Received April 4, 2022, accepted May 9, 2022, date of publication May 18, 2022, date of current version May 25, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3176098

Generating Labeled Training Datasets Towards Unified Network Intrusion Detection Systems

RYOSUKE ISHIBASHI[1], KOHEI MIYAMOTO[1], CHANSU HAN[2], TAO BAN[2], (Member, IEEE), TAKESHI TAKAHASHI[2], (Member, IEEE), AND JUN'ICHI TAKEUCHI[1], (Member, IEEE)

[1]Graduate School and Faculty of Information Science and Electrical Engineering, Kyushu University, Fukuoka 819-0395, Japan
[2]National Institute of Information and Communications Technology, Koganei, Tokyo 184-8795, Japan

Corresponding author: Kohei Miyamoto (miyamoto@me.inf.kyushu-u.ac.jp)

This work was supported by the Ministry of Internal Affairs and Communications, Japan, conducted under a Contract of Research and development on IoT malware removal / make it non-functional technologies for effective use of the radio spectrum (MITIGATE) among Research and Development for Expansion of Radio Wave Resources under Grant JPJ000254.

ABSTRACT It is crucial to implement innovative artificial intelligence (AI)-powered network intrusion detection systems (NIDSes) to protect enterprise networks from cyberattacks, which have recently become more diverse and sophisticated. High-quality labeled training datasets are required to train AI-powered NIDSes; such datasets are globally scarce, and generating new training datasets is considered cumbersome. In this study, we investigate the possibility of an approach that integrates the strengths of existing security appliances to generate labeled training datasets that can be leveraged to develop brand-new AI-powered cybersecurity solutions. We begin by locating communication flows that the deployed NIDSes detect as suspicious, investigating their causal factors, and assigning appropriate labels in a universal format. Then, we output the packet data in the identified communication flows and the corresponding alert-type labels as labeled data. We demonstrate the effectiveness of the labeling scheme by evaluating classification models trained with the labeled dataset we generated. Furthermore, we provide case studies to examine the performance of several commonly used NIDSes and on practical approaches to automating the security triage process. Labeled datasets in this study are generated using public datasets and open-source NIDSes to ensure the reproducibility of the results. The datasets and the software tools are made publicly accessible for research use.

INDEX TERMS Network intrusion detection system, security alert, packet data analysis, security data labeling, public dataset, packet replay.

I. INTRODUCTION

Recently, cyberattacks have become more advanced and prevalent. To defend our networks from such attacks, it is necessary to distinguish and detect anomalous communications. Therefore, the process that monitors and analyzes events occurring on computer systems and the Internet, is used to detect suspicious signs of intrusion [2]. An intrusion detection system (IDS) is a system that automatically detects intrusions and alerts the system administrator of the results. Particularly, a network intrusion detection system (NIDS) scans network traffic to detect anomalous activities [3].

Recently, there has been extensive studies on anomaly-based NIDS with AI [4]–[21]. A high-quality labeled dataset is an essential requirement for developing such AI-powered NIDSes. For this purpose, we need labeled training datasets, even if unsupervised learning is employed because labeled ones are necessary in evaluation. Many researchers and organizations have made datasets for an NIDS publicly available, covering several attack types, data formats, execution environments, etc. [22]–[24]. To develop an NIDS, many researchers use public datasets.

Many datasets predate 2000, such as *KDD'99* [25] and *DARPA'98* [26], [27], are still widely used. These outdated datasets cannot handle various cyberattacks, and there is no way to prevent attacks that have not been observed. Although numerous new datasets have been released recently, not all conditions are perfect, and there are several limitations. For example, 1) attacks' types and volumes are limited based on the dataset. 2) Several datasets capture simulated attacks in a virtual environment, which are likely to differ from real behavior. 3) Numerous datasets only disclose flow data,

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

limited features, or logs, not packets. 4) Owing to privacy concerns, it is difficult to release datasets observed in a real environment, and assigning labels to real attacks is difficult. 5) Finally, few datasets consider the training and testing for machine learning models.

To train an AI-powered NIDS model, the above datasets are not satisfactory to use. Hence, we need a method to create nice datasets for AI-powered NIDS models. It would be convenient to quickly generate labeled datasets in one's environment, depending on one's purposes. Therefore, in this study, we propose an approach that integrates the strengths of existing security appliances (NIDSes) to facilitate the generation of labeled training datasets. Once the communication flow causing an alert issued by existing NIDSes is identified, the raw packet data directly related to the alert can be discovered. However, most existing NIDSes do not provide direct information on the communication flows that trigger the alerts. Based on alerts issued by numerous NIDSes, we should automatically associate triggering communication flows uniquely with the alerts and properly label them. Here, we define a "communication flow" as a group of the packets observed over a continuous period and identified by the following five tuples: source IP address, source port, destination IP address, destination port, and protocol. The packet data corresponding to the identified communication flows and the alert-type labels are output as a labeled training dataset.

Next, we show the labels' validity in the generated dataset and investigate the possibility of using this dataset to train an AI-powered NIDS model. In fact, we verified that all the identified communication flows include all the packets that caused alerts, that those did not include wrong packets, and that those were properly labeled. Furthermore, we train gradient boosted tree classifiers implemented in *XGBoost* [28] with the dataset to examine the possibility of developing AI-powered NIDS models. Thus, we found that the generated datasets represent characteristics which classifiers can learn. Additionally, we present case studies to evaluate the performances of the numerous existing NIDSes used in the dataset generation and to solve tasks such as automating the security triage process.

In summary, this study afforded the following contributions:

- This study is the first practical effort to generate a labeled training dataset for NIDS independent of the data observation environment and is a state-of-the-art, general-purpose study that shows its possibility.
- We propose an approach that leverages the existing NIDSes' solutions to associate the trigger packets for alerts, with the caused alerts automatically.
- We show the labels' effectiveness in the generated dataset, train gradient boosted tree classifiers and present the possibility of implementing an AI-powered NIDS model.
- To ensure this study's reproducibility, we generated labeled training datasets using public datasets and open-source NIDSes. The generated dataset and the complete program code of our method are publicly available.[1]

The remainder of the paper is organized as follows. Section II provides an overview of the related studies on public datasets and AI-powered NIDSes. Section III presents the preliminaries regarding a proposed approach. In Section IV, we propose an approach to identify the communication flows that cause alerts issued by existing NIDSes. Section V explains how to generate labeled training datasets based on a public dataset using the proposed approach. In Section VI, we present an experimental evaluation of the labels' effectiveness in the generated dataset and the classifier training's effectiveness and conclude this study in Section VII.

II. RELATED WORK

In this section, we describe related works on NIDS and datasets published for NIDS research. We also provide an overview of related studies on NIDS models using machine learning.

A. NIDS AND PUBLIC DATASET

This section introduces NIDS in general and describes related studies on publicly available datasets for NIDS R&D. There are three main types of IDS: signature-based, anomaly-based, and hybrid [29]. A Signature-based IDS has a low false-positive rate but cannot detect zero-day attacks. However, an anomaly-based system can detect zero-day attacks but generally has a high false-positive rate. Hybrid IDS incorporates both approaches. Furthermore, there are NIDS, which analyzes traffic, and a host-based IDS, which analyzes the software environment on the host. This study focuses primarily on NIDS.

Data used in NIDS are mostly packet data (stored as PCAP files) and flow data (stored as NetFlow records). Flow data are a more abstract form of packet data that aggregates packet information into sessions of the same source and destination. Here, we present some survey studies on public datasets and investigate how some datasets were generated.

The survey study by Ring *et al.* [22] provides a comprehensive compilation of useful datasets for NIDS research with the format and characteristics of 34 datasets presented. Information such as the type of attack included in the datasets, data format, whether the data are anonymized, data volume, traffic observation environment, whether the data are split between training and testing data, and whether the data are labeled are systematically investigated. The survey study by Hindy *et al.* [24] details the types of attacks each dataset contains for 30 datasets. They discovered that the *CSE-CIC-IDS 2018* dataset [30] covers the most diverse attacks. They also examined the usage of each dataset and reported that *KDD'99* [25] was used in more than half of the NIDS studies. As mentioned above, characteristics vary among the datasets and should be carefully selected according to study objectives.

[1]https://github.com/suuri-kyudai/Generating-Dtaset-for-NIDS

Next, we evaluate how some datasets were generated. The *CSE-CIC-IDS 2018* dataset publishes raw packet data observed in a virtual network environment. A simulated attack was conducted, and its processing time, source IP address, and destination IP address were given as labels. Thus, many labeled datasets are generated by conducting attacks in a virtual environment, although the types of attacks differ significantly. However, such simulated attacks are unrealistic, and the labels offer only rough information.

Similarly to our study, the *PUF* dataset [31], is the only dataset that uses the output of existing NIDS and labels the traffic captured in a real network environment. However, it differs from our approach because it is flow data, DNS-specific, and the dataset is not publicly available. The NIDS alerts' IP address, port number, and timestamp in the *PUF* dataset were compared, and a label was assigned when a match was found. Although the concept is similar to our proposed approach, our study allows finer-grained packet-based labeling and leverages multiple existing NIDS. Additionally, this study proposes a generic approach for generating labeled training datasets by replaying public datasets, and the generated datasets and source code are publicly available for reproducibility.

B. AI-POWERED NIDS

In this study, we not only generate datasets for NIDS research but also examine whether AI-powered NIDS models can be practically trained. To this end, we investigate the latest NIDS research using AI technology. AI-powered NIDS mainly consists of data input, feature extraction, model training, and alert output. Several studies leverage and apply various existing AI approaches to their models, such as machine learning, data mining, and deep learning. Active studies have been conducted from the late 1980s to the present [23], [24], [29], [32].

