An Effective Feature Extraction Mechanism for Intrusion Detection System

Cheng-Chung Kuo†, Member, Ding-Kai TseNG†, Chun-Wei Tsai†(b), and Chu-Sing Yang†, Nonmembers

SUMMARY The development of an efficient detection mechanism to determine malicious network traffic has been a critical research topic in the field of network security in recent years. This study implemented an intrusion-detection system (IDS) based on a machine learning algorithm to periodically convert and analyze real network traffic in the campus environment in almost real time. The focus of this study is on determining how to improve the detection rate of an IDS and how to detect more non-well-known port attacks apart from the traditional rule-based system. Four new features are used to increase the discriminant accuracy. In addition, an algorithm for balancing the data set was used to construct the training data set, which can also enable the learning model to more accurately reflect situations in real environment.

key words: NetFlow, intrusion detection system, feature extraction, machine learning

1. Introduction

Identifying a method for detecting abnormal network traffic from huge network packets has become a critical topic in the field of network security because of the development of network technology, rapid growth in available bandwidth, and increased hacking skills. However, the mode of network attacks can vary among wired technology [1], wireless technology [2], [3], web [4], apps [5] and so on. The diversity of network attacks renders the detection of abnormal traffic difficult. Firewalls are the simplest solution to filtering out malicious network traffic and controlling the entire topology to further prevent the entire network from being easily attacked. The concept behind first-generation firewall [6] was a packet filter, which examined particular packets and performed given actions such as accept, reject, or drop when capturing specific packets with signatures. The stateful filter firewall [7], [8] can be considered the second generation of firewall that tracks the state of connections. This type of stateful packet inspection monitors all packets and stores the information of transmission states in the dynamic state table to track the connections. Inefficiency and a lack of application-level information are the main disadvantages of a stateful filter firewall.

Use of only some simple technologies, of course, may not be sufficient to filter out malicious events. Deep packet inspection (DPI) provides a superior solution for inspecting deeper into the packet payloads [9] when network managers must monitor traffic in detail to further prevent attacks. The DPI system can be applied to inspection tasks such as limiting the traffic of a special protocol, inspection of known attacks, data leakage prevention, and phishing detection. A simple example can be found in peer-to-peer (P2P) that cannot be detected by port number because of the random port used in P2P connection, but it can be easily detected by a DPI system. However, DPI technologies still have some unresolved problems. The problems with packet-level inspection pertain to performance, encryption, and privacy; DPI is not currently considered a powerful security mechanism. Because DPI inspects the payloads of traffic, if the payloads are encrypted, then the DPI will fail to match the pattern. However, if the payloads are not encrypted, users may be concerned that their private information will be detected while using a DPI system.

An alternative method of detecting abnormal network behaviors is using NetFlow to further develop an intrusion detection system (IDS) or intrusion prevention system (IPS) [10], [11]. A flow typically is defined as a group of continuous packets that have the same source and destination. To be clear, packets are collected at a router, and a flow is defined as packet streams with the same source IP address, source port, destination IP address, destination port, protocol, type of service, and router input interface. When packets pass through the NetFlow exporter (usually, a switch), the NetFlow exporter aggregates the packets with the same attributes (source IP, source port, destination IP, destination port, protocol, and so on) into the same flow.

In the NetFlow information, network managers do not know exactly what the sender transmits to the receiver, but there is a recorded connection between the sender and receiver. The flow information records the total number of packets, total bytes of packets, and duration of this flow. In other words, NetFlow changes units from packets to flows. As a result, NetFlow can easily describe the status of connections in a huge network topology so that anomalous behavior can be detected in the connections. The main contribution of this paper is the provision of a simple but efficient preprocessing mechanism with a machine learning method for the early detection of network attacks. The research process can be summarized as follows.
1. Four new features and extraction methods are used for improving the accuracy of detecting abnormal network traffic and are detailed in Sect. 3.1.

2. To achieve a higher accuracy rate, the proposed undersampling method for retrieving training data sets can provide a balanced training dataset; it is detailed in Sect. 3.2.

3. The proposed detection method can be used in a real campus network environment to reveal the hiding attacks; it is verified in Sect. 4.3.

The remainder of this paper is organized as follows. Section 2 provides a brief introduction to NetFlow-based network attack detection. Section 3 presents the basic idea and details of the proposed system. Section 4 compares the proposed system with other IDSs to compare differences in their performance. Conclusions and recommendations for future research are presented in Sect. 5.

2. Related Work

2.1 Representative Network Attacks

Various network attacks are identified every day, and most security systems encounter difficulty in detecting every type of network attack by using a single method or signature. In [12], [13], the authors categorized network attacks into four types: Denial of Services, Probe, User to Root, and Remote to User. In [14], [15], network scanning was classified into horizontal scan, vertical scan, coordinated scan, and stealth scans. In [16], anomalies in the network, steps of launching network attacks, and how to detect these attacks are described. Network attacks are launched from scanning, brute force attacks, and flooding attacks, which are main concerns in networks [17]. If network managers can detect these network attacks at an early stage, these attacks can be contained. That is why this study used horizontal scan, vertical scan, flooding, and brute force attacks as the representative network attacks, which are discussed as follows.

