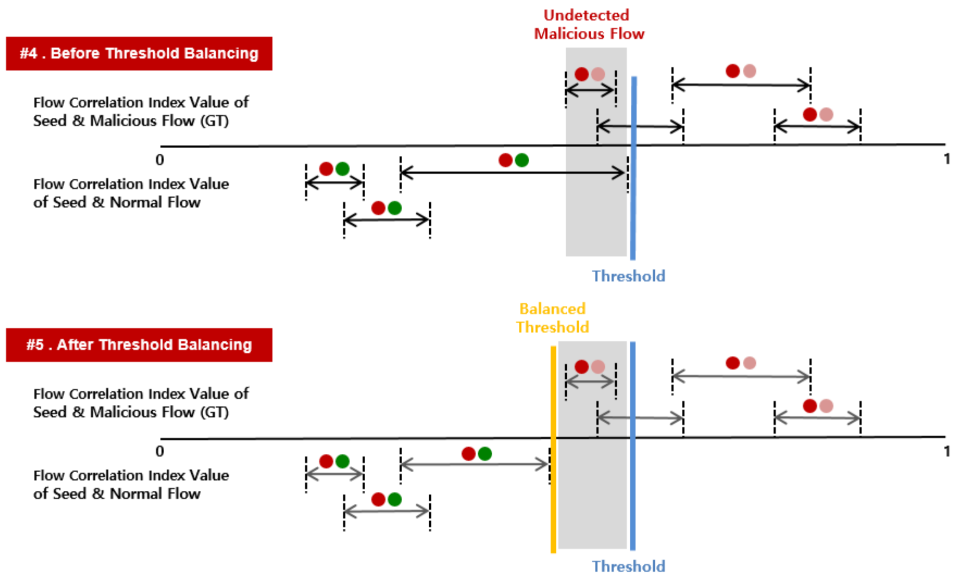

#4 . Before Threshold Balancing
Undetected Malicious Flow
Flow Correlation Index Value of Seed & Malicious Flow (GT)
0
1
Flow Correlation Index Value of Seed & Normal Flow
Threshold
#5 . After Threshold Balancing
Balanced Threshold
Flow Correlation Index Value of Seed & Malicious Flow (GT)
0
1
Flow Correlation Index Value of Seed & Normal Flow
Threshold

Fig. 4. An Example of Applying the Threshold Balancing Algorithm

The first step of Algorithm 1 is similarity threshold balancing. The initial similarity threshold is set to 1, and the similarity index is compared with the number of flows in the traffic. If the initial similarity threshold is greater than the similarity index of the flow, and the maximum similarity index with normal flows is smaller than the similarity index of the flow, it is confirmed if the flow has been grouped (in Alg. 1 line 1-4). If the flow has not been grouped, the initial similarity threshold is set to the similarity index of the flow, and similarity threshold is set to initial similarity threshold (in Alg. 1 line 6).

Subsequently, the maximum value of the connectivity index with malicious flows and the minimum value of the connectivity index with normal flows are compared (in Alg.1 line 7, 8). If the minimum connectivity index with normal flows is larger, the largest connectivity features with that flow are compared with the smallest connectivity features with the malicious flow (in Alg.1 line 9-11). If the two values are identical, the second largest connectivity feature of the malicious flow is obtained (in Alg.1 line 13). If the two values are different, the weight of the malicious flow is increased and the normal flow weight is adjusted by decreasing it. If the maximum value of the connectivity index with malicious flow is smaller than the minimum value of the connectivity index with normal flow, the maximum value of the connectivity index with malicious flow is set to be the threshold (in Alg.1 line 17, 18). If this process is repeated until the weight of malicious flow reaches 1, it is defined to not have been adjusted and the minimum value of normal flow connectivity index is set to be the threshold (in Alg.1 line 19, 20).

After threshold balancing is completed, a threshold and weight combination for one particular malicious traffic emerges from the guideline. However, there several malicious flows exist within one particular malicious traffic, and the detection rate and accuracy vary over the malicious flows

Algorithm 1. Pseudo Algorithm for Threshold Balancing

Notation - M : Malicious GT Flows / N : Normal GT Flows / F : Feature
SI : Similarity Index / CI : Connectivity Index / F : Feature Value

Input : Flows, Initial TH_{sim}, Initial TH_{con} / Output : balanced TH_{sim} and TH_{con}

```
1     Initial_$TH_{sim}$ = 1.0 // Similarity Threshold Balancing
2   for i=1 to Numbers of Flow
3       If Initial_$TH_{sim}$ > $Flow_i$SI and N.$SI_{max}$ < $Flow_i$SI
4           If $Flow_i$ is not Grouped
5               Initial_$TH_{sim}$ = $Flow_i$_SI
6               $TH_{sim}$ = Initial_$TH_{sim}$
7   if M.$CI_{min}$ < N.$CI_{max}$ // Best case
8           $TH_{con}$ = N.$CI_{max}$
9   else if M.$CI_{min}$ > N.$CI_{max}$ // Usual case
10      $F_M$ = find the MAX Feature in M.$CI_{min}$
11      $F_N$ = find the MAX Feature in N.$CI_{max}$
12      if $F_M$ == $F_N$// Select the Feature
13          $F_N$ = find the Second largest Feature in N.$CI_{max}$
14      while true// Connectivity Threshold Balancing
15          Increase the $F_M$.weight and Decrease the $F_N$.weight
16          Figure out the balanced M.$CI_{min}$ and N.$CI_{max}$
17          if M.$CI_{min}$ < N.$CI_{max}$
18              $TH_{con}$ = N.$CI_{max}$
19          if $F_M$.weight == 1.0 // balancing fail
20              $TH_{con}$ = A.$CI_{min}$
21  return $TH_{sim}$ and $TH_{con}$
```

