JISTaP http://www.jistap.org
Journal of Information Science Theory and Practice

RESEARCH PAPER
J Inf Sci Theory Pract 10(Special): 143-153, 2022

eISSN : 2287-4577 pISSN : 2287-9099

https://doi.org/10.1633/JISTaP.2022.10.S.14

A Network Packet Analysis Method to Discover Malicious Activities

Taewoong Kwon
Korea Institute of Science and Technology Information (KISTI), Daejeon, Korea
E-mail: taewoong.kwon@kisti.re.kr

Jun Lee
Korea Institute of Science and Technology Information (KISTI), Daejeon, Korea
E-mail: jun.lee@kisti.re.kr

Jungsuk Song*
Korea Institute of Science and Technology Information (KISTI), Daejeon, Korea
E-mail: song@kisti.re.kr

Joonwoo Myung
Korea Institute of Science and Technology Information (KISTI), Daejeon, Korea
E-mail: joonwoo.myung@kisti.re.kr

Kyu-il Kim
Korea Institute of Science and Technology Information (KISTI), Daejeon, Korea
E-mail: kisados@kisti.re.kr

ABSTRACT

With the development of networks and the increase in the number of network devices, the number of cyber attacks targeting them is also increasing. Since these cyber-attacks aim to steal important information and destroy systems, it is necessary to minimize social and economic damage through early detection and rapid response. Many studies using machine learning (ML) and artificial intelligence (AI) have been conducted, among which payload learning is one of the most intuitive and effective methods to detect malicious behavior. In this study, we propose a preprocessing method to maximize the performance of the model when learning the payload in term units. The proposed method constructs a high-quality learning data set by eliminating unnecessary noise (stopwords) and preserving important features in consideration of the machine language and natural language characteristics of the packet payload. Our method consists of three steps: Preserving significant special characters, Generating a stopword list, and Class label refinement. By processing packets of various and complex structures based on these three processes, it is possible to make high-quality training data that can be helpful to build high-performance ML/AI models for security monitoring. We prove the effectiveness of the proposed method by comparing the performance of the AI model to which the proposed method is applied and not. Forthermore, by evaluating the performance of the AI model applied proposed method in the real-world Security Operating Center (SOC) environment with live network traffic, we demonstrate the applicability of the our method to the real environment.

Keywords: security monitoring, data preprocessing, machine learning, artificial intelligence, natural language processing

Received: April 27, 2022 **Revised:** May 5, 2022
Accepted: May 17, 2022 **Published:** June 20, 2022

***Corresponding Author:** Jungsuk Song
https://orcid.org/0000-0003-1755-8636
E-mail: song@kisti.re.kr

1. INTRODUCTION

With the development of network communications technologies and the increasing number of devices connected to the networks, the number of cyber threats targeting them has also increased. Not only have the number of cyber threats increased, but they are also sophisticated and gradually accelerating in quantity using Machine Learning (ML)/Artificial Intelligence (AI) and automation tools. In line with the growth in cyber threats, network security technologies naturally have been studied to protect information assets from various cyber threats. Various security devices, such as Intrusion Detection System (IDS), Intrusion Prevention System (IPS), Unified Threat Management (UTM), Enterprise Security Management (ESM), and Security Information Event Management (SIEM) are established and operated for a multi-stage defense system. In recent years, moreover, advanced technologies using ML and AI are especially being studied, not only to detect cyber-attacks but also to automate the classification of security events from various network security device. However, most of these technologies cannot be used in real environments because there are many cases where AI models verified with high performance in previous research show low accuracy in real environments. There are two main reasons for this. One is that most existing studies do not consider real environments. They mostly use open data sets, which are usually simulation data or outdated data sets that do not fit the current protocol and environments. Therefore, such studies show high performance in the given experimental data, but inevitably show low performance when testing data that is not used for learning in a real environment. The other issue is uniformly preprocessing methods that do not consider the characteristics of the data. Network packets have characteristics of both natural language (unstructured data) and machine language (structured data). Some of them have only natural language (video, music, letters), others have only machine language (IoT device signals, formatting files), and others have both. In particular, in the case of web-related packets, since there are various types including source code, file path, personal information, and infection signals related to malicious code, appropriate preprocessing methods that understand their characteristics are required, but most existing studies do not consider this.