Horizontal Scan: This attack is well-known as a host sweep attack [18] in which a group of hosts are searched for particular services. There are several reasons for conducting a horizontal scan. For example, botnets perform horizontal scans for searching the vulnerable machines to infect and recruit into the botnet, probing networks for enumeration or penetration.

Vertical Scan: This type of network attack is well-known as a port scan, which is a phase in footprinting and information gathering. A vertical scan aims to find opening ports in a particular system. A vertical scan sends variable packets to the same destination IP with different destination ports and waits for a response. With a response from the destination hosts, the attackers can identify the opening ports or services in the destination hosts.

Flooding Attack: A flooding attack uses a large number of similar flows to exploit the system, such as by sending a large number of small packets or sending less but large size packets to deny the services. In order to achieve better performance, each flooding flow will be triggered in a short period.

Brute force: This attack uses a large amount of username and password information to attempt login. This is similar to flooding flow; however, flooding flow always causes a larger number of flow counts and bytes. Since flooding flow and brute force attack have similar behaviors, we can identify them from the time duration of the attack. Flooding flow requires a large number of NetFlow records to deny the services, so that each flow will be closer together than those of the brute force attack. To hide malicious behaviors in normal traffic, brute force attack may continue for a longer duration to avoid detection.

These four attacks have a property that generates plenty of NetFlow records. Table 1 provides a comparison to explain the statuses of the source IP, destination IP, and destination port of these four attacks.

| Profiles     | Src IP | Dst IP | Dst Port | Duration |
|--------------|--------|--------|----------|----------|
| Horizontal Scan | V      | X      | V        | short    |
| Vertical Scan      | V      | V      | X        | short    |
| Flooding          | V      | V      | V        | short    |
| Brute force       | V      | V      | V        | long     |

V: fixed; X: don’t care

2.2 Intrusion Detection Systems

As mentioned, the use of only firewall and DPI mechanisms to analyze network traffic according to the IP address and port number may not be sufficient for the detection of malicious network behaviors. Additionally, most recent attacks cannot be defended by only configuring the IP address and port number for filtering on the firewall. Therefore, IDS has become a prominent research topic in recent years. Misuse detection and anomaly detection are the two major intrusion detection mechanisms. Misuse detection [19]–[21] is only successful when the attack and vulnerability are understood and can detect intrusion with high accuracy, excluding unknown attacks. However, most traditional IDS systems cannot easily detect new network attacks because attackers can bypass the rules that have been found [22]. If attackers identify the IDS signature rules, then they can encode the malware with shell or encrypt the signature to avoid detection. A popular example of misuse detection is the open-source project Snort [19]. It uses signature-based rules to detect patterns in the packet level and activates an alarm when matched. In addition to signature comparison, methods of expediting the pattern-matching process by using new signature algorithms [20], [21] are important research topics in misuse detection.

Compared with misuse detection, anomaly detection [23] uses another method of detecting attacks that involves recording normal network behaviors and providing a baseline of normal networks. Once the network behavior breaks the threshold, an alarm notifies the manager to
address it. However, this type of detection incurs a high false-positive rate because such a system cannot always easily judge a suitable baseline. Moreover, with anomaly detection, hackers can distribute the packets into normal traffic to evade detection. This anomaly detection benefits from the ability to detect unknown threats but may have a relatively high false-positive rate. Sperotto et al. [24] presented a detailed definition of NetFlow and detection methods for detecting scanning, worm, DDoS, and botnet according to NetFlow information. Methods for detecting anomalous behaviors by using NetFlow is an important research topic in network security [24]–[27]. Many investigations have focused on statistics-based technologies [28], [29]. Moreover, some studies have used information concerning entropy of IP address and port number distribution to detect anomalies [30].

Some events such as scanning cause a specific port number to frequently appear. Statistical methods are a suitable means of judging an abnormal flow from incoming flows. Therefore, some studies have used a Bayesian network to classify Internet traffic [31], [32]. In [32], a multi-layer filter was used to improve the performance of the Bayesian network to further detect user-to-root (U2R) and remote-to-local (R2L) attacks. Francois et al. [33] and Maryam et al. [34] used aggregated NetFlow records to detect large-scale network anomalies.

### 2.3 Machine Learning Based Approach

Because attackers employ statistical methods to avoid detection, some studies have used machine learning technologies to enhance the performance of IDS, which can be divided into three types: supervised, unsupervised, and semi-supervised.

- **Supervised**: This method uses a labeled training data set as a basis for classifying unknown incoming data. It is efficient for detecting known network attacks because if the data set is sufficiently large, then the methods can be well trained. Kalaivani and Vijaya [35] employed supervised learning techniques to detect botnet traffic by using a CTU-13 data set. In another study [36], Zhao et al. used naïve Bayesian classification, SVM, and a decision tree to achieve real-time detection in a data center.