In addition, threshold balancing is difficult to apply in this case because the threshold of each flow in the malicious traffic is different. Therefore, to address this problem, we define an optimal threshold that reflects the detection accuracy of each flow in a single traffic and this process called threshold optimization. Since the precision can only be calculated when there is a distinction between malicious and normal flows, we proceed with threshold optimization from Training in Section 3.1. The entire process of threshold optimization has been presented in Algorithm 2.

$$GL_i\_TH\_Rating = G{:}Flows_M \times Precision = G{:}Flows_M \times \frac{G{:}Flows_M}{G{:}Flows_M + G{:}Flows_N} \qquad (3)$$

$$G{:}Flows_M = Grouped\ Malicious\ GT\ Flows\ \ /\ G{:}Flows_N = Grouped\ Normal\ GT\ Flows$$

$$GL_i\_TH\_Rated = GL_i\_TH\_Rating \times GL_i\_Threshold \qquad (4)$$

$$GL_i\_TH_{Optimal} = \frac{\sum_{i=1}^{Numbers\ of\ GL} GL_i\_TH\_Rated}{\sum_{j=1}^{Numbers\ of\ GL} GL_j\_TH\_Rating} \qquad (5)$$

To execute threshold optimization, we obtain two values referred to as Threshold Rating and Threshold Rated. As presented in Eq. (3), Threshold Rating is the product of the precision and the number of flows detected when the flow is used as a seed flow. As given in Eq. (4), Threshold Rated is the product of the previously calculated Threshold Rating and the threshold. An optimal threshold is the average Threshold Rated value of the guideline. As given by Eq. (5), it is the sum of the Threshold Rated values divided by the sum of Threshold Rating Values.

Algorithm 2. Pseudo Algorithm for Threshold Optimization

Notation - M : Malicious GT Flows / N : Normal GT Flows / TH : Threshold
SI : Similarity Index / CI : Connectivity Index / GL : Guideline / T : Traffic Trace

Input : Flows, GLs / Output : Optimal GLs and Converged GL

```
1    For x = 1 to The Number of Traffic Traces
2        For y = 1 to The Number of Guidelines  // Get Grouped Flow Count
3            For i = 1 to The Number of Flows
4                if flow_i GroupType == Similarity
5                    $T_xGL_ySI$.count++
6                else if flow_i GroupType == Similarity
7                    $T_xGL_yCI$.count++
8                $T_xGL_ySI$.precision = precision($T_xGL_ySI$.count)
9                $T_xGL_yCI$.precision = precision($T_xGL_yCI$.count)
10               // Get Rating SI and CI (Eq. 3)
11               $T_xGL_y$Rating = getRating($T_xGL_ySI$, $T_xGL_yCI$)
12               // Get Rated Threshold (Eq. 4)
13               $T_xGL_y\text{RatedTH}_{sim}$ = getRatedTH($T_xGL_y\text{TH}_{sim}$, $T_xGL_y$Rating)
14               $T_xGL_y\text{RatedTH}_{con}$ = getRatedTH($T_xGL_y\text{TH}_{con}$, $T_xGL_y$Rating)
15           // Get Optimal Guideline (Eq. 5)
16           $T_x\text{OptimalTH}_{sim}$ = getRatedTH($T_xGL_y\text{RatedTH}_{sim}$)
17           $T_x\text{OptimalTH}_{con}$ = getRatedTH($T_xGL_y\text{RatedTH}_{sim}$)
18           // Get Optimal Guideline (Eq. 5)
19   $\text{ConvergedTH}_{sim}$ = getConvergedTH( $T_x\text{OptimalTH}_{sim}$)
20   $\text{ConvergedTH}_{con}$ = getConvergedTH ( $T_x\text{OptimalTH}_{con}$)
21   return  $\text{ConvergedTH}_{sim}$ and $\text{ConvergedTH}_{sim}$
```

Following threshold optimization, each malicious traffic gives rise to a separate guideline. However, it is difficult to detect all malicious traffic with only one optimal guideline because a different optical guideline is derived for each trace. Therefore, it is necessary to converge the optimal guidelines generated for each trace into one converged guideline. The process of combining the guidelines has been presented in **Fig. 5** and Algorithm 2.

The convergence process averages the thresholds corresponding to each guideline and stores them in one converged guideline (in Alg. 2 line 19, 20). As the trace increases, we can derive a more sophisticated converged guideline. This process yields a converged guideline that reflects the optimal guidelines of multiple traces, which can increase coverage for a single malicious type.