In this research, we focused on the absence of a preprocessing method. To make an AI model for real-world monitoring, this study proposes a novel data preprocessing method consisting of three steps. The three steps consist of (1) Preserving significant special characters, (2) Generating a stopword list, and (3) Class label refinement. Firstly, by defining significant special characters to be preserved, we can prevent the semantic change that occurs during preprocessing and preserve the performance of the ML/AI model. A detailed description of each step is provided in Section 3. Generating and adopting stopwords is also an important task in the packet preprocessing since the features not related to the label are also major impediments to model training. We defined appropriate stopwords in consideration of the characteristics of natural language and machine language. Furthermore, we adopted them to the packet. Finally, Class label refinement is the process of correcting incorrect labels. In Security Operating Center (SOC), because the security monitoring agent makes a final decision to classify benign/malicious packets, the agents can make a wrong decision by analytical ability, condition, timing, and even mistakes. This misjudgment is a major reason of low performance when training an AI model by labeling the same data differently, so it is necessary to correct it. After conducting refinement, we also define stopwords. We summarize the main contributions of this paper as follows:

- Data consistency: It is possible to ensure a consistent data set through label refinements so that the process can correct incorrect labels caused by human misjudgment.
- High usability in real-world: The proposed method makes it possible to produce installable and operable AI models with high performance in the real world by making high-quality training data.
- Generalization: It can be used in combination with other methods that utilize the payload of the packet.

The performance of the models applied proposed method is evaluated in terms of accuracy, especially under the real-world SOC environment with live network traffic. The rest of the paper is organized as follows. Section 2 provides related work. Section 3 describes our packet preprocessing method for ML/AI, and Section 4 discusses the evaluation effectiveness of the proposed method. Finally, we summarize our results and conclusions in Section 5.

2. RELATED WORKS

2.1. Intrusion Detection with ML/AI Techniques

To protect information assets from cyber threats, net-

work intrusion detection has been studied for a long time. There are many network security systems (i.e., NIDS/NIPS, UTM, ESM, SIEM, firewall [FW], and web application firewall [WAF]) that are developed and installed at the entrance of an internal network. They detect and alert to known cyber threats based on detection rulesets (signature or pattern) (Cisco, 2018; McAfee, 2020; Open Information Security Foundation, 2020). Network security analysts have to make detection rules rapidly and properly, because without rules, even if it is a known attack, it cannot be detected, and the damage from them can spread. However, as cyber threats are increasingly sophisticated, making the detection ruleset rapidly is a challenging task for network security analysts. To overcome the inherent limitation of a ruleset-based system, ML/AI techniques have recently been utilized to enhance known attack detection as well as detect unknown threats without a ruleset. Well established ML/AI models outperformed conventional ruleset-based systems in terms of performance (e.g., accuracy, precision, recall, and F1-score) (Almseidin et al., 2017; Sommer & Paxson, 2010). Besides this, a model can be also utilized for anomaly detection, which learns benign states and detects behaviors that deviate from it. The study by Li et al. (2016) introduced MVPSys, developed by them. It is an efficient system to reduce false alarms based on a semi-supervised approach. Recent research by Medjek et al. (2021) even considers fault-tolerant IDS using AI algorithms to protect vulnerable IoT environments against cyber threats. Furthermore, an interpretable classification method (e.g., XAI) (Wang et al., 2020) has been studied for understanding and interpreting how the trained ML/AI model classifies various traffic as benign or abnormal.

2.1.1. Packet Preprocessing

Preprocessing means manipulation of data, such as dropping, filtering, transformation, normalization, integration, cleaning, and handling of missing values, and so on, before it is used to ensure or enhance performance, and this is an important step in the data process. In network packet analysis, since the packet structure has various types, the process should be properly chosen according to the purpose. Hence many existing studies related to packet analysis have used a combination of preprocessing methods from simple encoding (e.g., base 64, UTF8) to word embedding (e.g., Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and FastText (Bojanowski et al., 2017)) to get the results they want (Lin & Chen, 2021). As well, because the packets are used to freely deliver messages according to predefined protocols and machine language, they have the characteristics of natural language and machine language. Thus we can apply natural language process (NLP) or text mining techniques for packet preprocessing, and they have already been applied to packet analysis since the early 2000s. Word2Vec is a representative NLP technique. It learns word term relation in a large text corpus and can be applied to various fields. It can be used in packet processing for security monitoring. It is used to learn the relationship between a word's composition and order in true-positive (TP)/false-positive (FP) packets to distinguish them. There are various studies applying it, such as Packet2Vec (Goodman et al., 2020) and TLS2Vec (Ferriyan et al., 2022), which have been studied and utilized in security monitoring. Term-Frequency-Inverse Document Frequency (TF-IDF) is also a technique often used in NLP.