- **Unsupervised**: The basic aim of this method is to find patterns or features in unlabeled data. For example, Leung and Leckie [37] used feature vectors collected from a network to detect intrusions. In another study [38], Hochst et al. used statistical properties of flows and a neural autoencoder to classify the network traffic from file operations, websites, videos, live streams, and interactive communications.

- **Semi-supervised**: This type of method is a combination of supervised and unsupervised methods. This means that semi-supervised methods are trained with both labeled and unlabeled data sets because labeled data sets cannot be easily obtained and well labeled. The training for semi-supervised methods uses fewer labeled data sets combined with unlabeled data. In [39], Liu and Xu used a naïve Bayes training model to train the data set from nProb. In another study [40], Ashfaq et al. employed a fuzzy algorithm and neural network with random weights for IDS.

### 2.4 Sampling Methods for Imbalance Dataset

In many domains, one critical issue is how a better data set for machine learning can be obtained [41]–[44]. Collected NetFlow records are also imbalanced in real network traffic—with its variety of applications, services, and attacks [45]. Thus, addressing this imbalance in NetFlow records is also crucial to increasing detection rate. Methods for handling the imbalance data set can be grouped into oversampling and undersampling variants.

**Oversampling**: An easy approach to addressing imbalance in the data set is to increase the size of the minority to balance the distribution. Random oversampling (ROS) [46] is an example of such an approach, where it randomly replaces the majority set to generate a balanced data set. In [47], the authors proposed a synthetic minority oversampling technique (SMOTE) to create synthetic data with k-nearest neighbors (kNN). In [48], the authors discussed the small class problem in imbalanced classification, and they constructed a data set using synthetic data to address oversampling.

**Undersampling**: Another efficient way to address imbalance in a data set is to use only a part of the majority class rather than all of it. Random undersampling (RUS) [46] is an example of such an approach, where it ignores data from random selection. However, potential information may be randomly lost in the randomly generated data.

Another way to solve the problem of imbalance in the data set is by modifying the algorithms to fit the data set. In [49], the authors proposed a modified proximal support vector machine (MPSVM) by adding a matrix to optimize the decision function. In [50], the authors proposed a modified random forest (RF) for replacing the majority set with a minority set. Other studies on feature selection have also contributed to solving imbalance in the data set. For example, Chen et al. proposed a feature assessment by sliding thresholds (FAST) method based on the ROC curve, generated by moving the decision boundary of a single feature classifier [51]. In [52], the authors formulated a different filtering method for extracting features from a high imbalanced data set. Zhu and Wang [53] used the entropy to evaluate the class certainty of a pattern and evaluated their method against its counterparts.

In summary, by comparing traditional methods (e.g., firewall or string matching of DPI) for detecting the attacks, several studies have demonstrated that machine learning and data mining technologies can provide an alternative means of enhancing the performance of IDS, not only to improve accuracy but also to reduce computation costs. Because
using only machine learning may not always be suitable for IDS to detect the attacks, how to design an effective IDS or IPS by using such technologies for the real network system has become a relatively popular research topic in the last decade.

3. The Proposed System

This study used a machine learning method to achieve near real-time preprocessing with relatively high accuracy of misuse detection. This section details the concepts used in this study and how they are combined in the proposed system. The process of NetFlow traffic detection in the system is mainly divided into three parts: data set generation, off-line training, and online testing. As shown in Fig. 1, after data set generation and offline training have been completed, online testing is performed periodically to detect abnormal traffic. The following section introduces the system-related algorithm and architecture in detail.

Data set generation and offline training: In the proposed model, a better data set for training can achieve higher detection accuracy. This study used time intervals and a training data balancing mechanism to generate a better training set. In this stage, NetFlow records are classified into several types, such as vertical scan, horizontal scan, flooding flow, brute force, and normal. The classified NetFlow records are labeled according to the network types and added into the training set. New features used in this study are used for finding similar connections with the same correlated IP and port. For example, horizontal scanning attacks have many connections with the same Src IP and Dst IP with the same Dst Port. However, this type of information is not recorded as one type of NetFlow information. The new feature extraction method is described in Sect. 3.1. This paper provides an algorithm based on the entropy of the data sets to deal with the imbalanced network attacks and benign traffic. A detailed description is presented in Sect. 3.2.

Online Testing: While offline training is being completed, the unlabeled incoming NetFlow raw data are retrieved from routers. When the NetFlow records are retrieved from a router or storage, the unlabeled NetFlow records enter the trained model and judge the incoming NetFlow records to determine which network type they belong to. There is a time delay because the NetFlow records are stored while they wait for a time slot to enter the Detection module. The response time of detecting network attacks is restricted in the next time slot. In NetFlow, a flow record will be recorded after the connection is ended. If a connection started at time slot $T_1$ and ended in time slot $T_3$, the record will be recorded in $T_3$; however, active flow will not be recorded. According to our observations, this information will be very useful in allowing the mechanism to detect attacks. According to the detection results, a report is generated, which notifies the network managers what type of network attack they may have faced. The report specifies the victim’s IP address to enable inspection of vulnerability of the system as well as the attacker’s IP address to enable it to be blacklisted. Moreover, this study proposed a training data balancing algorithm and new feature extraction method for achieving real-time detection and a higher accuracy rate. Section 4 provides descriptions of online testing.