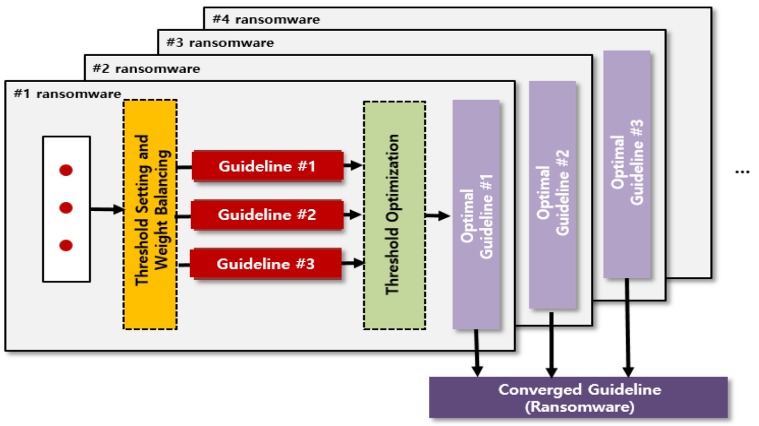


Fig. 5. A Process of Converging the Optimal Guidelines

3.5 Multiple Seeds and Guidelines for Better Detection

However, as mentioned in Section 2, several kinds of malicious traffic exist in the network, and each type of malicious traffic exhibits distinct attack patterns and characteristics. Even if a unified guideline is created via guideline optimization and convergence, the performance is liable to suffer if the characteristics of different types of malicious traffic types vary widely. To address this issue, we apply multiple seeds and guidelines to improve detection performance. An example of applying multiple guidelines has been depicted in **Fig. 6**, and an example of applying multiple seeds has been depicted in **Fig. 7**.

Multiple guidelines are applied when characteristics and attack patterns are different for different types of malicious traffic by generating converged guidelines for malicious traffic types in advance and applying them to network traffic. As a result of applying multiple guidelines, it is possible to identify which malicious flows are included in network traffic. Therefore, multiple guidelines are used to improve performance against various types of malicious traffic.

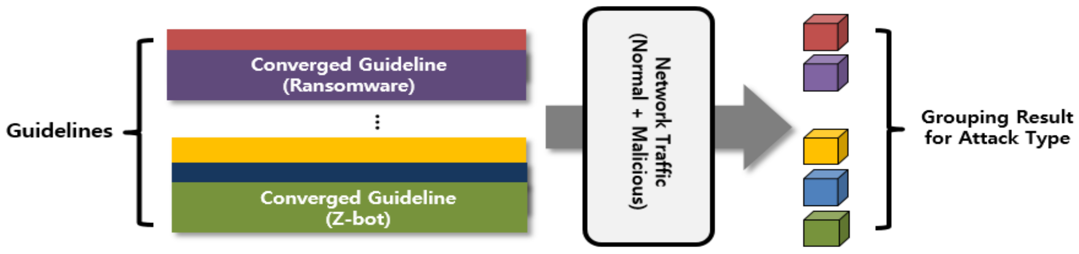

Guidelines
Converged Guideline (Ransomware)
⋮
Converged Guideline (Z-bot)
Network Traffic (Normal + Malicious)
Grouping Result for Attack Type

Fig. 6. An Example of Applying the Multiple Guidelines

Multiple seeds are applicable to one target network traffic. In **Fig. 7**, when each of the four seeds is applied to the target network traffic, the coverage is observed to be 40%. However, if all four seeds are used, the coverage increases to 88%. Hence, by using multiple seeds, the overall detection rate can be improved. But the prerequisite for this is that all seeds must be malicious flows.

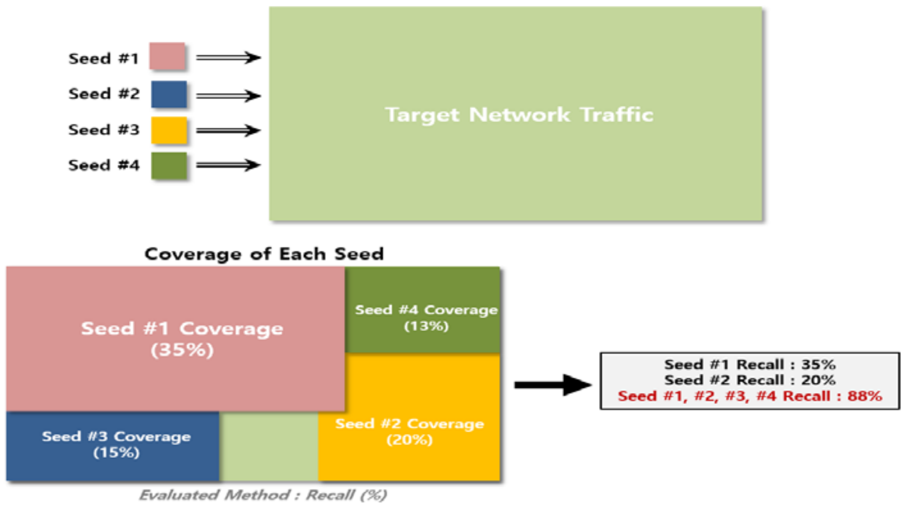


Fig. 7. An Example of Applying the Multiple Seeds

If there is a seed for a normal flow within multiple seeds, the normal flow associated to other normal flows will be falsely detected as a malicious one. In that case, the precision will be drastically reduced. Therefore, multiple seeds should be used additionally when detection performance (recall) is low with a single seed.

4. Evaluation

4.1 Experiment Environment

In this research, we have several experiments to verify our proposed method. The experiments were performed on a desktop computer which has a configuration of an Intel Core™ i7-4770K CPU @ 3.50GHz, 32GB memory and 64bit. In order to evaluate the proposed method, experiments were conducted using normal and malicious traffic.