For example, deep learning approaches such as deep neural networks [4]–[6], recurrent neural networks (RNN) [7]–[9], and convolutional neural networks (CNN) [10]–[12] are applied as supervised learning models. Furthermore, decision trees, random forests [13]–[16], and support vector machines [17], [18] are widely used. However, unsupervised learning models are often used as anomaly detection approaches for unlabeled data because they are generally less accurate than supervised learning models. Autoencoder approaches have attracted attention [19]–[21] but are generally few because of the difficulty of quantitative evaluation in unsupervised learning models. Thus, NIDS research using existing AI models has been active recently.

Meanwhile, a crucial aspect of AI-powered NIDS is how feature extraction is conducted, greatly affecting performance. The *KDD'99* dataset[2] [25] is the most frequently used dataset, consisting of unique statistics different from flow data. It has 41 features consisting of basic features about TCP connections, features about content using domain knowledge, features about traffic per time window, and features about destination hosts. Most of the related studies used features from the *KDD'99* dataset as-is. The *NSL-KDD*[3] [33] dataset is an enhanced version of the *KDD'99* dataset and consists of the same features.

Other impressive feature extraction approaches from recent studies are presented below [34]–[36]. Recently, numerous neural network-based (NN-based) feature extraction approaches have been proposed. Embedding a packet's header and payload into a vector using *word2vec* [37] for each byte is widely applied. The embedded representation is derived from the information in the surrounding packets (context). However, these NN-based feature extraction approaches require periodic retraining because of the possibility of data drift. Furthermore, packet information for patterns that have never appeared before would likely not yield the expected embedded representation.

Next, we introduce the features proposed by Mirsky *et al.* in their NIDS called *Kitsune* [21]. From the meta-information for each packet, several incremental statistics are calculated, and 23-dimensional features are extracted. The distinctive aspect of this incremental statistic is that it introduces a damping algorithm in which a decay function is calculated for each time window. They specified five-time windows and extracted a total of 115-dimensional features. Anomaly detection using autoencoders recorded robust and highly accurate results for various attack types. This approach is used as a feature extraction approach in the experiments in Section VI in this study because, unlike *KDD'99*, it can extract features at the packet-level and does not suffer from data drift compared to NN-based feature extraction approaches.

III. PRELIMINARIES

In this section, we present the preliminaries for the proposed approach in Section IV. First, we introduce the data handled in this study. Second, the detail of NIDS alerts, which is a necessary factor, is shown. Third, we describe a communication flow defined in this study. Finally, we discuss the performance of $\mathrm{NIDS_A}$ that is different from the others.

A. USED TRAFFIC DATA

In Section IV, we used traffic data that were observed by the *T-Pot* [38] and alerts issued by three NIDSes which constantly monitor the traffic data generated at *T-Pot* to identify the communication flows that cause alerts issued by existing NIDSes. Here, the three NIDSes are represented as $\mathrm{NIDS_A}$, $\mathrm{NIDS_B}$, and $\mathrm{NIDS_C}$. A honeypot can generate a collection of exploit codes and malware samples by being attacked on purpose [39]. *T-Pot* can be related to various attacks since it enables various honeypots to function. We captured the traffic data of *T-Pot* by a packet sniffer and saved it in *PCAP* format.

[2]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[3]https://www.unb.ca/cic/datasets/nsl.html

TABLE 1. Main items of each NIDS alert. $NIDS_A$ has two items, *start* and *end* for each timestamp, but $NIDS_B$ and $NIDS_C$ have only one item: *rt*. In addition, $NIDS_B$ and $NIDS_C$ have *spt*, but $NIDS_A$ does not.

$NIDS_A$		$NIDS_B$, $NIDS_C$	
Item	Explanation	Item	Explanation
start	Start Timestamp of Event	*rt*	Timestamp of Event
end	End Timestamp of Event	*src*	Source IP Address
src	Source IP Address	*dst*	Destination IP Address
dst	Destination IP Address	*cat*	Alert Category
cat	Alert Category	*lev*	Alert Level
lev	Alert Level	*spt*	Source Port Number
dpt	Destination Port Number	*dpt*	Destination Port Number

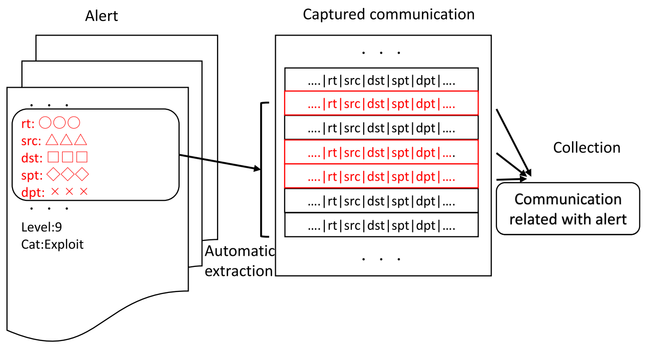

FIGURE 1. Flow of extracting communication related with the alert.

B. FORMAT OF ALERTS

An NIDS issues an alert after detecting a suspicious communication. The alert contains information about the communication. The format of all alerts issued by the three NIDSes is *Common Event Format (CEF)*. *CEF* is a standard log format developed by Micro Focus ArcSight.[4] *CEF* is an extensible, text-based format designed to support multiple device types by offering the most relevant information [40]. The typical format of *CEF* is as follows.

CEF: Version| Device Vendor| Device Product| Device Version| Signature ID| Name| Severity| Extension

For example, the signature ID shows the classification of an event, and the severity shows the threat level. We can obtain alerts with these features in the same format as NIDSes that use *CEF*, but the threat level classification and scale depend on the product. Additionally, we can perform an extension of several different items. Depending on the product, the extension includes various items. For example, the three NIDSes include the items shown in Table 1. Here, *lev* is the alert level that indicates the threat level of a detected event, and *cat* is the alert category that indicates the type of detected event.

The extension field contains details of an event, such as the source IP address, and the destination IP address.

C. DEFINITION OF COMMUNICATION FLOWS

Generally, NIDSes do not specify the packets that caused each alert. However, by referring to the alert, we can extract the communication associated with the alert. First, we refer to the items that were presented in Section III-B. The data associated with these items include a timestamp, IP address, and port number. Based on these data, we can extract packets.

Fig. 1 shows the process of referring to an $NIDS_B$ alert and extracting the related packets. Here, we used the five items: *rt* is the timestamp that indicates the alert's received time, *src* indicates the source IP address, *dst* indicates the destination IP address, *spt* indicates source port number, and *dpt* indicates the destination port number. We extracted the packets that matched the IP addresses and port numbers and were conducted within the range of $rt \pm \alpha$. In this study, we set α to 1 day.

Our aim is to automatically generate abundant training data. In this process, we determined to follow the convention to summarize the packet data as unidirectional flows using the 5-tuples, namely, source IP, source port, destination IP, destination port, and protocol. A so defined flow can be a reasonable unit for grouping communication packets as the causal factor of an NIDS alert. A coarser flow definition will render the analysis more complicated as irrelevant packets have a much larger chance to be grouped together. For example, explicit beginning and ending of the communication can be located for a unidirectional flow by searching for the controlling TCP_SYN and TCP_FIN packets; while in a coarsely defined flow based only on matching source and destination IP addresses, the beginning and ending of the communication cannot be easily located in most cases. The experiment results prove that such a flow definition is efficient and effective for most cases, and it is compatible with all the examined NIDSes.

In summary, in this study, the communication flow of a TCP communication refers to a single TCP connection that is considered as the cause of alerts. Specifically, a causing flow that associates with an alert is the one which matches the 5-tuples of the alert and whose starting and ending time have *rt* falls in between of them. On the other hand, the communication flow of non-TCP communications are defined to be the packets that matches the 5-tuples and are captured in the time range of $rt \pm \alpha$.

D. REGARDING $NIDS_A$

Among the three NIDSes, $NIDS_A$ has two main differences from the others. First, alerts from $NIDS_A$ do not have a source port number. The proposed approach requires a source port number. Thus, it is essential to develop another approach for acquiring the source port number associated with alerts from $NIDS_A$. Second, $NIDS_A$ can show causative communication data associated with alerts. These data were used in the evaluation of the proposed approach.

We can obtain the above two types of data using the vendor's API. We can obtain the trigger packets and other information about the alert after sending a request with the timestamp and IP address from the alert. The information includes a timestamp, IP address, port number, and more. The relationship between alerts and API response data is many-to-many. Sometimes, one query returns multiple responses, and different requests returns the same responses. However, this

[4]https://marketplace.microfocus.com/arcsight

issue does not arise in this study since we can identify and make the groups of the same communication flow using the proposed approach. We will provide more information about this in Section IV.

IV. ASSOCIATING NIDS ALERTS WITH THEIR COMMUNICATION FLOWS

The proposed approach for generating labeled training datasets can be separated into two main phases. The first is the phase of associating NIDS alerts with their communication flows, and the second involves generating a labeled training dataset using the associated information. The former is presented in this section and the latter in Section V. Evaluation experiments were conducted in each of these phases. In this section, we present the details of the former approach and evaluation experiments and examples of its application to other problems.

A. PROPOSED METHOD

As indicated in Fig. 2, the approach has three elements: obtaining the information about the communication flows from alerts or the API, extracting the communication flows related to the alerts, and grouping the alerts with the same communication flows. At the end, one communication flow will be associated with one or more alerts.

1) OBTAINING INFORMATION REGARDING COMMUNICATION FLOWS

First, we must obtain the information about the communication flows from the alerts to extract them. As mentioned in Section III-C, the following information is required to extract the communication flows: IP address, port number, and timestamp. Among the three NIDSes, alerts from NIDS_B and NIDS_C contain this information, but alerts from NIDS_A do not have a source port number. However, we can obtain the source port number using the API provided by the vendor of NIDS_A.