To simplify the discussion that follows, the following

![Fig. 1 Flowchart of the proposed model.](image-url)
notation is used throughout the remainder of the paper.

$C(s)$ process of counting the actual items in the set $s$.

$F_x$ set of original items of feature $x$ in the tentative training data.

$T_x$ threshold of feature $x$.

$E(s)$ Shannon entropy of set $s$.

$S(s)$ Shannon entropy of selected set $s$.

$N(s)$ function representing the amount of its item, for example, $N(F_x)$ means the quantity of $F_x$.

$A$ amount decided for selection in one round

$TP$ count of true positives in the experiments.

$FN$ count of false negatives in the experiments.

$TN$ count of true negative in the experiments.

$FP$ count of false positives in the experiments.

$\gamma$ how much influence that extreme or infrequent values have on the training data.

$P_r$ precision rate that can be calculated by $P_r = \frac{TP}{TP + FP}$.

$R_r$ recall rate that can be calculated by $R_r = \frac{TP}{TP + FN}$.

$\alpha_x$ accuracy rate when applying the feature $x$.

$Ir$ influence ratio while apply different feature comparing with original features. $Ir = \frac{\alpha_{feature}}{\alpha_{original}}$

### 3.1 New Feature Extraction

Network attacks are usually consistent and can be observed only when the information of the feature is sufficient to give coherence to these independent NetFlow records. For example, in Fig. 2, the source IP sends $N$ flows to different destination IPs with the same port number to achieve a wide scanning range. However, in the original NetFlow data obtained from the Cisco device, these $N$ flows are treated as different records. This characteristic moderates the relationship between these NetFlows. Thus, the relevance of these flows is crucial when we need to distinguish network attacks from normal traffic. Furthermore, although the IP address in each record provides some information for identification, it is not meaningful when it comes to the machine learning method; therefore, we drop these two features (source IP, destination IP) but use alternatives to maintain the relation between these IP connections. Therefore, this paper proposes four new features to enhance the detection rate on the basis of the original NetFlow information.

As indicated in Table 2, this study used a set of features from NetFlow, and four new features that were extracted from the original netflow information: $Dst_{IP\_CNT}$, $Dst_{Pt\_CNT}$, $Conn$, and $Direction$.

$Dst_{IP\_CNT}$ is the correlated flows in a time slot with tuples aggregation with the same $(srcip, dstport, proto)$, as shown in Fig. 3. First, the flow records are sorted and grouped by their $Src\_IP$ and $Dst\_Pt$. Then the number of flows is calculated according to the pair $(Src\_IP, Dst\_Pt)$. The using of sorted lists and groups operation can achieve higher performance than aggregation as presented in Algorithm 1.

$Dst_{Pt\_CNT}$ is the correlated flows in a time slot with tuples aggregation with the same $(srcip, dstport, proto)$. The generation algorithm of $Dst_{Pt\_CNT}$ is shown in Algorithm 2. These two features are used for representing the overall connection relation of NetFlows in a testing time slot.

$Conn$ represents the geographic information of the connection. This study defined four connection types with One-Hot Encoding as shown in Table 3. In the first one, the source and the destination are within the same nation. In the second type, the connection is not within the same nation. The third type is private or unusual and should not appear in a public network. The last one is connection unknown, which means that the geographic information does not appear in the databases.

$Direction$ uses One-Hot encoding for categorical variables as shown in Table 4 and has two types: the first one is
the source inside the LAN/WAN, which refers to a connection from inside the network to the outside. The other one is the destination inside the LAN/WAN, which means that the connection is from the outside to the inside. In a network attack, the direction is an important feature for determining where the attack started.

These four features can increase the depth and breadth (dimension) of the training data set when transforming IDS detection from a rule-based to a machine learning–based method.

3.2 Training Data Balance

In the real network environment, the volume of normal traffic (higher than 90%) is usually overwhelming compared with that of malicious traffic. Moreover, the proportion of each network attack varies for different time slots. In the machine learning training model, the large difference in volume among each label of training data causes the model to be malformed, and the result is poor. In extreme cases, if the training data does not include the brute force attack, it causes a testing stage fail or influences the accuracy of the result because this type of attack has never appeared in the training data set before. In order to balance the training data set and select a more ideal distribution of data, this paper proposes a mechanism to remove similar NetFlow records and a sampling method to first address the training data set. The proposed mechanism can be divided into three parts: 1). malicious data balancing, 2). malicious data sampling, and 3). normal data sampling.