We conducted several experiments using real malicious traffic to verify the proposed method. As the data for verification, there is the KDD data set, which is public data for verification. In the case of KDD'99 of the KDD data set, the attack type was classified by record, and it has been used in experiments in many papers. However, there is a problem in that the data size is large because there are duplicate records. Although the NSL-KDD data set was created to solve the shortcomings of the KDD'99 data set, there is a study result showing the detection rate for a specific malicious traffic is very low [31-33]. Also, the features that can be obtained from the NSL-KDD data set are different from the features used in the proposed method. Flow header information such as IP, port, and protocol can be used, but in the case of statistical feature such as the packet size distribution and inter arrival time, it is difficult to use in the KDD data set because we calculate the values from the first 5 packets of the flow. Therefore, we used the sample malicious traffic (packet capture file - pcap) from the web site which offers a collection of various types of malicious traffic, instead of using the KDD'99 and NSL-KDD data set [30].

Table 3. An Information of Experiment Traffic

Malicious Traffic Information				
Trace #	**Attack Type**	**Size**		
		Flow	**Packet**	**Byte**
1	*Dreambot*	30	4,254	3,643,162
2	*Ransomware*	14	5,688	5,070,109
3	*Ramnit*	232	7,321	3,702,860
4	*Z-bot*	23	1,232	1,236,869
5	*Trickbot*	39	11,680	14,663,528
6	*Qakbot*	817	74,404	52,380,938
7	*Dridex Malware*	62	3,485	2,987,300
8	*Amadey*	37	521	202,410
9	*Bokbot*	47	4,387	4,244,273
10	*Socgholish*	92	2,219	1,472,729
Normal Traffic Information				
Trace #	**Description**	**Size**		
		Flow	**Packet**	**Byte**
1	General Network Traffic (Chrome / KakaoTalk / Youtube ...)	844	53,655	49,471,332

Every binary file in pcap files has been recognized as malicious by IDS and Antivirus software [31]. We also collected normal traffic through our internal server. The collected normal traffic used ordinary application services such as Chrome and KakaoTalk.

Testing was conducted on network traffic with a mixture of malicious and normal flows. 10 types of malicious traffic were used in the experiment, and **Table 3** records the flow, packet, and byte counts corresponding to each traffic.

As aforementioned, we used recall, precision, and f-measure, to evaluate the FCI System. Recall indicates the percentage of detected malicious flows. Precision indicates the accuracy of detection. However, if there are only two measurements, it is difficult to evaluate the performance objectively. For example, if the recall is 95% but the precision is 10% for one method and the recall is 70% but the precision is 50% for another, it is difficult to judge which is better. Therefore, we use the f-measure that reflects two evaluation measurements to objectively evaluate the performance. Although there are several types of f-measures such as f1-measure and f2-measure, we used f1-measure, which assigns equal weights to the recall and precision.

In order to verify objective validity, we tried to conduct a comparative experiment on the related research method, such as signature-based analysis and deep learning-based analysis method. However, it is difficult to implement the model of each methodology, because each methodology is composed of various methods according to the applied algorithm or method, it is difficult to compare the proposed method with the performance. Therefore, we conduct several experiments on the Similarity Index-based model (SI Model), the Connectivity Index-based model (CI Model), and the Similarity, Connectivity-Index-based model (SI & CI Model). And we also compare the average performance of the proposed system with the other methods used in other papers. The same data and classification method was applied to all three models to conduct a malicious traffic detection experiment.

The three models are distinguished according to the flow correlation index used. That is, the SI model uses only the statistical characteristics of the flow, the CI model uses the header information of the flow, and the SI & CI model uses both characteristics. We also applied the multiple seed algorithm to verify the algorithm. The result of experiments are shown in **Table 5**.

As described in Section 3.5, a large amount of various malicious traffic is required to verify the performance of multiple guidelines. However, since it is difficult to obtain a large number of malicious traffic data, we performed a simple experiment using relatively easy to obtain and frequently occurring malware spam and ransomware in this paper.

A description of the traffic used in the experiment is shown in **Table 4**. The guidelines were created according to the attack type described in **Table 4**, and the normal traffic is the same as the traffic used in **Table 3**. The result of experiments for applying the multiple guideline are shown in **Table 6**.

Table 4. An Information of Experiment Traffic

Traffic for Guideline Generation				
Trace #	**Attack Type**	**Size**		
		Flow	**Packet**	**Byte**
1-1	*Malware*	62	3,485	2,987,300
1-2		51	2,905	3,028,700
1-3		70	4,028	4,828,601
2-1	*Ransomware*	14	5,688	5,070,109
2-2		20	3,792	4,701,101
2-3		18	4,921	5,207,100

4.2 Experiment Result

The results of the experiment have been recorded in **Table 5**. Recall, precision, and f-measure represent the results obtained by using only one seed. F-measure (MS) denotes the f-measure obtained by applying multiple seeds to each trace and MS indicates the number of used seeds. For example, F-measure (2) denotes the f-measure obtained by using two seeds.

The SI, CI, SI & CI models were observed to exhibit 100% precision corresponding to all traces. The similarity index-based model exhibited a detection rate of 30–50%, and the connectivity index-based model exhibited a detection rate of 70–95%. The proposed model, which used the complete similarity connectivity index, exhibited a recall of 85–98%. Among the three models, the model using similarity and connectivity indices exhibited the best result with respect to recall and precision. In particular, similarity and connectivity index-based models performed better when compared to other existing methods such as signature-based and deep learning-based methods.