In packet processing, a packet is treated as a document, and a word (or a sequence of bytes or a term) extracted from the packet through a tokenizer or chunking algorithms is treated as a word. The study by Yang et al. (2021) proposed a distinguish method combining TF-IDF and ML such as CNN and AdaBoost for classifying malicious encryption traffic and normal traffic. Apart from the Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), the Co-clustering algorithm has also been applied as a data dimensionality reduction technique by clustering records and fields (Banerjee et al., 2005; Madeira & Oliveira, 2004). These techniques are also used for preprocessing of anomaly-based intrusion detection. Research by Kakavand et al. (2016) introduced TMAD (Text Mining-based Anomaly Detection) using *n-gram* text categorization (TC) and TF-IDF. They make a feature through the *n-gram* TC and TF-IDF and show high detection rates through it. In this way, most existing studies use the NLP or text mining techniques for feature extraction from simulation datasets. Yet most of these models performed with a high performance in their test environments but performed with low performance in the real environment where various types of packets exist, because a batch of pre-processing methods are applied without considering the characteristics of various types and types of packets. In addition, since they use a natural language-based dictionary and not a security-specialized one, they are not also suitable for packet preprocessing. In recent years, many studies using end-to-end learning that does not consider such preprocessing have been made, but they are based only on some simulation data, not real packets captured in the real world (Chen et al., 2017; Le et

al., 2020).

3. PROPOSED METHOD

Data processing is one of the most important tasks to build high performance ML/AI works. However, the data from security devices, especially IDS/IPS events, usually contain various types of payloads such as predefined machine language, natural language, and even both. Therefore, the process is more burdensome and complex than other structured data such as CSV or JSON format, and the learning effect of ML can be maximized only by using a data processing method that considers these characteristics. Moreover, label inconsistency, which is a critical attribute for ML/AI tasks, easily occurs because there are many unexpected variables, including a wide range of career (i.e., ability) of human agents, different decision standards and timings, complex SOC situations, and so on. To decrease these negative biases from the security data, we proposed the following three steps as in Fig. 1: preserving significant special characters, generating a stopword list, and class label refinement. More details are detailed in the section below.

3.1. Preserving Significant Special Characters

Different from natural language, some special characters have important meaning (e.g., file indicator, data separator, parameter, or some evidence of threats) in payload. They can describe relationships between two different terms and are used as a delimiter for a specifically formatted string. That is, if these characters are deleted according to the same process as NLP, the original meaning of the payload will be faded or lost; consequently, it has effect on the low performance of the AI model. As a result, special characters with important meanings should be preserved and suitable methods for dealing with them must be considered. We define the significant special characters to be preserved which are related to cyber threats or have important information, and replace with white spaces all other special characters except for them. Table 1 shows ten significant characters with usage examples in payload. For instance, a colon (:) is generally used as a delimiter that distinguishes keys and values in the key-value structure, so that without it, the key and the value are difficult to distinguish; however, it is easily eliminated in text mining. Moreover, at (@) is also a typical special character used in email addresses. It can give us some information of email related to phishing, malicious file distribution, and other abuse behaviors; thus it needs to be kept with an original form.

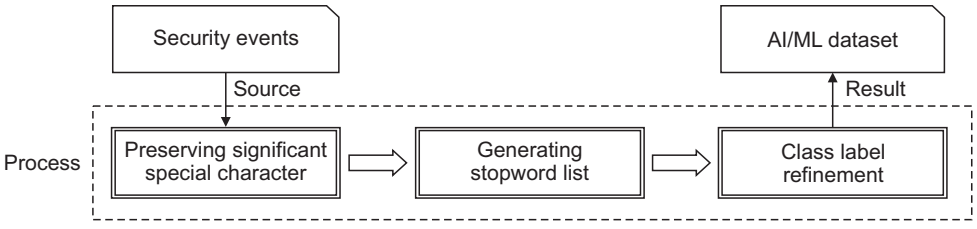

Fig. 1. Whole process of this research. AI, Artificial Intelligence; ML, Machine Learning.

Table 1. Pre-defined significant special characters and usages

Character	Description (Unicode)	Usage in payload
@	At sign (U+0040)	@hotmail.com, @gmail.com
\	Backslash (U+20A9)	C:\User\passwd
-	Hyphen (U+2010)	Down.dll-biu, x-installer.exe, or 1=1 –
:	Colon (U+003A)	{id:xxx}{passwd:xxx}
%	Percent (U+0025)	%20%23
_	Underbar (U+005F)	_0X, /system_get/
.	Period (U+002E)	Show.asp, shime.exe
/	Solidus (U+002F)	/etc/passwd
?	Question Mark (U+003F)	Index.php?option=com
$	Dollar Sign (U+0024)	cmd=$var

 https://doi.org/10.1633/JISTaP.2022.10.S.14

These significant characters are used when defining stopwords in Section 3.2. Terms with predefined special characters, as well as words containing special characters, should not be deleted in order to preserve their meaning. More details about this are explained in Section 3.2.2.