**Malicious data balancing** is used to collect the malicious NetFlow records in the given time slot. This module will identify all the attacks in the training data and in the first time slot; it will not contain the same number of instances for each attack to prevent imbalance in the data set. If the malicious NetFlow records are still insufficiently balanced, then the module traces back to the last time slot to collect data until the malicious training data is balanced according to the user’s previously defined settings. As illustrated in Fig. 4, if the module cannot identify a sufficient number of brute force attacks in the time interval \( I_1 \), then the module will try to identify brute force attacks in \( I_2, I_3, \ldots \) until the requisite number of attacks is reached.

**Malicious data sampling** is the main procedure for collecting malicious training data, and the data balancing check is also included in this stage. According to Fig. 5, for every time slot at which we trace back to select the malicious training data, it is necessary to first filter out the normal data and

---

**Table 3** Conn with One-Hot encoding

| Conn          | The same nation | Not the same nation | Private | unusual |
|---------------|-----------------|--------------------|--------|--------|
| The same nation | 1               | 0                  | 0      | 0      |
| Not the same nation | 0           | 1                  | 0      | 0      |
| Private       | 0               | 0                  | 1      | 0      |
| Unusual       | 0               | 0                  | 0      | 1      |

**Table 4** Direction with One-Hot encoding

| Direction       | LAN/WAN → outside | Outside → LAN/WAN |
|-----------------|--------------------|-------------------|
| LAN/WAN → outside | 1                | 0                 |
| Outside → LAN/WAN | 0                | 1                 |
then distinguish the NetFlow data on the basis of the source IP. For every source IP, we only randomly select one flow and put it into the malicious training data set until the number of training data of each attack meets the given number. Notably, there is a restriction in this step because many flows of attacks have the same source IP. For example, if IP A is a malicious device, then the flows coming from this IP may present a similar status; thus, acquiring too many training data from the same source IP may result in skewing of specific types, and the types will be highly sensitive in the testing step. In order to overcome this problem, we only obtain one flow per source IP in one round with random selection; moreover, every time we obtain a new flow for the training data, we also must check whether this source IP appears in the training data set over a specific ratio. If one IP appears too often and exceeds the threshold, then the flows coming from this IP will be ignored. We propose this method to increase the diversity of malicious training data and make it more balanced and robust.

Normal Data Sampling is used for the selection of normal training data. Generally, random selection or sampling can be used in a specific sequence to eliminate this bulk of normal traffic, and this is also the main goal to accomplish. However, if the normal training data are selected randomly or sequentially without any restriction, then they are not stable and effective because the content of the features might be in an uncertainly distributed state, meaning that it cannot be guaranteed that the diversity of the data is sufficiently high. Therefore, an additional sampling process is proposed. As shown in Fig. 6, normal data are randomly sampled in the first step. Selected data are evaluated to determine whether they are sufficiently diverse for depicting normal traffic. In this part, some of the features of the original data set are divided into two groups according to type, and the proposed method operates on the basis of these groups. Definitions of the two types are as follows:
Type I: Features with low diversity and of which there exists some rare value throughout the data set.

Type II: Features with high diversity and of which the value distribution presents an apparent tendency throughout the data set.

Depending on the data set and the features themselves, in this research, we select Proto, Conn, Dst_IP_CNT, and Dst_Pt_CNT to prevent the randomly sampled data from being too homogeneous and biased. Every round in which the system randomly samples a new data set of normal data, the data should be examined by sequential rules. The mechanism proceeds to sample the data randomly until the sampled data set is certified.

The features Proto and Conn, belonging to I, both contain some rare value and only have a few possibilities. Proto features, which usually are TCP, UDP, ICMP, that is, one identified in every 10 minutes of processing NetFlow data, seldom follow protocol such as BGP but do on occasion. For this reason, it is a problem when it turns to the online testing state if the training data do not contain any of these exceptions. Conn features, which comprise only four types, have the same situation as Proto features. Therefore, imposing some restrictions when sampling is imperative, but one cannot overinfluence the random sampling process without rendering the entire sampling process meaningless. In I, the constraints are as follows:

\[ C(S(F_x)) > T_x \quad \& \quad E(S(F_x)) > E(F_x), \]  

where \( C(\cdot) \) represents the process of counting the actual volume of different items appearing in a set. \( F_x \) is the set of original items of feature \( x \) in the tentative training data. \( T_x \) denotes the number of threshold types in the feature \( x \).

One manually given threshold is needed for determining the lowest number of values per category that should be contained in the specific feature for the tentative training data. For instance, for \( C(F_{Proto}) > T_{Proto} \), the focus is on feature Proto in the set of the randomly selected training data in every round. When the number of items of feature Proto in the selected data exceeds the threshold, it is sufficient in the category, but whether it is sufficiently diverse for the entire set of feature Proto still cannot be verified because even though the number of categories may be sufficient for covering most real-world testing scenarios, it may still become extremely tilted in some cases, and according to the experiment, this will cause some false-positive and false-negative results in the margin of the testing data. Therefore, an appraisal method is also utilized in this process, Shannon entropy, which is defined as follows:

\[ -K \sum_{i=1}^{n} P_i \log(P_i), \]  

where \( P_i \) is the probability that the group is in the set and \( K \) is a constant. Proceeding with feature Proto, \( E(S(F_{Proto})) > E(F_{Proto}) \), which means that the entropy of feature Proto in the selected data \( S(F_{Proto}) \) is greater than that of the original data \( F_{Proto} \). Entropy can be easily interpreted as a value that represents the randomness of a set. If a group of numbers or items achieve a relatively high entropy value, then this means that the set is more complicated and diverse. According to the formula of Shannon entropy, the entropy value can be influenced by two factors, namely the sum of categories and repetitions for each category. The sum of categories is controlled by the first rule, and under this condition, entropy can be easily considered to be in charge of repetitions of each category. According to the two examinations, the selected training data will be treated as qualified for the condition of feature Proto.