We can obtain the trigger packets and the related information concerning them through the API of NIDS_A, as mentioned in Section III-D. This information includes the IP address, port numbers, and timestamp. There is an issue that the relationship between alerts and API response data is many-to-many. However, we can regard this relationship as one in which one API response is related to many alerts from an API response. That is, multiple alerts are associated with a communication flow. This demonstrates that for the alert of NIDS_A, the desired relationship is obtained using the API before applying the proposed approach. One alert is also related to several communication flows. In this case, one alert can affect several training labels later. However, this is an inherent feature of this system, so we tolerated it.

2) EXTRACTING COMMUNICATION FLOWS RELATED WITH ALERTS

In this approach, we extract communication flows according to IP Address, Port Number, and Timestamp. The extraction process is shown below.

First, all traffic data acquired before the alert's occurrence date and one day before and after are obtained. This is because events can continue over several days. Second, the packet data that match the IP address and port number of the alerts from all traffic data are extracted. Finally, the communication flows concerning the alert's timestamp are extracted from the above data by the approach shown in Fig. 3. If the target communication is not TCP, among the communications extracted under the conditions of IP address and port number, extract the packets of the same protocol closest to the timestamp. If the target communication is TCP, a TCP connection whose start time is closest to the timestamp is extracted. Here, we tend to extract a TCP flow that starts before the alert's *rt* time as its causal factor: The *rt* of an alert indicates the detection time of the attack which should be later than the starting time of the communication and variational time latency due to the detection mechanisms of different NIDSes have to be taken into consideration.

3) GROUPING THE ALERTS OF THE SAME COMMUNICATION FLOW

Finally, grouping the alerts from the same communication flows. This process aims to integrate information from the alerts related to the same events. There are two advantages to this approach.

First, we can reduce the cost of machine learning. This may depend on the learning approach, but an AI-powered NIDS that learns alerts from the same communication flow will output several alerts per communication flow. Here, the NIDS learns many times about the same communication flow, which may lead to unnecessary bias towards them. On the other hand, an NIDS that learned the alerts as a group will process the communication flow only once. We can obtain improved training speed and generalization performance by removing the redundancy in the data.

The second advantage is the degree of freedom associated with generating training data. Alert data are used to generate training data. The range of these data can be broadened by grouping alerts. In addition to using the alert data as is, we can, for example, calculate the average alert level for the same attack and use it as a new type of teacher label.

Alert grouping uses the communication flow according to the approach described in Section IV-A2. If each alert's communication flows are identical, the alerts associated with their communication flows are grouped.

B. EVALUATION OF THE PROPOSED METHOD

In this section, we examine whether the proposed approach is consistent with our purpose. We conducted evaluations for the following three measures:

- *Measure 1*: Whether the located communication flow contains packets that caused the alert.

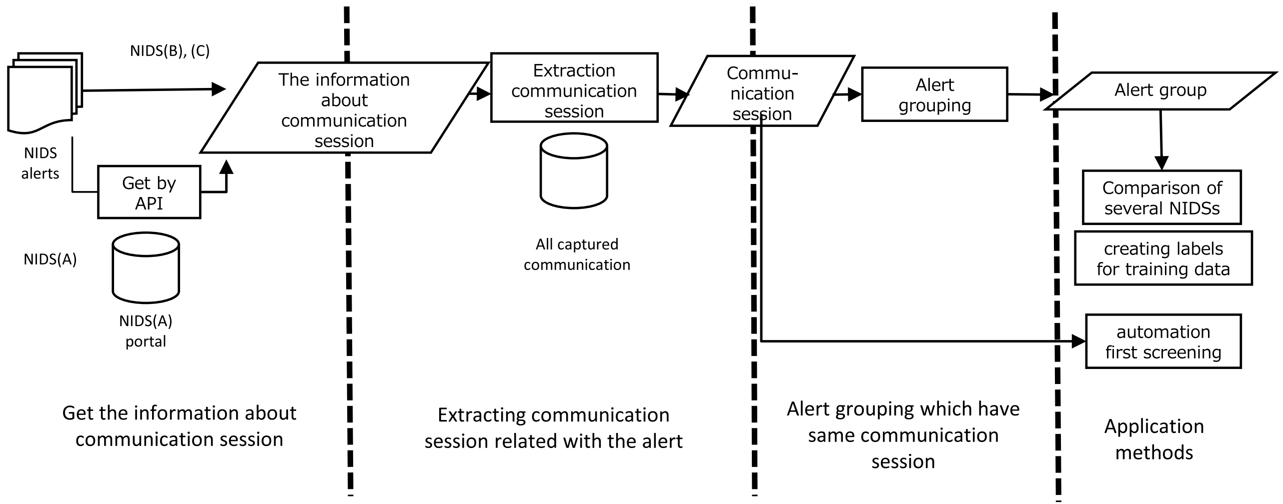
NIDS(B), (C)
NIDS alerts
Get by API
NIDS(A)
NIDS(A) portal
The information about communication session
Extraction communication session
All captured communication
Commu-nication session
Alert grouping
Alert group
Comparison of several NIDSs
creating labels for training data
automation first screening
Get the information about communication session
Extracting communication session related with the alert
Alert grouping which have same communication session
Application methods

FIGURE 2. **Overview of the approach in Section IV. Rectangles and parallelograms represent processing steps and processed data, respectively.**

TABLE 2. **The detail of data using each evaluation.**

	Period	Appliances	Data
Measure 1	Jul. 16, 2020 – Sep. 16, 2020	NIDS_A, NIDS_B, NIDS_C	Communication flows, trigger packets of each alert
Measure 2	Jul. 16, 2020 – Sep. 16, 2020	NIDS_A, NIDS_B, NIDS_C	Communication flows
Measure 3	Sep. 1, 2020 – Sep. 5, 2020	NIDS_A, NIDS_B, NIDS_C	Grouped alerts (that has alerts of several NIDSes)

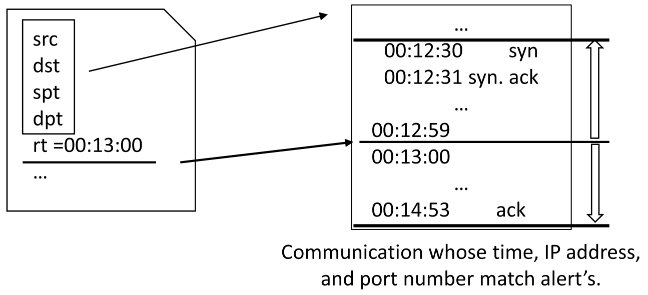

FIGURE 3. **Flowchart of the proposed approach. Checking the packet data according to the timestamp, which was extracted under the condition of IP address and port number.**

- *Measure 2*: Whether the located communication flows satisfy integrity condition – contain no extra communication and lack no related packets.
- *Measure 3*: Whether each the alert of the same group point to the same event.

1) DATA FOR EVALUATIONS

We used traffic data, which *T-pot* connected with clients in our network, and alerts served by NIDS_A, NIDS_B, NIDS_C. Each evaluation used the data generated by the proposed approaches from the above data: communication flows from Section IV-A2, true trigger packets of each alert from NIDS_A API, and alert groups from Section IV-A3. Table 2 shows the period, appliances, and detail of data, which each evaluation used.

2) *MEASURE 1*: WHETHER THE LOCATED COMMUNICATION FLOW CONTAINS PACKETS THAT CAUSED THE ALERT

We have checked whether each communication flow extracted using the approach in Section IV-A2 had trigger packets of each alert. In this evaluation, we used 6,542 pairs of data corresponding to communication flows extracted by the approach described in Section IV-A2, and the true packets of NIDS_A corresponding to it from the above data. This evaluation verified that each communication flow had the true trigger packets of each alert. Therefore, all the communication flows had the true trigger packets. Thus, communication flows of NIDS_A alerts include true trigger packets. This evaluation has not been able to verify NIDS_B and NIDS_C since they did not indicate true trigger packets of their alerts.

3) *MEASURE 2*: WHETHER THE LOCATED COMMUNICATION FLOWS SATISFY INTEGRITY CONDITION

In Section IV-A2, the purpose is to extract one TCP connection or only one protocol communication. If a communication flow has numerous TCP connections or several types of protocol communication, it is incomplete. We investigated whether the communication flows lacked extra communication. This can be self-evident however, we did it for confirmation. This evaluation used 1,000,316 communication flows related to three NIDS alerts extracted using the approach in Section IV-A2. We checked whether the communication flows included several TCP connections or other types of protocol communication. Particularly, we determined the number of three-way handshaking and protocols. The results show that none of the communication flows included any extra communication.

4) *MEASURE 3*: WHETHER EACH THE ALERT OF THE SAME GROUP POINT TO THE SAME EVENT

In the proposed approach, we extracted the communication flows and the grouped alerts of the same communication flow.

TABLE 3. The result of *Measure 3*. "AB" indicates the number of groups with only NIDS$_A$ and NIDS$_B$ alerts, and "ALL" indicates the number of groups with alerts of all NIDSes.

	ALL	AB	AC	BC	Total
number of all groups	29	62	18	5,617	5,726
number of groups whose alerts show the same event	29	60	18	5,616	5,723

Although this approach is based on communication, it is also essential to verify whether this approach is valid from the viewpoint of events. If the alerts of a group are related to other events, the validity of this approach is lost.

We used the alert category of each alert for evaluation. The alert category shows the type of event. In this evaluation, we checked whether the categories of the same group showed the same event. Particularly, this was judged in two stages. First, we checked whether the categories of each group included the same words. For example, if we compare "Suspicious File Download" and "Malicious File Download", these categories are not identical but include some of the same words: "File" and "Download." Therefore, We judge that these categories are related to the same event.

Second, we consider the categories that exclude the exact words. For example, if we compare "RDP Vulnerability" and "CVE-2019-0708," there are no common words. However, these categories pertain to the same event. RDP is the remote desktop protocol, and the first category shows attacks using the vulnerability of RDP. "CVE" is common vulnerabilities and exposures,[5] and MITRE Corporation,[6] which is a nonprofit organization supported by the U.S. government, has numbered validation of individual products [41]. "CVE-2019-0708" shows the validation using CVE-2019-0708.[7] Thus, we can judge that these two categories indicate the same event.