Table 5. Experiment Result

Detection Experiment Result											
Input Traffic		**Measurement**	**Coverage (%)**								
			SI Model			**CI Model**			**SI & CI Model**		
Attack	***Normal***		**Flow**	**Pkt**	**Byte**	**Flow**	**Pkt**	**Byte**	**Flow**	**Pkt**	**Byte**
Dreambot	All	Recall (%)	33.33	1.39	0.28	76.67	99.62	99.93	90.00	99.86	99.98
		Precision (%)	100	100	100	100	100	100	100	100	100
		F- measure (Flow)	49.996			86.795			94.737		
		F-measure(2)	100			100			100		
Ramnit	All	Recall (%)	40.95	29.64	4.90	57.33	97.27	98.93	99.57	99.97	100
		Precision (%)	100	100	100	100	100	100	100	100	100
		F- measure (Flow)	58.106			72.879			99.785		
		F-measure(2)	100			100			100		
Ransomware	All	Recall (%)	35.71	65.47	63.71	78.57	99.89	99.99	78.57	99.89	99.99
		Precision (%)	100	100	100	100	100	100	100	100	100
		F- measure (Flow)	52.627			87.999			87.999		
		F-measure(3)	100			100			100		
Z-bot	All	Recall (%)	26.09	5.52	3.32	82.61	99.35	99.95	82.61	99.35	99.95
		Precision (%)	100	100	100	100	100	100	100	100	100
		F- measure (Flow)	41.383			90.477			90.477		
		F-measure(2)	100			100			100		
Trickbot	All	Recall (%)	33.33	0.31	0.02	89.74	99.55	99.92	64.10	87.37	90.90
		Precision (%)	100	100	100	100	100	100	100	100	100
		F- measure (Flow)	49.996			94.593			78.123		
		F-measure(3)	100			100			100		
Qakbot	All	Recall (%)	44.31	1.39	0.20	92.04	71.59	79.53	95.35	71.70	.79.50
		Precision (%)	100	100	100	100	100	100	100	100	100
		F- measure (Flow)	61.409			95.855			97.619		
		F-measure(2)	100			100			100		
Dridex Malware	All	Recall (%)	11.29	0.52	0.05	43.55	95.58	99.48	43.55	95.58	99.48
		Precision (%)	100	100	100	100	100	100	100	100	100
		F- measure (Flow)	20.289			60.676			60.676		
		F-measure(5)	98.21			100			100		
Amadey	All	Recall (%)	86.49	61.42	15.16	97.30	99.62	99.92	97.98	99.62	99.92
		Precision (%)	100	100	100	100	100	100	100	100	100
		F- measure (Flow)	92.756			98.632			98.979		
		F-measure(2)	100			100			100		
Bokbot	All	Recall (%)	36.17	7.20	1.43	48.94	82.70	81.64	51.06	98.95	99.90
		Precision (%)	100	100	100	100	100	100	100	100	100
		F- measure (Flow)	53.125			65.718			67.602		
		F-measure(3)	93.14			97.66			100		
Socgholish	All	Recall (%)	50.00	56.20	62.33	89.13	99.10	99.74	89.13	99.10	99.74
		Precision (%)	100	100	100	100	100	100	100	100	100
		F- measure (Flow)	66.667			94.253			94.253		
		F-measure(2)	100			100			100		

However, in the case of Dridex, Bokbot and Trickbot, the recall and precision exhibited were relatively low. We analyzed these traces to analyze the cause of the low performance. As a result of analysis, we conclude that the traffic flow of all three traces was similar to the normal flow. In particular, the PSD and PIT Feature values of the Similarity Index are comparable to those of the normal flow. Therefore, since the range of the FCI with the malicious flow is similar to the FCI with the normal flow, it shows relatively low performance. To address this shortcoming, we suggest applying multiple seeds to these traces.

When multiple seeds applied, the recall and precision of Dridex and Bokbot were observed to improve. Based on the experimental results, it can be confirmed that the proposed method can perform the required classification efficiently and simply without degrading performance compared to the existing methods.

Table 6 shows experimental results for multiple guideline verification. In this experiment, we used SI & CI models that performed best in the previous experiment. The recall, precision, and f-measure present the overall detection results derived when using the guidelines and seeds for two types of malicious traffic. In the case of f-measure to which multiple seeds are applied, the number of each used seed and the result are shown. For example, the f-measure (2, 2) in trace #1 is the result of using 2 malware seeds and 2 ransomware seeds. When the experiment is performed on three traces, the detection rate is about 78~93%, and when multiple seeds are applied, the results are more improved.

As previously described in Section 3.5, the multiple guideline algorithm is applied to various malicious behaviors or complex network attacks. As we mentioned earlier, in order to get a good performance for applying the multiple guidelines algorithm, a large amount of traffic is required for each attack. Therefore, as the future work, we will collect a large amount of malicious traffic and improve the performance of the multiple guideline algorithm by conducting several experiments.

Table 7 shows the comparison between the proposed system performance and the existing research. In **Table 7**, methodology represents the description of each method, and applying algorithm represents the description of the applied algorithm. The target traffic represents the experimental traffic applied in each method, and the f-measure represents the performance of each methodology.