\( Dst_{IP_CNT} \) and \( Dst_{Pt_CNT} \) have relatively wide ranges in value and usually have a high tendency, as mentioned; therefore, a solution for addressing the uncertainty in data sampling is limiting the volume ratio to extreme and uncommon values. The rules are listed as follows:

\[ N(\forall f_x \in S(F_x) > F_x(N(F_x) \times 3/4)) < A \times \gamma, \]  

where \( \gamma = \{x \in R \mid 0 \leq x \leq 0.25\} \) and \( N(\cdot) \) represents the amount of an item; for example, \( N(F_x) \) refers to the quantity of \( F_x \). One can simply follow the rules, such as not obtaining a value that is too extreme in one selection, and the amount is controlled by \( A \) and \( \gamma \). \( A \) is the amount that is decided for selection in one round, and \( \gamma \) represents how much influence the extreme or uncommon value has on the training data. In this part, the third quartile is used in acquiring the boundary of each original data; after all, the values of specific features in a sampling process must not exceed the specified ratio selected in the round. If the data sampled this time do not pass the test, then it must be resampled by following the other steps in the sampling process. Once it passes verification, the data maintain both their flexibility and conservative characteristics.

By using the aforementioned automatic selection method, this research may retrieve a balanced and genuine data set to use for training in a dynamic and real-world network environment.

4. Experimental Results

The proposed intrusion detection system was deployed in the campus network, and the results were compared with those of a generic statistical intrusion detection system. This study evaluated the proposed method according to score time, fit time, precision, confusion matrix, and online testing.

4.1 Experimental Environment and Parameter Settings

The proposed model was designed on the basis of NetFlow records, which are exported from the router in campus networks. The router, Cisco 7609, was configured to export the traffic in NetFlow v9 format. The packet header of the traffic flowing through the router is parsed and aggregated to generate the data in NetFlow format and then streamed to the NetFlow collector where it is configured for storage, as shown in Fig. 7. Information regarding the deployed
The experimental environment.

Table 5  System information of the experimental environment

| Name              | type                                      |
|-------------------|-------------------------------------------|
| CPU               | Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz   |
| Operation System  | Ubuntu 16.04.3 LTS                        |
| Kernel version    | 4.4.0-134-generic                        |
| Memory            | DIMM DDR3 Synchronous 1600 MHz 4G x 4     |
| Programming Language | Python 3.5.2                        |
| Machine Learning Package | sklearn 0.19.1          |

Table 6  Parameter settings

| Methods | Parameters                        |
|---------|-----------------------------------|
| kNN     | Number of neighbors: 5            |
| RF      | Max depth: 5                      |
| NN      | Number of epochs: 1000            |
| RNN     | Train on batch: 1000              |
| XGB     | Max depth: 5                      |

Nfdump [55] was used to manage the NetFlow records. Nfdump has many functions, one of which is nfcapd, which is used to convert the NetFlow stream into files for storage in network attached storage (NAS). The cycle time for generating the new NetFlow files in the experiment was one minute. The server mounts the NAS, which contains the NetFlow files, and the proposed model dumps these files into a readable string through the nfdump function to perform feature extraction and append new features. The time slice in the experiment environment was set to 5 minutes.

Parameters for machine learning methods are presented in Table 6.

4.2 Data Sets

The training model used in the evaluation was XGBoost (eXtreme Gradient Boosting, XGB) [56]. XGB is a training model with relatively short training time and high accuracy. In the evaluation, Python was used to implement the XGB model. NetFlow records were collected from Mar 10, 2020 to Mar 16, 2020 in the Department of Electrical Engineering of National Cheng Kung University. The evaluation consisted of four types of network attacks and normal traffic for training the machine learning model. Horizontal scans, vertical scans, flooding flow, and brute force attacks were detected according to the statistical threshold, and the NetFlow records were labeled according to the statistical threshold. The features used in the evaluation are as follows: duration, protocol, destination_port, packets, Bytes, Flows, byte_per_second, packet_per_second, and byte_per_packet. The new proposed features DstIP_CNT, DstPt_CNT, Conn, and Direction are also included in the evaluation. Basically, according to the campus network, the threshold of each attack was set to 15 scans in 5 minutes. Because network traffic varies rapidly, this threshold also varies among different campus networks. This study evaluated the proposed model against the KNNs [57], RF [58], NN [59], recurrent NN (RNN) [60], and XGB methods with respect to performance in detecting malicious traffic.