3.2. Generating Stopword List

The purpose of generating stopwords in terms of security classification is to both eliminate noise and preserve core components to understand the meaning of the payload. It is a very important process because the method helps to maximize the effectiveness of learning when learning the meaning and relationship of terms, and it is a method often used in NLP. However, unlike the case of general NLP, the payload is a mixture of machine language and natural language, and its form and type are very complex, so a domain-specific processing method is required. In this section, we propose the novel method of generating a stopword list considering the characteristic of the packet. The proposed method introduced in this section consists of two parts as shown in Fig. 2: generation and adoption. In the generation part, stopwords are defined by calculating the weights of words and removing statistically significant words. The adoption part describes how to apply the stopwords generated in the generation part. We describe the detail of parts in the section below, respectively.

3.2.1. Term Weight Calculation

To eliminate meaningless terms, we should quantify the importance of each term included in payloads. We can realize the term's importance and select stopwords through the quantified value. TF-IDF is a well-known method to calculate the weights of the terms. It is a traditional but very efficient method. We also use modified TF-IDF to calculate the weight of the terms within each event. Here is a detailed modified TF-IDF as follows:

$$TF(t,d) = \frac{f_{t,d}}{|d|} \tag{1}$$

where $f_{t,d}$ is the raw count of a term as word t in a payload of security event (i.e., document $|d|$) and $|d|$ is the length of document (i.e., all the number of terms in the document). We should consider that all payloads have different lengths, and the weights of terms should be calculated for all payloads. So, by dividing the frequency of terms in the document into the length of the document,

$$IDF(t,D) = \frac{|D|}{1 + |\{d \in D : t \in d\}|} \tag{2}$$

where $|D|$ is total number of documents D, $|\{d \in D: \mathrm{t} \in d\}|$ is the number of payload where the term t appears, and the denominator is adjusted with 1 to prevent division-by-zero. Also, log is applied to prevent bias that occurs when the number of documents is large. Finally, TF-IDF is calculated by multiplying TF by IDF values as follows:

$$TF \cdot IDF(t,d,D) = TF(t,d) \cdot IDF(t,D) \tag{3}$$

After the calculation, every term in each payload has an importance score. And then, by adding all the TF-IDF values for each term, we can get the weights of the term t in all payloads calculated such as in Equation 4. Fig. 3 shows all processes in this section as a table.

$$weight(t,D) = \sum_{n=1}^{|D|} TF \cdot IDF(t,d_n), (d_n \in D) \tag{4}$$

3.2.2. Security-biased Term Elimination

After finishing term weight calculation, stopwords are determined according to threshold k. Following the previous step, the large weight of the term means that the term has a large effect on the payload. That is, if the weight of a term is greater than or equal to k, it can be classified as a meaningful term, and otherwise it is classified as a stopword. Equation 5 shows this process, where k can be determined by the experience of the human agent or any other standard (e.g., Zipf's law).

$$Term \begin{cases} meaningful\ term, & if\ weight(t,D) \geq k \\ candidate\ term & otherwise, \end{cases} \tag{5}$$

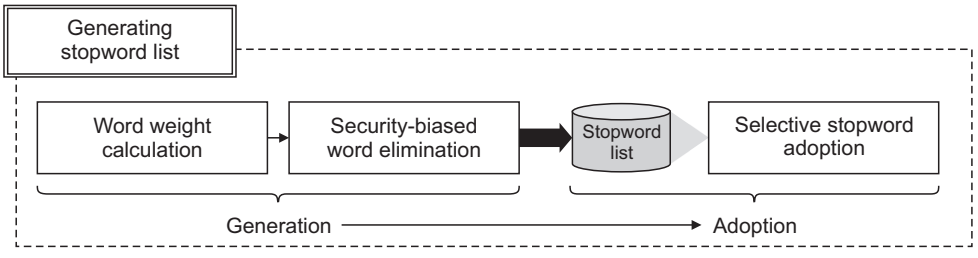

Fig. 2. Process of generating stopword list.

Fig. 3. Examples of TF-IDF matrices. TF-IDF, Term-Frequency-Inverse Document Frequency.