Table 6. Experiment Result for Multiple Guideline Verification

Detection Experiment Result							
Trace	**Input Traffic**		**Model**	**Measurement**	**Coverage (%)**		
	Attack	**Normal**			**Flow**	**Pkt**	**Byte**
#1	Malware 1-1 Ransomware 2-1	All	SI & CI Model	Recall (%)	78.72	97.15	99.10
				Precision (%)	100	100	100
				F- measure (Flow)	88.09		
				F-measure (2, 2)	100		
#2	Malware 1-2 Ransomware 2-2	All	SI & CI Model	Recall (%)	93.30	99.00	99.81
				Precision (%)	100	100	100
				F- measure (Flow)	96.53		
				F-measure (1, 4)	100		
#3	Malware 1-3 Ransomware 2-3	All	SI & CI Model	Recall (%)	88.78	98.81	99.90
				Precision (%)	100	100	100
				F- measure (Flow)	94.06		
				F-measure (2, 3)	100		

In the proposed method, when using multiple seeds, the average accuracy value is.99.28 %. The performance of deep learning based analysis method varies greatly depending on the DL algorithm and dataset used. In Table 7, Yin el al. [27], Loukas et al. The method of [28] showed an average accuracy of about 83~87%, but Yu et al. [29] shows an average accuracy of 98.96%.

As we do not use public experimental data, it is difficult to verify the performance of the proposed method. Therefore we considered the experiments and results of other studies to verify the performance shown in Table 7. Various studies are being conducted in the field of malicious traffic classification and detection, and since each study uses different data according to their system input data format, there is no problem with the verification experiment procedure. Although we cannot compare the performance better than other researches, but when considering at the detection experiment results, the average detection accuracy of 99.28% is considered to be high performance.

Table 7. The Result of Proposed Method and Other Methods

	Proposed Method	**Yin et al. [27]**	**Loukas et al. [28]**	**Yu et al. [29]**
Methodology	Flow Correlation Index	Deep Learning	Deep Learning	Deep Learning
Algorithm	Multiple Seed	RNN	LSTM	CNN, RNN
Dataset-Used	Various Malicious Traffic	NSL-KDD	Various Malicious Traffic	Alexa, OSINT
Average Accuracy (%)	99.28 %	83.28 %	86.9 %`	98.96 %

5. Conclusion

In this paper, we have investigated the problems that plague existing malicious traffic detection methods and propose a method to solve it. We discussed the necessity of classifying malicious traffic and the problems with the existing research methods, such as payload signature-based and deep learning-based methods in the Introduction and Related Works sections. To address these problems, we proposed a malicious traffic detection method based on statistical and header information of the flow. We explained the structure of the method, the basic concepts like the flow correlation index, and the detailed algorithm in Section 3.

In order to evaluate the proposed method, we conducted several experiments using 10 types of real malicious traffic and collected normal traffic. During the evaluation, we used three models for comparing the performances. Experimental results demonstrated that the performance of the SI & CI Model yielded the best results with respect to recall and precision. Corresponding to most traces, the recall was over 90–98% and precision was 100%. In particular, by applying multiple seeds, it was possible to increase the recall to 100% for all flows in all traces. The performance was verified to be 100% precision using the threshold setting algorithm and 100% recall by applying multiple seeds. We also compare the performance of the proposed method with other existing research methods. In proposed method, when we apply the multiple seed algorithm, an average of 99.28% accuracy was obtained, which shows higher performance than other methods. We also conducted the experiment to verify the multiple guideline algorithm. We conducted an experiment on two types of malicious traffic: malware and ransomware. Even when two or more types of malicious traffic are generated through an experiment, good detection performance can be achieved by applying multiple guidelines. Since it is difficult to obtain a large amount of malicious traffic, we performed the experiment on only two types of malicious traffic. Therefore, in future research, we will apply the multiple guidelines to various malicious traffic and study how to apply it effectively.

In addition, the proposed method can solve some of the limitations of the learning-based analysis method mentioned in Section 2. The learning-based analysis method has several problems in that it is highly dependent on the data set and it is difficult to select an appropriate feature. Since the proposed method generates the guidelines with an algorithm similar to the learning-based analysis method, the dependence on the data set still exists. However, the feature used in the proposed method utilizes the flow header and statistical information defined through various malicious and normal traffic analysis. Therefore, the proposed method does not need to select an appropriate feature according to the applied model and data.

However, there are some limitations in the proposed system. First, it can take a long time to generate guidelines during the Training Section. Certain types of malicious traffic are capable of generating large amounts of flow depending on attack patterns and characteristics. In this case, the process of generating a guideline takes a considerable time because of the correlation index calculation between seed flows and other flows. For example, Qakbot has more than 500 seeds, and the process took significantly longer in its case than for other traces. When using the experimental data in Section 4, the guideline was generated on average within 15 to 30 minutes in Training Section (defined weight value = 0.01). However, if we adjust the weight values in detail, it will take longer than 30 minutes. Due to the algorithmic nature of the proposed method, the more detailed the features and weights are adjusted, the longer it takes to generate the guideline. However, if the proposed system is applied in Testing Section, it will take less time because only the detection process is performed using pre-generated guidelines. For the experimental data used in Section 4, the detection process was performed within 1 minute. Second, for some traces, a low detection rate appears when multiple seeds are not applied. If multiple seeds cannot be applied, performance may be poorer than other methodologies. We believe that this phenomenon is caused by less sophisticated threshold setting algorithm during the creation of guidelines. Therefore, it is necessary to improve the threshold setting algorithm to generate more sophisticated guidelines.