4.3 Experimental Results

Score time: Score time is used for training the generated data set. The required time for training a model is indicated in Fig. 8. In the proposed model, score time is used for training offline, which is performed in the first experiment rather than in every experiment. Moreover, the training model requires less time in RNN and XGB with the proposed features. This means that RNN and XGB (based on the decision tree) is more suitable with these new features. The proposed new features can help to converge the RNN and XGB algorithm in the training stage. Although the score time for XGB is much higher than it is in other methods, the training stage is completed offline and can reduce the impact of score time. NN and RNN had a better score time but required epochs to train a better model. In the experiments, the training stages for NN and RNN were set to terminate at 1000 epochs.

Fit time: Fit time is used for measuring the time required to run the cross-validation trip. As shown in Fig. 9, XGB still had more fit time to complete cross-validation. However, NN and XGB with the proposed features require less time for running the cross-validation trip than with the
original features. The proposed new features can reduce the time required for the iteration stage of the XGB method to reduce the fit time to lower than that without new features. However, kNN, RF and RNN increase the fit time when using new features. NN and RNN took longer to finish the fit stage. Nevertheless, fit time in this study was not a critical criteria since the fit time is used for off-line training. It will not effect the on-line processing while applying the proposed system.

**Precision rate:** Precision rate ($P_r$), also called positive prediction rate, indicates the quality of machine learning methods. As presented in Fig. 10, XGB can achieve the highest precision rate among all of the machine learning methods. In the random forest method, the training stage involves fetching training data and training them into several trees. However, the precision rate decreases if the training data are skewed in toward any set of labelled network attack types. For all the machine learning methods considered in this paper, the addition of the new features increased the precision rates compared with those obtained using the original features.

**Recall rate:** Recall rate ($R_r$) is the proportion of data in the labeled set that are successfully retrieved from the data set. As displayed in Fig. 11, XGB can achieve the highest recall rate among all the other machine learning methods included in this paper. Additionally, a higher recall rate was obtained when using the proposed features. The random forest method had the lowest recall rate because it was skewed toward real traffic.

**Receiver Operating Characteristic (ROC) and Area Under the Curve (AUC):** As the ROC and AUC are usually used for evaluating the performance of a classification method, we used them to further interpret the performance of the proposed method [61]. Note that the ROC curve is a graph that shows the probability of a true positive rate versus a false positive rate. AUC values, which display a better classifier, are typically close to 1.0. As shown in Fig. 12, XGB has the best AUC value compared with other algorithms, such as NN, RNN, RF, and kNN.

**Sampling Method Influence:** This paper proposes a training data set sampling method to address imbalance in the training data set. The sampling method can improve the training set with improved diversity. As indicated in Table 7, XGB had the highest precision rates compared with the proposed sampling method. The experimental results indicated that of all methods, the proposed sampling method increased precision. Compared with RUS, proposed

| Method               | without sampling | RUS | Entropy based |
|----------------------|------------------|-----|---------------|
| Precision rate of kNN| 82.17%           | 64.49% | 67.55% |
| Recall rate of kNN   | 82.94%           | 67.29% | 62.26% |
| Precision rate of RF | 73.59%           | 77.48% | 73.45% |
| Recall rate of RF    | 76.5%            | 74.52% | 61.25% |
| Precision rate of NN | 90.55%           | 85.01% | 94.64% |
| Recall rate of NN    | 81.5%            | 75.11% | 94.77% |
| Precision rate of RNN| 73.6%            | 82.39% | 95.29% |
| Recall rate of RNN   | 69.71%           | 82.21% | 95.59% |
| Precision rate of XGB| 96.23%           | 81.85% | 99.53% |
| Recall rate of XGB   | 96.29%           | 95.29% | 93.75% |
Table 8  Accuracy for attacking tools

| Attack types      | kNN  | RF   | NN   | RNN  | XGB  |
|-------------------|------|------|------|------|------|
| Vertical scan     | 99.89% | 85.45% | 95.2% | 97.2% | 92.78% |
| Horizontal scan   | 85.77% | 84.5%  | 85.2% | 85.5% | 84.74% |
| Brute force (SSH) | 75.77% | 98.23% | 99.98% | 99.98% | 98.23% |
| Brute force (web) | 98.91% | 99.89% | 99.7%  | 99.9%  | 99.88% |
| SYN flooding      | 96.01% | 90.02% | 99.99% | 99.99% | 99.97% |
| UDP flooding      | 72.04% | 91.02% | 99.98% | 99.99% | 99.98% |

Table 9  Confusion matrix of online testing

|          | Normal | Vertical | Horizontal | Brute | Flood |
|----------|--------|----------|------------|-------|-------|
| Normal   | 3632060 | 33788    | 92139      | 4124  | 8879  |
| Vertical | 500    | 43709    | 0          | 0     | 0     |
| Horizontal | 6230   | 0        | 54569      | 3     | 0     |
| Brute    | 17     | 0        | 0          | 591   | 0     |
| Flooding | 0      | 0        | 0          | 0     | 2116  |

 entropy-based sampling method yielded better accuracy and recall rate. Interestingly, data sets constructed without any sampling method applied can sometimes yield greater accuracy compared with those with RUS or entropy-based sampling method applied. Because of the variety in real network traffic and the similar behavior of attacks of different types—such as scanning, brute force or flooding—in a given period, if the sampling method picks similar attack records for the data set, the machine learning method may suffer from overfitting in the training stage. However, proposed method attempts to construct the new data set with respect to entropy, thus ensuring that the data are robust to various attacks.