Subsequently, we also remove meaningful terms, because our purpose is to define the meaningless terms. A meaningful term means that it can relate to labels or be helpful to distinguish the payload. Consequently, it cannot be a stopword. For eliminating these meaningful terms, we classified the entire payloads into two groups by labels called set B (benign) and set M (malicious). To extract terms from each classified group, our own tokenizer is used. The tokenizer separates terms based on spaces and special characters, except for significant characters defined in Section 3.1. Then, by comparing the extracted terms from each group, we eliminate terms that only appeared in one of the two groups. Equation 5 shows the process of extracting candidate terms, where T_{fp}, T_{fn} is the set of a word term (t) in a payload (i.e., document d in M or B). M and B are sets of benign and malicious payloads, respectively.

$$stopwords = candidate\ term - T_{fp} - T_{fn} \begin{cases} T_{fn} = \{t | t \in d : d \in M\} \\ T_{fp} = \{t | t \in d : d \in B\} \end{cases} \quad (6)$$

3.2.3. Selective Stopword Adoption

As mentioned in the previous section, the characteristic of natural language and machine language should also be considered when applying stopwords in packet preprocessing. Instead of simply removing them, like NLP, they should be replaced or eliminated depending on the type of terms. Stopwords generated according to Section 3.2.1 and 3.2.2 can be divided into two types: terms including significant characters or consisting of significant characters, called a mixed type, and terms without any other special characters, called a pure type. These two types are applied in different ways. In case of the pure type, string match and replace algorithm is used. If the given term from the payload is a pure type, the stopword can be replaced with white space through 100% string matching. For example, if we have a stopword 'pen' and two given terms 'pen' and 'pencil' extracted from the payload, then, since the given term 'pen' is matched to stopwords 'pen,' the given term 'pen' is eliminated from the payload. But 'pencil' has the extra string 'cil' against stopword 'pen'; so, it remains. In case of a mixed type, substring matching and replace algorithm is used. If the given term is a mixed type, then we use substring match matching to find terms that contain stopwords. For example, let us suppose '@gmail.com' is the stopword and term 'good@gmail.com' is extracted from the payload. Then, we use the substring match and check whether stopwords are included in the given term, because 'good@gmail.com' has significant special character '@.' In this case, since '@gmail.com' is a substring of 'good@gmail.com,' 'good@gmail.com' is eliminated. In natural language, there are cases where the meaning is changed due to the difference of only one character (e.g., book,

cook, hook, and so on), whereas in ML, similar formats often have similar meanings. The '@gmail.com' aforementioned is an e-mail address format, so 'good@gmail.com' and 'nice@gmail.com' are different terms, but the fact that they are e-mail addresses is the same, and they have the same special characters and format. In this way, removing words that have the same pattern as stopwords helps AI-model training. Table 2 is pseudo code about stopword adoption.

3.3. Class Label Refinement

The purpose of the class label refinement is to correct mislabeled data by various situations in SOC. In data such as text or images, labels are clearly distinguished, whereas payload analysis in monitoring may have different labels depending on the analyst or situation. Hence, it is an important thing to minimize mislabeling so that the characteristics of each label can be revealed clearly and the intrusion detection model can be trained well. In SOC, mislabeling is inevitably caused by changes in the decision basis over time, differences in the monitoring agent's analysis ability, and even simple mistakes. Even where there is the same payload of security events or raw traffic, different analysts and timing make it possible to label differently. To solve these problems, label refinement, which is data-driven labeling, is used in this paper based on similarity measures. Similarity analysis is performed based on payloads previously clearly labeled as malicious by the security monitoring agent, and payloads with similarity above a threshold are classified as true-positive and false-positive otherwise. To compare a similarity between verified payload and other payloads, firstly all terms in the payload are scored by the TF-IDF method used in Section 3.2.2. But, unlike in Section 3.2.2, we divide the frequency of the term into the mode value of terms, not total number of terms, in a payload in this section, because the similarity comparison between two payloads do not need to consider the other payloads. That is, TF is as follows:

$$TF(t,d) = \frac{f_{t,d}}{\max\{f_{t',d}: t' \in d\}} \quad (7)$$

Each payload is considered as each document and each term extracted from the payload is considered as each term. After the calculation, all terms in the payload have the importance score, and TF-IDF matrices for all payload are formulated. Using the values on the matrix as shown in Fig. 3, the similarity between payload, especially malicious events vs. unlabeled events, is compared based on similarity measures such as cosine similarity, Minkowski distance, and Mahalanobis distance.

The label of the unrefined event is lastly determined based on the similarity value with a threshold (σ) which is examined by know-how from the agent experience as follows:

$$ue_l = \begin{cases} 1(maliciou), & if\ S_{m,c} > \sigma, \\ 0(benign), & otherwise \end{cases} \quad (8)$$

where *el* is the determined label for an unrefined event e, m is a malicious event, $S_{m,c}$ is a similarity value (e.g., cosine similarity=$\frac{m \cdot e}{\|m\|\|e\|}$), and σ is a threshold defined by the human agent. In other words, all events will be classified as malicious if the similarity with the malicious events is over σ; otherwise, it will be grouped to benign.