Nevertheless, the proposed method addresses several existing problems and exhibits good performance. In particular, it uses statistical and header information of the flow, which is simpler than the methodology of other processes. In the proposed system, if the detailed malicious traffic information is received from an external firewall as a seed, the detection rate and precision are expected to be better than current result.

In this paper, experiments were conducted using only three models. However, various models can be created by adjusting the grouping order, method. If various models are used and guidelines according to the models are made in advance, even if a new type of malicious traffic occurs, it can be analyzed efficiently.

As we mentioned previously, the generation of the guideline can take a considerable amount of time. Therefore, we will improve the threshold setting and optimization algorithm for the future work to decrease guideline generation time and improve the performance of the system. In addition, we will collect more malicious and normal traffic to generate more sophisticated guidelines through various experiments to improve detection performance.

References

[1] M. S. Kim, Y. J. Won, and J. W. K. Hong, “Application-Level Traffic Monitoring and an Analysis on IP Networks,” *ETRI Journal*, Vol. 27, pp. 22-42, 2015. Article (CrossRefLink)

[2] K. S. Shim, S.H. Yoon, S.K. Lee, S.M. Kim, W.S. Jung, M.S. Kim, “Automatic Generation of Snort Content Rule for Network Traffic Analysis,,” *KICS*, Vol.40, No.04, pp.666-677, April, 2015. Article (CrossRefLink)

[3] A. Callado, C. Kamienski, G. Szabo, B. Gero, J. Kelner, S. Fernandes, et al., “A Survey on Internet Traffic Identification,” *IEEE Communications Surveys and Tutorials*, Vol. 11, pp. 37-52, 2009. Article (CrossRefLink)

[4] A. Dainotti, A. Pescape and K. Claffy, “Issues and Future Directions in Traffic Classification,” *Network IEEE*, Vol. 26, no. 1, pp. 35-40, 2012. Article (CrossRefLink)

[5] B. C. Park, Y. J. Won, M.-S. Kim, and J. W. Hong, “Towards Automated Application Signature Generation for Traffic Identification,” in *Proc. of Network Operations and Management Symposium, NOMS 2008*, IEEE, pp. 160-167, 2008. Article (CrossRefLink)

[6] K. C. Lan and J. Heidemann, “A Measurement Study of Correlations of Internet Flow Characteristics,” *Computer Networks*, Vol. 50, pp. 46-62, 2006. Article (CrossRefLink)

[7] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, “A Survey of Intrusion Detection Techniques in Cloud,” *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 42–57, 2013. Article (CrossRefLink)

[8] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, “An Overview of IP Flow-Based Intrusion Detection,” *IEEE Commun. Surveys Tutorials*, Vol. 12, no. 3, pp. 343–356, quarter 2010. Article (CrossRefLink)

[9] R. Perdisci, W. Lee, and N. Feamster, “Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces,” *NSDI*, Vol. 10, 2010.

[10] H. M. An, S. K. Lee, J. H. Ham, and M. S. Kim, “Traffic Identification based on Applications using Statistical Signature free from Abnormal TCP Behavior,” *JOURNAL OF INFORMATION SCIENCE AND ENGINEERING*, Vol.31, no.5, pp.1669-1692, Sep. 2015. Article (CrossRefLink)

[11] J. S. Park, S. H. Yoon and M. S. Kim, “Performance Improvement of the Payload Signature based Traffic Classification System using Application Traffic Temporal Locality,” *The Journal of Korean Institute of Communications and Information Sciences*, vol. 38B, pp. 519-525, 2013. Article (CrossRefLink)

[12] Y. J. Won, S. C. Hong, B. C. Park, and J. W. K. Hong, "Automated Application Signature Generation for Traffic Identification,” *POSTECH*, Korea, Aug. 16, 2008.

[13] S. H. Yoon, J. S. Park, and M. S. Kim, “Behavior Signature for Fine-grained Traffic Identification,” *Applied Mathematics & Information Sciences*, Vol. 9, No. 2L, pp. 523-534, Apr. 2015.

[14] X. Feng, X. Huang, X. Tian, and Y. Ma, “Automatic Traffic Signature Extraction based on Smith-Waterman Algorithm for Traffic Classification,” in *Proc. of Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on*, pp. 154-158, 2010. Article (CrossRefLink)

[15] M. Finsterbusch, C. Richter, E. Rocha, J. A. Muller and K. Hanssgen, “A Survey of Payload-Based Traffic Classification Approaches,” *Communications Surveys & Tutorials IEEE*, Vol. 16, no. 2, pp. 1135- 1156, 2014. Article (CrossRefLink)

[16] F. Risso, M. Baldi, O. Morandi, A. Baldini, and P. Monclus, “Lightweight, Payload-Based Traffic Classification an Experimental Evaluation,” in *Proc. of IEEE International Conference on Communications*, Beijing, China, pp. 5869-5875, May. 19-23, 2008. Article (CrossRefLink)

[17] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, “Machine Learning for Networking: WorkFlow, Advances and Opportunities,” *IEEE Network*, Vol. 32, no. 2, pp. 92–99, Mar./Apr. 2018. Article (CrossRefLink)