In this evaluation, we used 5,726 groups with alerts of different NIDS of all the alert groups determined using the approach in Section IV-A3 from 12,074 alerts of the three NIDSes observed from Sep. 1, 2020, to Sep. 5, 2020. Table 3 shows the results. Among all the groups, 5,723 groups have been associated with alerts that are subject to the same cat. For the other 3 groups, as the alert messages do not provide intuitive categorical information about the type of the attack, we find it was difficult to confirm whether these alerts are subject to the same category. In summary, most group alerts had categories that indicate the same event, so the proposed approach can be considered valid from the event perspective.

C. APPLICATION OF RESULTS

This section introduces the three application approaches using communication flows and the alert groups obtained using the proposed approach.

[5]https://cve.mitre.org/

[6]https://www.mitre.org/

[7]https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-0708

◆$flags$: {'A':1, 'B':1, 'C':0},	# whether group has each NIDS
◆N : {'A':2, 'B':2, 'C':0},	# numbers of each NIDS
◆$\max_{\{i\in\{1,\cdots,n\}\}} L_i$: {'A':10, 'B':6, 'C':0},	# maximum levels of each NIDS
◆$\min_{\{i\in\{1,\cdots,N\}\}} L_i$: {'A':8, 'B':4, 'C':0},	# minimum levels of each NIDS
◆S : {'A':18, 'B':10, 'C':0},	# sums of each NIDS level
◆μ : {'A':9, 'B':5 'C':0},	# averages of each NIDS level
◆$\hat{S}$: {'A':1.80, 'B':1.00, 'C':0},	# normalized sums of each NIDS level
◆$\hat{\mu}$: {'A':0.90, 'B':0.50, 'C':0},	# normalized averages of each NIDS level
◆$\max_{\{i\in\{1,\cdots,N\}\}} \hat{L}_i$: {'A':1.00, 'B':0.67, 'C':0},	# normalized maximum levels of each NIDS
◆$\min_{\{i\in\{1,\cdots,N\}\}} \hat{L}_i$: {'A':0.80, 'B':0.33, 'C':0},	# normalized minimum levels of each NIDS
◆cats : {'A':['Mirai', 'Mirai'], 'B':['suspicious', 'login'], 'C':[]}	# categories of each NIDS
◆IP : IP address	
◆Port : port number	
◆Time : timestamp	

FIGURE 4. Format example of labeled training data.

1) CREATING LABELED TRAINING DATASET

This study's main purpose is to generate a training dataset for supervised learning automatically. The proposed approach uses the associated alerts with communication flows. Furthermore, it is necessary to assign suitable labels to the data. There are various items in an alert. Among them, the alert level seems to be usable as a label.

An example is shown in Fig. 4. Each item has three items; A, B, and C for each NIDS. For example, "A" of μ shows the average alert level of NIDS$_A$ alerts in the group. "cats" also contains a set of alert categories for each NIDS in the same group. We use "cats" as an abbreviation for categories. In the example of the figure, the item "cats" in "A" indicates an attack called Mirai.

The standard Alert Level also depends on the NIDS. For example, there are NIDSes with ten and five stages scales of Alert Level. Thus, we normalized Alert Level. Normalized Alert Level $\hat{L}$ is calculated according to the following equation:

$$\hat{L}_i = \frac{L_i - L_{\text{inf}}}{L_{\text{sup}} - L_{\text{inf}}}, \quad (i \in \{1, \cdots, N\}.) \tag{1}$$

For each NIDS, N shows the number of alerts, L_{sup} shows the maximum alert level, and L_{inf} shows the minimum alert level in a group.

2) COMPARISON OF NIDSes

Grouping alerts from different NIDSes can be achieved by comparing each NIDS evaluation associated with the same event. We can examine each NIDS for the event by comparing the alerts of the different NIDSes in the same group. Additionally, we can regard a single alert as an alert that is yet to be detected by other NIDSes. Therefore, using the datasets obtained by the proposed approach, we experimented by comparing multiple NIDSes. In this comparison, we used the data generated in Section IV-C1 and verified the difference in normalized alert levels in the same groups. Table 4 shows the results.

TABLE 4. The comparison result of Alert Levels from different NIDSes. Row $\hat{\mu}$ of column "A-B" shows the difference in averaged normalized-alert-level between NIDS$_A$ and NIDS$_B$ on the same group.

Parameters	A-B	A-C	B-C
$\hat{\mu}$	-0.295	-0.573	-0.042
$\max_{i\in\{1,\cdots,N\}} L_i$	-0.295	-0.604	-0.070

TABLE 5. Categories of alerts and rules of security triage which depend on each category.

Category of alert	Security triage
Suspicious File Download	If download was complete, continue to analysis. If RST packet found from client, end the analysis.
Exploit Code	Check whether the exploit code from the outside to the inside has 200OK or continuous communication.
Drive by Download	Check the progress of the attack by verifying the communication flow.
Command and Control	If the communication was from the outside to inside, and the attack failed, wait. If the communication was from the inside to the outside, identify the causative host.
Suspicious Mail	Same as Command and Control.

Thus, NIDS$_A$ tends to examine the same event to be less harmful than other NIDSes. By contrast, there is a slight difference between NIDS$_B$ and NIDS$_C$. In this study, the comparison of NIDSes is accomplished, but there is a potential to compare NIDSes in diverse settings and detail by preparing more data and using additional categories.

3) AUTOMATION OF SECURITY TRIAGE

In this study, we call the process of analyzing communication-related to the alert to judge the success or failure of an event *security triage*. If it is challenging to determine whether a communication is normal or abnormal with existing security applications, a security analyst's support is needed. For example, an analyst checks whether the event is harmful or not by referring to numerous information to determine if the security appliance detected a false positive alert [42]. However, it is difficult to perform *security triage* entirely manually due to the enormous number of alerts. Thus, automation of *security triage* is greatly anticipated.

The communication flows extracted using the approach described in Section IV-A2 are specifically and formally about TCP communication. Thus, generating automation rules of *security triage* will be easier than ever by using the above data. In this section, we show an example of this. As shown in Table 5, we generated an automation rule for each attack. We will show the two example rules: "Exploit Code" and "Suspicious File Download."

a: EXPLOIT CODE

"Exploit Code" is an attack that overuses a vulnerability. In this study, we generated the automation rule about "Mirai" of "Exploit Code." First, "Mirai" tries to log into IoT devices by telnet. After several login attempts, "Mirai" continues to attack if it is successful with login [43]. The *security triage* of "Mirai" involves checking whether the login is successful or not.

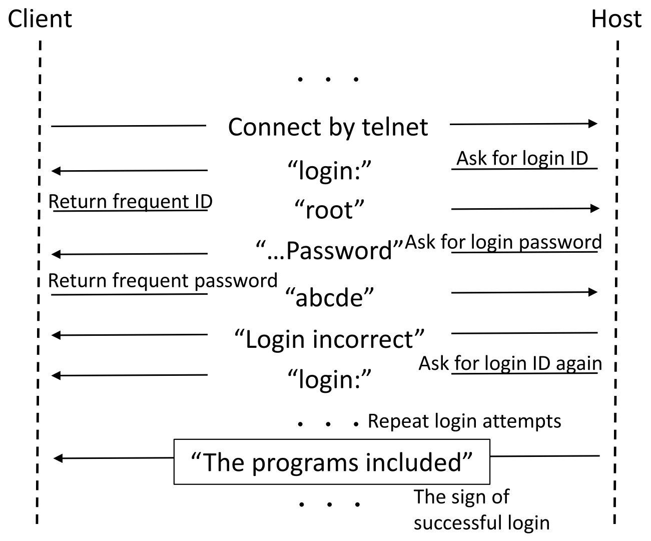

FIGURE 5. Example communication of Mirai. If you can confirm the login success message, it is judged that the attack was successful.

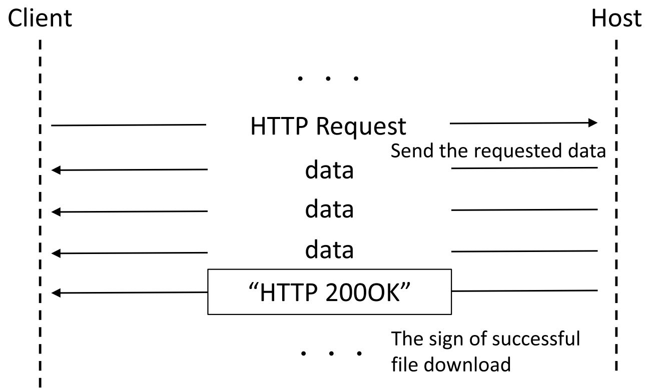

FIGURE 6. Example communication of suspicious file download. If you can confirm the download success message, it is judged that the attack was successful.

As shown in Fig. 5, while attempting a login, the client sends several IDs or passwords, but the host only sends the message, which requests the ID or password. By contrast, the host sends several messages to the client after successful login. We deem the login as success by verifying whether the host requests an ID or a password. If there is any message other than a login attempt after the login attempts, we will regard the communication as a successful attack.

b: SUSPICIOUS FILE DOWNLOAD

"Suspicious File Download" is the attack that makes the target download the suspicious file. To determine whether the attack is successful, It is necessary to examine whether an HTTP request is successful. As shown in Fig. 6, if the download is complete, we can determine the attack's success by examining whether the communication shows that the download is complete.

The case exists in which attacks are not completed within the communication flows. If the attack embeds a command in the HTTP request, we cannot judge if the attack is successful by confirming the communication flows. It is essential to check the communication between the host and other clients and internal logs. There is a potential for automation of *security triage* for generating each rule depending on the attacks. However, in this study, we did not attempt a specific approach.