Table 2. Pseudo-code of stopword adoption

```
Input: Significant Special Characters C, Stopwords S, Wordset
       segmented from the payload W_i
Output: Wordset W_o removed stopwords

W_o ← Φ
for word in W_i
        if word contained c ∈ C then
                i = 0
                for s in S
                        if IsSubstring(word, s) then
                                break
                        i = i+1
                end for
                if i == len(S)
                        W_o ← W_o ∪ word
        else
                if word ∉ S then
                        W_o ← W_o ∪ word
                else
                        continue
                end if
        end if
end for
Return W_o
```

4. EXPERIMENT AND VERIFICATION

To verify the feasibility of the proposed method, we prepare massive IDS events as a dataset for the evaluation, in cooperation with a real-world SOC. We utilized actual security events collected from network security device IDS/IPS installed in the SOC. The dataset consists of various types of signature-based IDS/IPS events numbering more than 100 types, excepting threshold-based events such as scanning or flooding. Moreover, to apply the proposed method, only security events with payload encoded

Table 3. Detail information of IDS dataset before label refinement

Character	Description (Unicode)	Usage in payload
Provider	Real-world SOC	Name is blinded
Period	2017.01.01-2018.12.31	2 years
Type	123 signatures	ASCII encoding based security events (e.g., web.)
Size	1,186,110 events	Malicious: 0.0002%, Benign: 1,185,907 (99.9998%)
Class	Malicious and Benign	Verified by agents
Feature	12 kinds	IP, Port, Payload, etc.

IDS, Intrusion Detection System; SOC, Security Operating Center.

Table 4. Change in word count by step

Step	Count
# of all terms	237,949
# of class biased terms	11,865 (TP: 289/FP: 11,576)
# of terms terms with a frequency of less than 100	976
# of terms with term weight greater than *k*	5,879
# of stopwords finally selected	12,052

TP, True-Positive; FP, False-Positive.

as ASCII values were selected. The detail of the dataset is shown in Table 3.

The dataset basically includes 12 features, such as IP (source, destination), port (source, destination), protocol, direction, packet size, encoded payload, decoded payload, attack type, time, and class. First, the terms are extracted from payloads through our own tokenizer after defining the significant characters following Section 3.1. As can be seen in Table 2, we extracted 237,949 terms from 1,1864,110 payloads, calculated TF-IDF values according to Section 3.2, and sorted them in descending order, called set *U*. To find and remove class-biased terms from set *U*, we count the frequency of term appearances for each label. As a result, label unique terms (TP: 289 terms, FP: 11,576 terms) were extracted from true-positive and false-positives, respectively. At this time, we segmented the sentences using our own tokenizer used to TF-IDF. And then, we should also determine the threshold *k* to eliminate meaningful terms. In this research, the *k* is determined following Zipf's law (Gabaix, 1999) to distinguish between meaningful and meaningless terms. By considering the frequency appearance and the ratio of appearances for each term, we can decide an appropriate threshold value *k* following Equation 9. In order to exclude high-frequency words from the stopwords, after sorting in descending order by word frequency, all terms with indices from 1 to *k* were deleted from stopwords; we found the point *k* where it exceeds 90% by calculating the cumulative distribution function of word usage frequency.

$$remove\ term_i\ from\ U\ \{1 \le i \le k, \quad k | \frac{\sum_{d=1}^{k} frequency\ of\ term\ t}{\sum_{d=1}^{n} frequency\ of\ term\ t} \ge 90\%\} \quad (9)$$

Also, although not introduced in Section 3, we also excluded terms with a frequency of less than 100 from stopwords because most of them are one-time occurrences. After removing the label biased terms, the number of members in set *U* is 226,084, the number of terms with a frequency of less than 100 is 976, and the value of *k* is 5,879. In this progress, we finally defined a total of 12,052 terms as stopwords. Table 4 shows the summary of stopwords generation.

Finally, following the experiment for identifying the effectiveness of label refinement in Section 3.3 based on decoded payload, we select 203 malicious security events verified by humans who worked in the real-world SOC carefully and calculate similarity between them and the other events. In this process, we determine the similarity measure and threshold by cosine similarity and 80%, respectively. To experimentally find the most appropriate similarity threshold, we select 10 types of security events with the highest frequency appearance and surveyed the consensus rate of human agents for true-positive events

 https://doi.org/10.1633/JISTaP.2022.10.S.14

modified labeling, according to thresholds which were varied in 5% increments from 60 to 90, and draw the graph; According to Fig. 4, we can see the inflection point of graph at 80%. That is, at a threshold smaller than the inflection point, security events are not converted to true-positive, and at a threshold higher than the inflection point, too many security events are converted to true-positive. Accordingly, we determined the threshold to 80% and subsequent experiments were performed.