[18] T. T. T. Nguyen and G. Armitage, "A Survey of Techniques for Internet Traffic Classification using Machine Learning," *IEEE Communications Surveys and Tutorials*, Vol. 10, pp. 56-76, 2008. Article (CrossRefLink)

[19] S. Pouyanfar et al., "A Survey on Deep Learning: Algorithms, Techniques, and Applications," *ACM Comput. Surveys*, Vol. 51, no. 5, pp. 1–36, 2018. Article (CrossRefLink)

[20] Y. Dhote, S. Agrawal, "A Survey on Feature Selection Techniques for Internet Traffic Classification," in *Proc. of 2015 International Conference on Computational Intelligence and Communication Networks*, Jabalpur, pp. 1375-1380, 2015. Article (CrossRefLink)

[21] Z. B. Celik, R. J. Walls, P. McDaniel and A. Swami, "Malware Traffic Detection using Tamper Resistant Features," in *Proc. of Military Communications Conference, MILCOM 2015 - 2015 IEEE*, Tampa, FL, pp. 330- 335, 2015. Article (CrossRefLink)

[22] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Communications Surveys Tutorials*, Vol. 18, no. 2, pp. 1153–1176, Secondquater 2016. Article (CrossRefLink)

[23] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware Traffic Classification using Convolutional Neural Network for Representation Learning," in *Proc. of 2017 International Conference on Information Networking (ICOIN)*, IEEE, Jan, pp. 712–717, 2017. Article (CrossRefLink)

[24] R. K. Sharma, H. K. Kalita, and P. Borah, "Analysis of Machine Learning Techniques based Intrusion Detection Systems," in *Proc. of 3rd International Conference on Advanced. Computing, Network Informatics*, pp. 485–493, 2015. Article (CrossRefLink)

[25] M. J. De Lucia, and C. Cotton, "Detection of Encrypted Malicious Network Traffic using Machine Learning," in *Proc. of MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)*, IEEE, pp. 1-6, 2019. Article (CrossRefLink)

[26] D, Tirtharaj, "A Study on Intrusion Detection using Neural Networks Trained with Evolutionary Algorithms," *Soft Computing*, 21(10), pp. 2687-2700, 2017. Article (CrossRefLink)

[27] C. Yin, Y. Zhu, J. Fei, and X. He, "A Deep Learning Approach for Intrusion Detection using Recurrent Neural Networks," *IEEE Access*, 5, pp. 21954–21961, 2017. Article (CrossRefLink)

[28] G. Loukas, T. Vuong, R. Heartfield, G. Sakellari, Y. Yoon, and D. Gan, "Cloud-based cyber-physical intrusion detection for vehicles using Deep Learning," *IEEE Access*, 6, pp. 3491–3508, 2018. Article (CrossRefLink)

[29] B. Yu, D. L. Gray, J. Pan, M. D. Cock and A. C. A. Nascimento, "Inline DGA Detection with Deep Networks," in *Proc. of 2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, New Orleans, LA, pp. 683-692, 2017. Article (CrossRefLink)

[30] Malware traffic analysis.net. https://www.malware-traffic-analysis.net.

[31] I. Letteri, G. D. Penna, L. D. Vita, and M. T. Grifa, "MTA-KDD'19: A Dataset for Malware Traffic Detection," *ITASEC*, 2020. Article (CrossRefLink)

[32] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. of 2009 IEEE symposium on computational intelligence for security and defense applications*, IEEE, 2009. Article (CrossRefLink)

[33] L. Dhanabal, and S. P. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, Vol. 4, no. 6, pp. 446-452, 2015.

Jee-Tae Park was born in Busan, South Korea, in 1993. He received his B.S. degree computer and information science from Korea University, South Korea, in 2017, where he is currently pursuing the Ph.D. degree (integrated program). His research interests include Internet traffic classification, network management and Internet security.

Ui-Jun Back was born in Seoul, South Korea, in 1993. He received his B.S. degree computer and information science from Korea University, South Korea, in 2018, where he is currently pursuing the Ph.D. degree (integrated program). His research interests include blockchain transaction monitoring, network management and Internet security.

Min-Seong Lee was born in Iksan, South Korea, in 1994. He received his B.S. degree computer and information science from Korea University, South Korea, in 2020, where he is currently pursuing the Master degree. His research interests include network management and Internet security

Young-Hoon Goo was born in Cheonan, South Korea, in 1991. He received the B.S. and Ph.D. degrees (integrated program) in computer and information science from Korea University, South Korea, in 2016 and 2020 respectively. Since 2020, he has been a postdoctoral researcher with Korea Institute of Science and Technology Information (KISTI), South Korea. His research interests include Internet traffic classification, Internet security, network management, and wireless communication.

Song-Ho Lee was born in Seoul, South Korea, in 1991. He received the B.S. and M.S. degrees in computer and information science from Korea University, South Korea, in 2016 and 2018 respectively. Since 2018, he has been a researcher with AhnLab, South Korea. His research interests include Internet traffic classification, Internet security, network management, and wireless communication.

Myung-Sup Kim was born in Gyeongju, South Korea, in 1972. He received his B.S., M.S., and Ph.D. degrees in computer science and engineering from POSTECH, South Korea, in 1998, 2000, and 2004, respectively. From September 2004 to August 2006, he was a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of Toronto, Canada. He joined Korea University, Korea, in 2006, where he is working currently as a Full Professor with the Department of Computer Convergence Software. His research interests include Internet traffic monitoring and analysis, service and