D. DISCUSSION

In this section, we proposed an approach that associates each NIDS alert with their communication flows. The proposed approach's first purpose is to prepare datasets for developing a brand-new NIDS, which learns the detection approach for each existing NIDS. Particularly, the proposed approach enables generating desired datasets as long as communication data and related alerts are available.

However, there is an issue. In this section, we used *T-pot* data for the evaluations and the application approaches. However, there is a limit to the variety and reality of the communication data obtained since *T-pot* simulates the behavior of the host. In addition, although the communication flows were successfully extracted for the *T-pot* data, there may be omissions in the extraction for realistic communication data for more complex protocols and applications. Therefore, only these data are not enough for developing the new NIDS.

From the viewpoint of variety and reality, the most relevant data for the development is communication on a real live network that is usually conducted. However, it is not easy to use it because of privacy issues. Additionally, to examine a brand-new NIDS later, datasets with a true, correct label different from alerts are required. Thus, in the next section, we generate a labeled training dataset using a publicly available dataset for NIDS research.

V. OBTAINING ALERTS BY REPLAYING PUBLIC DATASETS

The proposed approach of Section IV can associate the input and output of NIDSes. What we need here is data that satisfies the following conditions

- The data are rich enough in reality and variety.
- The data do not have privacy issues.
- The data have correct labels.

In this section, we propose an approach to acquire data that satisfies the above conditions. We are also investigating and discussing the proposed approach's availability.

A. PROPOSED APPROACH TO OBTAIN ALERTS BY REPLAYING

Supervised public datasets can be considered as satisfying the above conditions. Supervised public datasets consist of several attack communication data and their labels. Such datasets were generated and published by various research institutes and organizations, and they are used in various security studies. Suppose training datasets can be generated from supervised public datasets. In that case, they are of good quality and can be used for performance evaluation. However, there are no related NIDSes alerts because they have not flowed on the specified network.

Algorithm 1 An Algorithm to Obtain Alerts by Replaying a Public Dataset

```
Require: OriginalPCAP: public datasets (PCAP file)
Ensure: ReplayedPCAP: retransmitted PCAP file,
    ReplayedAlerts: NIDS alerts obtained by monitoring
    retransmissions
    /* TCPRewrite is a partial tool of TCPReplay with packet
    editing functionality. */
1: Fragment the OriginalPCAP according to a given
   network's maximum transfer unit (MTU) using the
   TCPRewrite tool.
2: Convert OriginalPCAP's internal IP addresses to NIDS
   monitored network IP addresses using the TCPRewrite
   tool.
3: Retransmit OriginalPCAP by TCPReplay and at the
   same time captures the retransmitted communication,
   and NIDSes monitor the communication.
4: Get ReplayedPCAP and ReplayedAlerts.
```

Here, regarding the above problem, the use of *TCPReplay*[8] can be effective [44]. *TCPReplay* resend the traffic observed on another network on the prepared network. Additionally, we can get the alerts corresponding to the communication by making NIDSes monitor them. Specifically, any traffic data can be replayed by *TCPReplay* and alerts can be issued corresponding to attacks contained in the data. The desired training datasets can be generated by applying the flow shown in Fig. 7 to any supervised public datasets. The Algorithm 1 shows the details.

B. VERIFICATION OF REPLAY

Section IV shows the approach to generate training datasets from communication data and the alerts issued by the existing NIDSes. Here, we replay the supervised public datasets and verify that sufficient alerts are obtained.

1) USED PUBLIC DATASET

Supervised public datasets and NIDSes are required for replay validation. From many supervised public datasets, we chose *CSE-CIC-IDS 2018* datasets[9] [30] with reference to the survey studies on security datasets [22]–[24]. *CSE-CIC-IDS 2018* datasets contain the most diverse attacks among existing public datasets, class labels are specifically offered, and the communication data are given as packet data. Based on the above, we have determined that *CSE-CIC-IDS 2018* datasets are the most suitable datasets as the one based on which we generate training datasets by the method of replay. Furthermore, NIDS_A, NIDS_B, NIDS_C, *Snort*,[10] and *Suricata*[11] were used for NIDSes to monitor the replay.

[8]https://tcpreplay.appneta.com/
[9]https://www.unb.ca/cic/datasets/ids-2018.html
[10]https://www.snort.org/
[11]https://suricata.io/

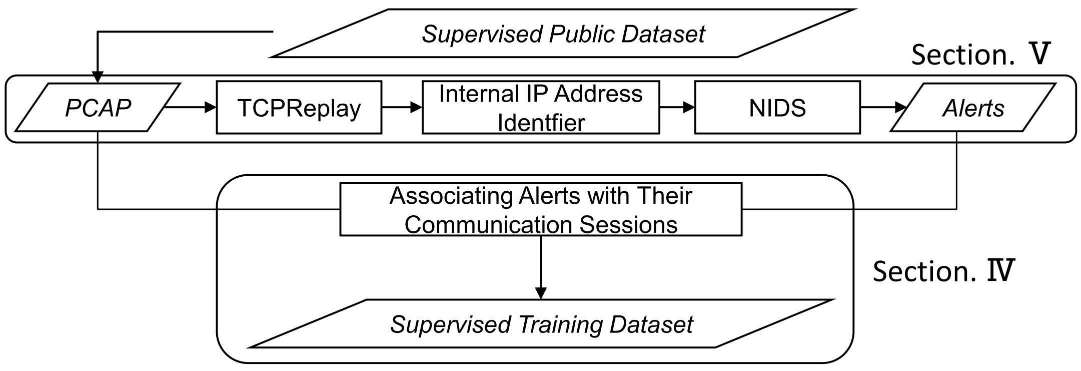
Supervised Public Dataset
Section. V
PCAP
TCPReplay
Internal IP Address Identfier
NIDS
Alerts
Associating Alerts with Their Communication Sessions
Section. IV
Supervised Training Dataset

FIGURE 7. Flow of replaying communication data and obtaining alerts. Rectangles and parallelograms represent processing steps and processed data, respectively.

TABLE 6. The result of replaying *CSE-CIC-IDS 2018* datasets with *Snort* and *Suricata*.

Date	Attack	Original	*Snort*	*Suricata*
02/14/2018	SSH-BruteForce	14,146	14,116	46
02/14/2018	FTP-BruteForce	14,116	0	0
02/15/2018	DoS-GoldenEye	14,116	13,452	13,890
02/15/2018	DoS-Slowloris	6,979	4,815	0
02/16/2018	DoS-SlowHTTPTest	14,116	0	0
02/16/2018	DoS-Hulk	14,116	14,116	0
02/22/2018	Web-BruteForce	137	135	0
02/22/2018	XSS-BruteForce	43	42	2
02/22/2018	SQL-Injection	17	16	0
02/23/2018	Web-BruteForce	123	122	0
02/23/2018	XSS-BruteForce	73	73	0
02/23/2018	SQL-Injection	33	29	0
03/02/2018	Bot-1	4,967	4,967	0
03/02/2018	Bot-2	5,345	0	1
02/28/2018	Infiltration-1	1	1	1
02/28/2018	Infiltration-2	1	1	1
03/01/2018	Infiltration-1	2	0	1
03/01/2018	Infiltration-2	2	0	2

2) VERIFYING REPLAY RESULT

In this verification, replayed *CSE-CIC-IDS 2018* datasets using *TCPReplay* on the specified network and monitored it with $\mathrm{NIDS_A}$, $\mathrm{NIDS_B}$, $\mathrm{NIDS_C}$, *Snort*, and *Suricata*. The replay speed by *TCPReplay* was set to the default, which is the original speed of the PCAP data. The maximum transfer unit (MTU) was set to the default value of 1,500 bytes. Although the *CSE-CIC-IDS 2018* datasets also include DDoS datasets, we excluded them from the verification due to the huge amount of time required to replay them on our network with a limited communication speed, 10 Mbps, that is, the original speed cannot always be realized.

Table 6 shows the results. *CSE-CIC-IDS 2018* datasets are differentiated for each attack by day. The "Original" column indicates the total number of IP addresses and port numbers combinations corresponding to each attack in the original datasets. "*Snort*" and "*Suricata*" columns indicate the number of combinations of IP addresses and port numbers detected by each NIDS.

As a result, although omitted in Table 6, no alerts were obtained for $\mathrm{NIDS_A}$, and none were obtained for $\mathrm{NIDS_B}$ and $\mathrm{NIDS_C}$ except for infiltration attacks. We think that the obtained dataset is not good enough for our purpose, which is training of supervised learning models. Then, we investigated the reason why the unsatisfactory dataset was obtained.

C. INVESTIGATING THE REASON FOR UNDETECTED ALERTS

If it is irregular that usual NIDSes do not issue alerts for the replayed public datasets which include well-known cyber attacks, there are two possibilities: 1) there is a problem with *TCPReplay* and its compatibility with the three NIDSes, 2) *CSE-CIC-IDS 2018* datasets have a problem. We verified both points.

1) WHETHER *TCPREPLAY* HAS A PROBLEM

We verified whether *TCPReplay* has a problem by replaying *T-pot* communication data. The *T-pot* communication data have flowed once on the specified network and has raised an alert of the NIDSes. Suppose the *T-pot* communication data are replayed, and no alert is obtained. In that case, it means that the replay cannot reproduce the original communication by *TCPReplay*.

In this verification, the *T-pot* communication data obtained on May 21, 2021, was replayed. Table 7 shows the results. We conducted two replays over two days and investigated the number of alerts issued by the NIDSes. Additionally, in *TCPReplay*, the replay speed can be changed, but in this experiment, the original speed was used. As a result, a sufficient number of alerts were obtained in each replay, although the number varied. Thus, it is unlikely that *TCPReplay* and its compatibility with the three NIDSes are the cause of the alert not being issued.