As can easily be seen in Table 5, there is a significant change in the number of security events after label refinement. About 132,000 benign events are transformed as malicious events. Totally, the dataset is reorganized containing 88.84% of benign events (i.e., about 10% decrease) and 11.16% of malicious events increased about 65,000% from the original dataset. By refining labeling, we can keep the label consistent and solve the imbalance of training data.

We conducted an experiment to prove the performance of each step. To demonstrate our method, we generated a simple DNN model by learning the data constructed through each step. Table 6 is a confusion matrix showing the test results of the model trained through each data. It shows the effectiveness of each step. In Experiment 1, preprocessing was not performed at all, and in Experiment 2, preserving significant characters and stopwords were performed together. The two methods are closely related, so we tested them together. Finally, in Experiment 3 label refinement was performed. The insufficient TP training data set was supplemented by oversampling, and a test was conducted with 10,000 recently detected security events in SOC. To make up for the insufficient TP data, we generated 1,000 TP data through repeated oversampling. The experiments show that all metrics except precision improve sequentially as each step is applied. Experiment 3 has the highest score in accuracy, recall, and f1-score. Through this, we can show the superiority of the proposed method. Experiments 1 and 2 showed 100% precision due to oversampling to replenish for insufficient TP data, whereas in Experiment 3, after label refinement, the AI model was unable to predict the TP labels of some data due to the increase in the type and number of TPs. This is a problem that would not have appeared if there were enough TP data in advance. Since it is independent of the purpose of this experiment, we did not deal with it in this paper.

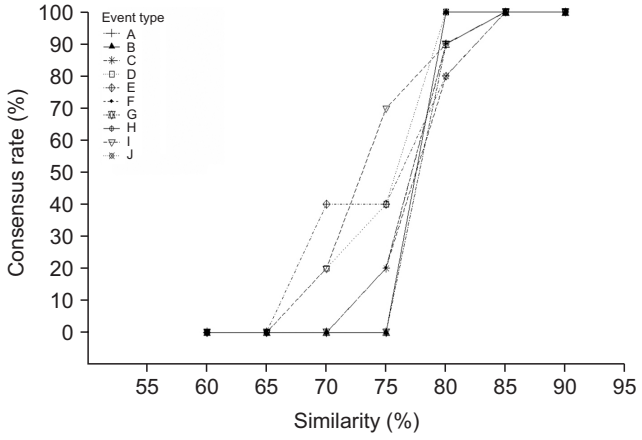

Fig. 4. Consensus rate according to the similarity threshold.

5. CONCLUSION

In this research, we proposed the network packet preprocessing method to build a high performance AI-model to be operated in in real-world SOC. To make a high-performance enough AI-model to be applied to the real SOC, a proper packet handling method is required, because the packets mix various formats such as machine

Table 5. Comparison of original and refinement dataset

Label	Before	After	Note
Malicious	203 (0.0002%)	132,391 (11.16%)	About 65,000% increase
Benign	1,185,907 (99.9998%)	1,053,719 (88.84%)	About 10% decrease

Table 6. Change of the number of dataset

Experiment no.	Experiment	Accuracy	Precision	Recall	F1-score
1	None	99.43%	100%	94.60%	97.23%
2	3.1 & 3.2	99.70%	100%	97.08%	98.52%
3	3.1 & 3.2 & 3.3	99.79%	99.30%	98.61%	98.95%

and natural language. To enhance the performance of the AI model, we suggested a security-specified packet preprocessing method consisting of preserving significant special characters, generating a stopword list, and class label refinement. The evaluation was performed in terms of identifying the effectiveness of these three steps; furthermore, we verified by comparing the performance of the models generated through the training data to which the proposed method was applied, and not. We were able to verify the superiority of the proposed method through the performance with accuracy measures; it was achieved with high F1-scores, such as 98.95%, which is 1.7% more higher accuracy over those that did not. For future work, we will improve our similarity measure using label refinement, extract stop, and critical characters. Moreover, we will try to apply a security specified language model such as GPT (Generative Pre-trained Transformer) (Brown et al., 2020) or BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) for embedding to enhance the AI-model.

CONFLICTS OF INTEREST

No potential conflict of interest relevant to this article was reported.

REFERENCES

Almseidin, M., Alzubi, M., Kovacs, S., & Alkasassbeh, M. (2017, September 14-16). Evaluation of machine learning algorithms for intrusion detection system. *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics* (pp. 277-282). IEEE.

Banerjee, A., Krumpelman, C., Ghosh, J., Basu, S., & Mooney, R. J. (2005, August 21-24). Model-based overlapping clustering. In R. Grossman, R. Bayardo, & K. Bennett (Eds.), *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining* (pp. 532-537). ACM.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 135-146. https://doi.org/10.1162/tacl_a_00051.