2) WHETHER *CSE-CIC-IDS 2018* DATASETS HAS A PROBLEM

Alerts may not be obtained due to a problem with the datasets. If *CSE-CIC-IDS 2018* datasets have any flaws, they may not be replayed correctly. Additionally, each NIDS will not issue an alert if it does not contain attack communications that *CSE-CIC-IDS 2018* datasets deserve to detect. To confirm it, it is essential to see the detection rule of each NIDS. However, since the three NIDSes are commercial ones, the detection rules are not disclosed.

TABLE 7. The result of replaying *T-pot* communication data with three NIDSes.

Replay Date	Original Alerts			Replay Alerts		
	NIDS_A	NIDS_B	NIDS_C	NIDS_A	NIDS_B	NIDS_C
May. 21, 2021	134	1,728	113,422	162	1,360	95,680
May. 24, 2021	134	1,728	113,422	195	2,345	183,780

Therefore, we attempted using open source software NIDSes (OSS NIDSes). OSS NIDSes are easy to verify because the detection rules and mechanisms are open to the public. The two OSS NIDSes, *Snort* and *Suricata*, were used in this study.

In this verification, the replayed *CSE-CIC-IDS 2018* datasets using *TCPReplay* on the specified network and monitored it with *Snort* and *Suricata* like Section V-B. Furthermore, we can select a rule set for each of OSS NIDSes. We chose *Registered Rules version.31110* for *Snort* and *Emerging Threats Rules version.11/11* for *Suricata*, respectively.

Table 6 shows the results. *Snort* issued a sufficient number of alerts for each attack. From this result, it is unlikely that *CSE-CIC-IDS 2018* datasets are the cause of the alert not being issued. However, *Suricata* issued alerts about half of the attacks and did not alert the rest. This result shows that the lack of alerts is not due to any specific problem, but simply due to differences in detection rules.

D. DISCUSSION

From our verification, we could not find why the three commercial NIDSes did not issue alerts. From this, it is likely normal for the three commercial NIDSes not to issue alerts. Since commercial NIDSes have characteristics according to the purpose, they often do not detect attacks that are not the target. This study aims to develop a brand-new NIDS. The biggest advantage is enhancing performance by combining NIDSes with different performances into one.

Consider the case of generating a training dataset based on *CSE-CIC-IDS 2018* datasets and the three commercial NIDSes and the two OSS NIDSes. A brand-new NIDS generated using this training dataset should inherit the characteristics of the above five NIDSes. When generating this new NIDS, we can add weights to the original NIDSes depending on the purpose. For example, if we place importance on getting various alerts, we can increase the weight of OSS NIDSes. Additionally, if we want a few alerts, it is better to increase the weight of commercial NIDSes. It does not matter whether alerts are obtained for each NIDS. It is more crucial to obtain diverse results for each NIDS, and the examples in this study satisfy that.

Numerous existing public datasets are claimed to be labeled. However, those labels are not strictly attached to each packet or flow, but are often vague information such as a rough timestamp and IP addresses of the attacker and victim. The proposed approach provides labels for those public datasets in finer units such as flows and packets.

We can also label unsupervised public datasets. Unsupervised public datasets are more practical and of higher quality than supervised public datasets because they often have realistic attacks performed. They include communication data obtained from practical networks such as intra-organizational and large-scale networks. Applying the proposed approach to unsupervised public datasets makes it possible to generate practical datasets with detailed labels.

The source codes used in our study and the training datasets generated by them are available on github.[12] This is a training dataset generated based on *ReplayedPCAP* and *ReplayedAlerts* obtained by replaying *CSE-CIC-IDS 2018* datasets. The data format is *pkl*, the pickle format of python and *csv*. Datasets are prepared for each flow and each packet. Furthermore, the source code is written in python, which automatically associates alerts with cause communication and generates training datasets.

VI. EVALUATION OF EFFECTIVENESS OF OBTAINED LABELS AND CLASSIFIER TRAINING

In this section, we present an experimental evaluation of the labels' effectiveness in the generated dataset and the training reuslt of classifier on the datasets.

A. COMPARISON WITH LABELS BASED ON META-INFORMATION

CSE-CIC-IDS 2018 provides meta-information on each included attack. The information includes the IP addresses of attackers and victims and the duration of each attack. For each attack, we can label packets transmitted in the duration between the attacker's IP address and the victim's IP address. The label will be the name of the corresponding attack. We also label packets not labeled by the procedure above, BENIGN. This label means that the packet is not malicious.

We regard these labels as the ground truth. To evaluate the effectiveness of the labels obtained by replaying, we compare them with the ground truth. In this section, we call the data with labels based on meta-information as *original data* and the data with labels obtained by replaying as *replay data*.

Labels of the replay data consist of alert categories defined by NIDSes. On the other hand, labels of the original data consist of names of attacks defined by the *CSE-CIC-IDS 2018* dataset. Hence, it is difficult to compare labels of the replay data to those of the original data as they are. In this paper, we compare them as in binary classification; attack or not attack. That is, we replace all labels which indicate some attacks by 1 and BENIGN labels by 0. This encoding losses information on types of attacks. However, it is enough for the purpose of comparison of both kinds of labels.

[12]https://github.com/suuri-kyudai/Generating-Dtaset-for-NIDS

TABLE 8. The comparison of labels in replay data to labels in original data. The columns "Total" show the numbers of packets. The columns "0" and "1" show the numbers of packets of each label, where 0 means benign or no-alerted and 1 means malicious.

Date	Replay			Original		
	0	1	Total	0	1	Total
02/14	4,094,056 (86.4%)	646,515 (13.6%)	4,740,571	2,356,727 (49.9%)	2,361,600 (50.1%)	4,718,327
02/15	259,904 (52.3%)	236,861 (47.7%)	496,765	185,246 (43.6%)	239,379 (56.4%)	424,625
02/22	18,281 (35.3%)	33,509 (64.7%)	51,790	29,020 (60.8%)	18,694 (39.2%)	47,714
02/23	20,873 (33.4%)	41,616 (66.6%)	62,489	30,279 (55.0%)	24,822 (45.0%)	55,101
02/28	795,795 (79.4%)	205,861 (20.6%)	1,001,656	918,763 (99.9%)	774 (0.1%)	919,537
03/01	634,415 (72.5%)	241,241 (27.5%)	875,656	780,516 (99.8%)	1,738 (0.2%)	782,254

TABLE 9. The classification results based on the time-based split.

Date	Replay									
	TP	TN	FP	FN	Accuracy	Recall	Specificity	Precision	NPV	F1-score
02/14	61	61	9	645,667	0.85	0.00	1.00	0.87	0.85	0.00
02/15	396	396	14	79	0.97	0.83	1.00	0.97	0.98	0.89
02/22	14,088	14,088	4,197	460	0.82	0.97	0.62	0.77	0.94	0.86
02/23	23,053	23,053	7,690	306	0.79	0.99	0.46	0.75	0.96	0.85
02/28	79,716	79,716	6,292	46,351	0.91	0.63	0.99	0.93	0.90	0.75
03/01	96,508	96,508	12,057	45,221	0.85	0.68	0.95	0.89	0.83	0.77
Date	Original									
	TP	TN	FP	FN	Accuracy	Recall	Specificity	Precision	NPV	F1-score
02/14	1,549	1,549	0	2,169,751	0.50	0.00	1.00	1.00	0.50	0.00
02/15	80,199	80,199	4,845	19	0.97	1.00	0.94	0.94	1.00	0.97
02/22	7,692	7,692	1,904	138	0.90	0.98	0.86	0.80	0.99	0.88
02/23	14,768	14,768	109	213	0.99	0.99	0.99	0.99	0.99	0.99
02/28	303	303	541	66	1.00	0.82	1.00	0.36	1.00	0.50
03/01	125	125	275	590	1.00	0.17	1.00	0.31	1.00	0.22

TABLE 10. The classification results based on the count-based split.

Date	Replay									
	TP	TN	FP	FN	Accuracy	Recall	Specificity	Precision	NPV	F1-score
02/14	94,192	94,192	5	229,066	0.90	0.29	1.00	1.00	0.90	0.45
02/15	109,169	109,169	8,307	9,262	0.93	0.92	0.94	0.93	0.93	0.93
02/22	14,661	14,661	3,704	2,094	0.78	0.88	0.59	0.80	0.72	0.83
02/23	20,665	20,665	395	143	0.98	0.99	0.96	0.98	0.99	0.99
02/28	67,825	67,825	4,996	35,106	0.92	0.66	0.99	0.93	0.92	0.77
03/01	87,869	87,869	114,086	32,752	0.66	0.73	0.64	0.44	0.86	0.54
Date	Original									
	TP	TN	FP	FN	Accuracy	Recall	Specificity	Precision	NPV	F1-score
02/14	1,180,800	1,180,800	37,466	0	0.98	1.00	0.97	0.97	1.00	0.98
02/15	119,676	119,676	5,887	14	0.97	1.00	0.94	0.95	1.00	0.98
02/22	9,219	9,219	1,872	128	0.92	0.99	0.87	0.83	0.99	0.90
02/23	12,214	12,214	102	197	0.99	0.98	0.99	0.99	0.99	0.99
02/28	319	319	0	68	1.00	0.82	1.00	1.00	1.00	0.90
03/01	49	49	359	820	1.00	0.06	1.00	0.12	1.00	0.08

We compared the labels for six days in *CSE-CIC-IDS 2018* (02/14, 02/15, 02/22, 02/23, 02/28, 03/01). Table 8 shows the numbers of packets of each label and the total number of packets. It also shows the ratio of each label to the total number of packets in each day. The types of attacks included in the daily datasets can be seen in Table 6.

The increase in the total number of packets in the replay label is caused by the packet fragmentation for the replay. For some dates, the distribution of replay data is significantly different from the original label. For example in 02/14, 50.1% of the total labels in the original data are positive. However, the replay data include only 13.6% of the total positive samples. A lot of positive samples were missed by the labeling with replaying. In 02/28 and 03/01, the original data include positive samples only less than 1% of total packets. However, the replay data include positive samples of more than 20%. For these dates, the replay data include many extra positive samples. These dates include Infiltration attacks. The malicious behaviors of Infiltration attacks may essentially consist of packets not between specified attackers and victims. Since our method to generate labels of the original data strongly depends on the specified IP addresses, the difference in the numbers of positive samples may be caused by a defect in the labeling criteria. On this problem, a more detailed research on the dataset will be required for future work.

B. EVALUATION VIA BINARY CLASSIFICATION

We evaluated the replay data via using them for training of binary classifiers. We used the datasets in six days in *CSE-CIC-IDS 2018* (02/14, 02/15, 02/22, 02/23, 02/28, 03/01) again. We employed the gradient boosted tree classifier implemented in *XGBoost* [28]. In concrete, we use *XGBClassifier* class with default hyper-parameters.

TABLE 11. The classification results based on the random split.

Date	Replay TP	TN	FP	FN	Accuracy	Recall	Specificity	Precision	NPV	F1-score
02/14	323,083	323,083	36	175	1.00	1.00	1.00	1.00	1.00	1.00
02/15	117,713	117,713	1,239	718	0.99	0.99	0.99	0.99	0.99	0.99
02/22	16,635	16,635	69	120	0.99	0.99	0.99	1.00	0.99	0.99
02/23	20,703	20,703	79	105	0.99	0.99	0.99	1.00	0.99	1.00
02/28	92,335	92,335	5,005	10,595	0.97	0.90	0.99	0.95	0.97	0.92
03/01	105,743	105,743	5,220	14,878	0.95	0.88	0.98	0.95	0.95	0.91
Date	**Original TP**	**TN**	**FP**	**FN**	**Accuracy**	**Recall**	**Specificity**	**Precision**	**NPV**	**F1-score**
02/14	1,180,630	1,180,630	1,504	170	1.00	1.00	1.00	1.00	1.00	1.00
02/15	119,681	119,681	40	9	1.00	1.00	1.00	1.00	1.00	1.00
02/22	9,323	9,323	19	24	1.00	1.00	1.00	1.00	1.00	1.00
02/23	12,389	12,389	57	22	1.00	1.00	1.00	1.00	1.00	1.00
02/28	360	360	15	27	1.00	0.93	1.00	0.96	1.00	0.94
03/01	760	760	16	109	1.00	0.87	1.00	0.98	1.00	0.92

In training, we specify an option *early_stopping_round* = 5, which is used to determine when the training stops.

To do packet-level classification, we require packet-level features extracted from each packet. Miyamoto *et al.* [45] proposed packet-level classification method based on features extracted by Kitsune [21]. In this section, we always use 60-dimensional Kitsune features consisting of statistics called 1D statistics in [21].

The evaluations were executed for each day. Each day's data was split into two sets in 3 ways, time-based, count-based, and random split. In time-based split, we split data based on their timestamps. Let t_{start} and t_{end} be timestamps at which positive samples appeared firstly and lastly, respectively. Then, we used $(t_{\text{end}} - t_{\text{start}})/2$ as the boundary for the split. In count-based split, we split each of the positives and negatives half-and-half. In the random split, we randomly shuffled data and sampled data half-and-half. The sampling was done with stratification.

In general, packets can have statistical relationships to near packets. Then, to avoid the effect of such relationships in testing, when we used the time-based or the count-based split, we used the former part of the split data for training and the latter part as a test set. However, these procedures of splitting have a defect. Since the dataset we used includes multiple kinds of attacks in each day and their duration is different from each other, the time-based split and the count-based split are possible to separate different kinds of attacks in splitting. When we use the random split, we can avoid this problem since the data are shuffled.

We randomly sampled half of the former part of the split data as validation data for each day, and the rest was used as training data. In training, the validation data were used for early stopping. After the training stopped, we evaluated the trained classifier using the test set.

Table 9–11 show the results of the evaluations for the time-based split, the count-based split and the random split respectively. These results include the numbers of True-Positive (TP), True-Negative (TN), False-Positive (FP) and False-Negative (FN). They also include metrics often used for binary classification, Accuracy ((TP + TN) / (TP + TN + FP + FN)), Recall also known as true-positive-rate (TP / (TP + FN)), Specificity also known as true-negative-rate (TN / (TN + FP)), Precision also known as positive-predictive-value (TP / (TP + FP)), NPV (negative-predictive-value) (TN / (TN + FN)) and F1-score (2 Recall * Precision / (Recall + Precision)).

These tables imply that our replay data represented some rules which classifiers could learn. For the data of 02/28 and 03/01, the performance achieved by the replay data exceeded the performance achieved by the original data. However, as Table 8 shows, the numbers of positive labels are significantly different between the replay data and the original data. This implies that the replay data includes a lot of positive samples not declared by the original dataset. These differences might affect the results. Table 11 shows that the classifiers could achieve high performance for the replay data and the original data when each of them was shuffled.

VII. CONCLUSION

In this study, we proposed an approach to extract communication flows related to alerts of multiple NIDSes and generated a labeled training dataset. The key point of our approach is to assign high-quality labels to the traffic data by collecting multiple existing NIDS solutions. A comprehensive study was conducted on the proposed extraction approach and the validity of the assigned labels. Furthermore, the labeled training dataset generated using the public dataset and the open-source NIDS, as well as the set of source code, were made available to the public to ensure the reproducibility of this study. We demonstrated the effectiveness of the labels in this generated dataset, trained gradient boosted tree classifiers, and presented the possibility of implementing an AI-powered NIDS model.

Our final aim is to develop an integrated virtual NIDS by training an AI-powered NIDS model using alerts of multiple NIDSes. There are still some challenges in generating the datasets needed for the development. We will attempt to develop advanced training models and adjust the approach of generating labeled data as needed. For instance, how labels are generated must also be considered. Since alert levels and category formats differ among NIDSes, they must be unified by a learning model.

We are studying a cyber threat hybrid analysis platform that integrates various analysis systems and automates a series of security operations [46]. This platform makes it easy for users to promptly get an overview and details of cyber threats. We hope that our proposed approach contributes to the hybrid analysis platform as threat intelligence information that responds quickly to intrusion detection.

ACKNOWLEDGMENT

An earlier of this paper was presented in part at the 16th Asia Joint Conference on Information Security (AsiaJCIS), 2021 [DOI: 10.1109/AsiaJCIS53848.2021.00012]. In this study, some experiments have been updated.

REFERENCES

[1] R. Ishibashi, H. Goto, C. Han, T. Ban, T. Takahashi, and J. Takeuchi, ''Which packet did they catch? Associating NIDS alerts with their communication sessions,'' in *Proc. 16th Asia Joint Conf. Inf. Secur. (AsiaJCIS)*, Aug. 2021, pp. 9–16, doi: 10.1109/asiajcis53848.2021.00012.

[2] K. Scarfone and P. Mell, ''Guide to intrusion detection and prevention systems (IDPS),'' *NIST Special Publication*, vol. 800, no. 2007, p. 94, 2007.

[3] A. S. Ashoor and S. Gore, ''Importance of intrusion detection system (IDS),'' *Int. J. Sci. Eng. Res.*, vol. 2, no. 1, pp. 1–4, 2011.

[4] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, ''Deep learning approach for network intrusion detection in software defined networking,'' in *Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, Oct. 2016, pp. 258–263, doi: 10.1109/wincom.2016.7777224.

[5] F. Feng, X. Liu, B. Yong, R. Zhou, and Q. Zhou, ''Anomaly detection in ad-hoc networks based on deep learning model: A plug and play device,'' *Ad Hoc Netw.*, vol. 84, pp. 82–89, Mar. 2019, doi: 10.1016/j.adhoc.2018.09.014.

[6] S. M. Kasongo and Y. Sun, ''A deep learning method with filter based feature engineering for wireless intrusion detection system,'' *IEEE Access*, vol. 7, pp. 38597–38607, 2019, doi: 10.1109/access.2019.2905633.

[7] C. Yin, Y. Zhu, J. Fei, and X. He, ''A deep learning approach for intrusion detection using recurrent neural networks,'' *IEEE Access*, vol. 5, pp. 21954–21961, 2017, doi: 10.1109/access.2017.2762418.

[8] 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*, vol. 6, pp. 3491–3508, 2018, doi: 10.1109/ACCESS.2017.2782159.

[9] M. A. Ferrag and L. Maglaras, ''DeepCoin: A novel deep learning and blockchain-based energy exchange framework for smart grids,'' *IEEE Trans. Eng. Manage.*, vol. 67, no. 4, pp. 1285–1297, Nov. 2020, doi: 10.1109/TEM.2019.2922936.

[10] M. Nasr, A. Bahramali, and A. Houmansadr, ''DeepCorr: Strong flow correlation attacks on Tor using deep learning,'' in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Jan. 2018, pp. 1962–1976, doi: 10.1145/3243734.3243824.

[11] Y. Zhang, X. Chen, L. Jin, X. Wang, and D. Guo, ''Network intrusion detection: Based on deep hierarchical network and original flow data,'' *IEEE Access*, vol. 7, pp. 37004–37016, 2019, doi: 10.1109/access.2019.2905041.

[12] Y. Zeng, H. Gu, W. Wei, and Y. Guo, ''$Deep-full-range$: A deep learning based network encrypted traffic classification and intrusion detection framework,'' *IEEE Access*, vol. 7, pp. 45182–45190, 2019, doi: 10.1109/access.2019.2908225.

[13] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, ''Disclosure: Detecting botnet command and control servers through large-scale NetFlow analysis,'' in *Proc. 28th Annu. Comput. Secur. Appl. Conf. (ACSAC)*, 2012, pp. 129–138, doi: 10.1145/2420950.2420969.

[14] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, ''Exposure: A passive DNS analysis service to detect and report malicious domains,'' *ACM Trans. Inf. Syst. Secur.*, vol. 16, no. 4, pp. 1–28, Apr. 2014, doi: 10.1145/2584679.