Broadcom. (2013). 12Gb/s SAS: Busting through the storage performance bottlenecks. https://docs.broadcom.com/docs/12353459.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., . . . Amodei, D. (2020, December 6-12). Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)* (pp. 1877-1901). Neural Information Processing Systems Foundation.

Chen, Y. C., Li, Y. J., Tseng, A., & Lin, T. (2017, October 8-13). Deep learning for malicious flow detection. *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)* (pp. 1-7). IEEE.

Cisco. (2018). *Next-generation intrusion prevention system (NGIPS)*. https://www.cisco.com/c/en/us/products/security/ngips/index.html#~resources.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June 2-7). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 4171-4186). Association for Computational Linguistics.

Ferriyan, A., Thamrin, A. H., Takeda, K., & Murai, J. (2022). Encrypted malicious traffic detection based on Word2Vec. *Electronics*, 11(5), 679. https://doi.org/10.3390/electronics11050679.

Gabaix, X. (1999). Zipf's law for cities: An explanation. *The Quarterly Journal of Economics*, 114(3), 739-767. https://www.jstor.org/stable/2586883.

Goodman, E. L., Zimmerman, C., & Hudson, C. (2020). *Packet2Vec: Utilizing Word2Vec for feature extraction in packet data*. arXiv. https://doi.org/10.48550/arXiv.2004.14477.

Kakavand, M., Mustapha, N., Mustapha, A., & Abdullah, M. T. (2016). Effective dimensionality reduction of payload-based anomaly detection in TMAD model for HTTP payload. *KSII Transactions on Internet and Information Systems (TIIS)*, 10(8), 3884-3910. https://doi.org/10.3837/tiis.2016.08.025.

Le, T., Wang, S., & Lee, D. (2020, November 17-20). MALCOM: Generating malicious comments to attack neural fake news detection models. In C. Plant, H. Wang, A. Cuzzocrea, C. Zaniolo, & X. Wu (Eds.), *2020 IEEE International Conference on Data Mining (ICDM)* (pp. 282-291). IEEE.

Li, W., Meng, W., Luo, X., & Kwok, L. F. (2016). MVPSys: Toward practical multi-view based false alarm reduction system in network intrusion detection. *Computers & Security*, 60:177-192. https://doi.org/10.1016/j.cose.2016.04.007.

Lin, C. J., & Chen, R. M. (2021, November 24-26). An effective preprocess for deep learning based intrusion detection. In H. T. Yau, R. Stenzel, M. L. Shyu, & H. C. Lin (Eds.), *2021 IEEE/ACIS 22nd International Conference on Software En-*

gineering, *Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)* (pp. 118-121). IEEE.

Madeira, S. C., & Oliveira, A. L. (2004). Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1), 24-45. https://doi.org/10.1109/TCBB.2004.2.

McAfee. (2020). *McAfee: Network security platform (IPS)*. https://www.mcafee.com/enterprise/en-us/products/network-security-platform/getting-started.html.

Medjek, F., Tandjaoui, D., Djedjig, N., & Romdhani, I. (2021). Fault-tolerant AI-driven intrusion detection system for the internet of things. *International Journal of Critical Infrastructure Protection*, 34, 100436. https://doi.org/10.1016/j.ijcip.2021.100436.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space*. arXiv. https://doi.org/10.48550/arXiv.1301.3781.

Open Information Security Foundation. (2020). *Suricata 5*.0.2, *open-source IDS/IPS/NSM engine*. https://suricata-ids.org/.

Pennington, J., Socher, R., & Manning, C. (2014, October 25-29). GloVe: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532-1543). Association for Computational Linguistics.

Sindrilaru, E. A., Peters, A. J., Adde, G. M., & Duellmann, D. (2017). EOS developments. J Phys: Conf Ser, 898, 062032. https://doi.org/10.1088/1742-6596/898/6/062032.

Sommer, R., & Paxson, V. (2010, May 16-19). Outside the closed world: On using machine learning for network intrusion detection. In P. Kellenberger (Ed.), *2010 IEEE Symposium on Security and Privacy* (pp. 305-316). IEEE.

Wang, Z., Lai, Y., Liu, Z., & Liu, J. (2020). Explaining the attributes of a deep learning based intrusion detection system for industrial control networks. *Sensors (Basel*, Switzerland), 20(14), 3817. https://doi.org/10.3390/s20143817.

Yang, H., He, Q., Liu, Z., & Zhang, Q. (2021). Malicious encryption traffic detection based on NLP. *Security and Communication Networks*, 2021, 9960822. https://doi.org/10.1155/2021/9960822.