[15] S. Aljawarneh, M. B. Yassein, and M. Aljundi, ''An enhanced J48 classification algorithm for the anomaly intrusion detection systems,'' *Cluster Comput.*, vol. 22, no. S5, pp. 10549–10565, Sep. 2017, doi: 10.1007/s10586-017-1109-8.

[16] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, ''Deep learning approach for intelligent intrusion detection system,'' *IEEE Access*, vol. 7, pp. 41525–41550, 2019, doi: 10.1109/access.2019.2895334.

[17] Y. Li, J. Xia, S. Zhang, J. Yan, X. Ai, and K. Dai, ''An efficient intrusion detection system based on support vector machines and gradually feature removal method,'' *Expert Syst. Appl.*, vol. 39, no. 1, pp. 424–430, 2012, doi: 10.1016/j.eswa.2011.07.032.

[18] E. Hodo, X. Bellekens, E. Iorkyase, A. Hamilton, C. Tachtatzis, and R. Atkinson, ''Machine learning approach for detection of nonTor traffic,'' in *Proc. 12th Int. Conf. Availability, Rel. Secur.*, Aug. 2017, pp. 1–6, doi: 10.1145/3098954.3106068.

[19] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, ''A deep learning approach to network intrusion detection,'' *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018, doi: 10.1109/TETCI.2017.2772792.

[20] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, ''TSDL: A two-stage deep learning model for efficient network intrusion detection,'' *IEEE Access*, vol. 7, pp. 30373–30385, 2019, doi: 10.1109/access.2019.2899721.

[21] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, ''Kitsune: An ensemble of autoencoders for online network intrusion detection,'' in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–15.

[22] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, ''A survey of network-based intrusion detection data sets,'' *Comput. Secur.*, vol. 86, pp. 147–167, Oct. 2019, doi: 10.1016/j.cose.2019.06.005.

[23] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, ''Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study,'' *J. Inf. Secur. Appl.*, vol. 50, Feb. 2020, Art. no. 102419, doi: 10.1016/j.jisa.2019.102419.

[24] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, ''A taxonomy of network threats and the effect of current datasets on intrusion detection systems,'' *IEEE Access*, vol. 8, pp. 104650–104675, 2020, doi: 10.1109/access.2020.3000179.

[25] (1999). *KDD Cup 1999 Data*. Accessed: Mar. 15, 2022. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[26] (1998). *1998 DARPA Intrusion Detection Evaluation Dataset*. Accessed: Mar. 15, 2022. [Online]. Available: https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-eval%uation-dataset

[27] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, ''Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation,'' in *Proc. DARPA Inf. Survivability Conf. Expo. (DISCEX)*, Jan. 2000, pp. 12–26, doi: 10.1109/discex.2000.821506.

[28] T. Chen and C. Guestrin, ''XGBoost: A scalable tree boosting system,'' in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[29] A. L. Buczak and E. Guven, ''A survey of data mining and machine learning methods for cyber security intrusion detection,'' *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016, doi: 10.1109/comst.2015.2494502.

[30] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, ''Toward generating a new intrusion detection dataset and intrusion traffic characterization,'' in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116, doi: 10.5220/0006639801080116.

[31] R. Sharma, R. K. Singla, and A. Guleria, ''A new labeled flow-based DNS dataset for anomaly detection: PUF dataset,'' *Proc. Comput. Sci.*, vol. 132, pp. 1458–1466, Jan. 2018, doi: 10.1016/j.procs.2018.05.079.

[32] R. Sommer and V. Paxson, ''Outside the closed world: On using machine learning for network intrusion detection,'' in *Proc. IEEE Symp. Secur. Privacy*, May 2010, pp. 305–316, doi: 10.1109/sp.2010.25.

[33] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, ''A detailed analysis of the KDD CUP 99 data set,'' in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6, doi: 10.1109/cisda.2009.5356528.

[34] R.-H. Hwang, M.-C. Peng, V.-L. Nguyen, and Y.-L. Chang, ''An LSTM-based deep learning approach for classifying malicious traffic at the packet level,'' *Appl. Sci.*, vol. 9, no. 16, p. 3414, Aug. 2019, doi: 10.3390/app9163414.

[35] E. L. Goodman, C. Zimmerman, and C. Hudson, ''Packet2 Vec: Utilizing Word2 Vec for feature extraction in packet data,'' 2020, *arXiv:2004.14477*.

[36] M. Hassan, M. E. Haque, M. E. Tozal, V. Raghavan, and R. Agrawal, ''Intrusion detection using payload embeddings,'' *IEEE Access*, vol. 10, pp. 4015–4030, 2022, doi: 10.1109/access.2021.3139835.

[37] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Adv. Neural Inf. Process. Syst., 27th Annu. Conf. Neural Inf. Process. Syst. Meeting Held December 5–8, 2013*, Lake Tahoe, NV, USA, 2013, pp. 3111–3119.
[38] (Nov. 2017). *Deutsche Telekom AG Honeypot Project, 'T-Pot 17.10*. Accessed: Jan. 28, 2021. [Online]. Available: http://dtagdev-sec.github.io/mediator/feature/2017/11/07/t-pot-17.10.html
[39] R. Akiyoshi, D. Kotani, and Y. Okabe, "Detecting emerging large-scale vulnerability scanning activities by correlating low-interaction honeypots with darknet," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jul. 2018, pp. 658–663, doi: 10.1109/compsac.2018.10314.
[40] Micro Focus International PLC. (2017). *HPE Security Arcsight Common Event Format Version 25*. Accessed: Mar. 15, 2022. [Online]. Available: https://community.microfocus.com/cfs-file/__key/communityserverwikis-components-files/00-00-00-00-23/CommonEventFormatV25.pdf
[41] IT Security Center, Information-Technology Promotion Agency, Japan. (Jan. 2009). *IPA/ISEC: Vulnerabilities: CVE (Common Vulnerabilities and Exposures) Overview*. Accessed: Mar. 15, 2022. [Online]. Available: https://www.ipa.go.jp/security/english/vuln/CVE_en.html
[42] Y. Kanemoto, T. Shibahara, and M. Akiyama, "Extracting common behavior of SOC analysts for efficient security operation," in *Proc. Comput. Secur. Symp.*, 2020, pp. 645–652.
[43] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Conchran, Z. Durumetric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, J. Ma, Z. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai botnet," in *Proc. USENIX Secur. Symp.*, 2017pp. 1093–1110.
[44] K. Masumi, C. Han, T. Ban, and T. Takahashi, "Towards efficient labeling of network incident datasets using tcpreplay and snort," in *Proc. 11th ACM Conf. Data Appl. Secur. Privacy*, Apr. 2021, pp. 329–331, doi: 10.1145/3422337.3450326.
[45] K. Miyamoto, H. Goto, R. Ishibashi, H. Chansu, T. Ban, T. Takahashi, and J. Takeuchi, "Malicious packet classification based on neural network using kitsune features," in *Proc. 2nd Int. Conf. Intell. Syst. Pattern Recognit.*, Mar. 2022.
[46] T. Takahashi, Y. Umemura, C. Han, T. Ban, K. Furumoto, O. Nakamura, K. Yoshioka, J. Takeuchi, N. Murata, and Y. Shiraishi, "Designing comprehensive cyber threat analysis platform: Can we orchestrate analysis engines?" in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops Other Affiliated Events (PerCom Workshops)*, Mar. 2021, pp. 376–379, doi: 10.1109/percomworkshops51409.2021.9431125.

RYOSUKE ISHIBASHI received the B.E. and M.E. degrees from Kyushu University, in 2020 and 2022, respectively. He has been studying security dataset generation. His research interest includes the understanding of better communication using machine learning.

KOHEI MIYAMOTO received the B.E. and M.E. degrees from Kyushu University, in 2014 and 2018, respectively. He is currently an Academic Researcher at Kyushu University. His research interests include data science and machine learning.

CHANSU HAN received the B.E. degree in computer science and the M.S. and Ph.D. degrees in informatics engineering from Kyushu University, in 2016, 2018, and 2021, respectively. He is currently a Researcher at the National Institute of Information and Communications Technology (NICT), Japan. His research interests include analyzing and solving problems in the cybersecurity field (especially networks and malware) using machine learning.

TAO BAN (Member, IEEE) received the B.E. degree from Xi'an Jiaotong University, in 1999, the M.E. degree from Tsinghua University, in 2003, and the Ph.D. degree from Kobe University, in 2006. He is currently a Senior Researcher with the Cybersecurity Research Institute, National Institute of Information and Communications Technology, Tokyo, Japan. His research interests include network security, malware analysis, machine learning, and data mining.

TAKESHI TAKAHASHI (Member, IEEE) received the Ph.D. degree in telecommunications from Waseda University, in 2005. He was a Researcher with the Tampere University of Technology, from 2002 to 2004, a JSPS Research Fellow with Waseda University, from 2004 to 2006, and a Business Consultant with Roland Berger Ltd., from 2006 to 2009. Since 2009, he has been with the National Institute of Information and Communications Technology, where he is currently a Research Manager. Further, he was a Visiting Research Scholar at the University of California at Santa Barbara, Santa Barbara, CA, USA, from 2019 to 2020. His research interests include cybersecurity and machine learning.

JUN'ICHI TAKEUCHI (Member, IEEE) received the B.Sc. degree in physics and the Dr.Eng. degree in mathematical engineering from The University of Tokyo, in 1989 and 1996, respectively. From 1989 to 2006, he worked with NEC Corporation, Japan. In 2006, he joined Kyushu University, Fukuoka, Japan, where he is currently a Professor of mathematical engineering. From 1996 to 1997, he was a Visiting Research Scholar with the Department of Statistics, Yale University, New Haven, CT, USA. His research interests include mathematical statistics, information geometry, information theory, data science, and machine learning. He is a member of IEICE and JSIAM.