Attacking tools testing: This experiment was set in a closed network topology to verify the accuracy while applying attacking tools in the network. Vertical scan and horizontal scan attacks were made by nmap [62], [63] from 5 hosts. Brute force attack for SSH was made by hydra [64], [65] while web was by burpsuit [66] and flooding flows were made by hping [67], [68]. Duration of each attack was set to 15 minutes. The accuracy is presented in Table 8. Horizontal scan has the lowest detecting rate for each method in this experiment, since horizontal scan will need to aggregate the flows to generate the features. The number of hosts will affect the result while applying the horizontal scan attacking tool. The other attacks only need fewer attackers or victims to achieve the classification.

Online testing: This experiment was set to the real network environment that operated in a real campus network in order to verify the accuracy of this approach in the real world. The confusion matrix of XGB is presented in Table 9. Duration of experiments was set from 16 Mar 2020 to 22 Mar 2020. In the experiment, there were 3770990 flows, and the accuracy rate was approximately 96.45%. The ground truth of these attacks is based on the statistical threshold. Therefore, if the attacks did not exceed the threshold, then the traffic was considered to be normal. In the real environment, TP is 3632060, FP is 33788 + 92139 + 4124 + 8879 = 138930. The precision rate for the proposed system was \( \frac{3632060}{3632060 + 138930} = 0.9631 = 96.31\% \). When further analyzing NetFlow records that were not matched in the proposed method, the remaining 145680 (33788 + 92139 + 4124 + 8879 + 500 + 6230 + 17 + 3) flow records were more important for improving the accuracy.

A deeper look into these unmatched flows reveals that these flows were composed of DNS queries (1283 records), suspicious remote access events (20120 records), stealthy scanning (29852 records), and unknown services or suspicious (94422 records). The stealthy scanning IPs could act as a means of evasion to enable hackers to avoid statistical detection. However, in the proposed method, the new features could correlate to the flows with similar information so that the proposed new features could identify the hidden threats from normal flows. The DNS queries in the experiment are viewed as normal queries; a good white list can help considerably in reducing the mismatch rate for these queries. The unknown services are services that act as network attacks; in this study, these services needed to be monitored or converted into digital forensics for advanced detection.

5. Conclusion

This research developed an efficient IDS by extracting profiles from network traffic in NetFlow format in almost real time with high accuracy by using XGB. The use of the proposed four new features can obtain higher accuracy than the original detection method. Additionally, the proposed sampling method can obtain a balanced training data set from imbalanced NetFlow records as well as improve accuracy. To understand the performance of the proposed system, this study collected real campus network traffic data as the data source. NetFlow records were grouped into four network attacks, and the trained model was used to detect new incoming and unlabeled NetFlow records. According to the evaluation results, the proposed model can achieve an accuracy of approximately 96.31% on average and detect network attacks in almost real time. The attacks of various types, such as the use of bots or malware, are present on the Internet. These attacks require layered defenses to detect—such as a host log collection for verifying host behavior. Our proposed model is the first layer of defense used to gather data on suspicious behavior in the network. Future studies can address certain topics in order to help improve this research. One is that each feature of network attacks can be further fine-tuned to obtain more precise results, such as threshold and filter. The experimental results of this research reveal that accuracy varies according to the labeling algorithm used. Therefore, the development of a more effective feature extraction method for detecting new or different network attacks can increase the number of detectable network attack types and even improve accuracy. Furthermore, the clustering method can be applied in future research to group together different network services or attacks to generate new network attack types. Consequently, the trained model can maintain a relatively high precision and recall rate in the first several days. The development of a retraining mechanism or
application of reinforcement learning constitute the next research steps that ought to be taken.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions on the paper. This work was supported in part by the Ministry of Science and Technology of Taiwan, R.O.C., under Contracts MOST 109-2221-E-006-167 and MOST109-2218-E-006-014.

References

[1] J.M. Estevez, G.T. Pedro, and E.D. Jesus, "Anomaly detection methods in wired networks: A survey and taxonomy," Computer Communications, vol.27, no.16, pp.1569–1584, 2004.
[2] P.M. Jawandhiya, D. Ghonge, and M.S. Ali, "A survey of mobile ad hoc network attacks," International Journal of Engineering Science and Technology, vol.2, no.9, pp.4063–4071, 2010.
[3] D.G. Padnavathi and M. Shanmugapriya, "A survey of attacks, security mechanisms and challenges in wireless sensor networks," International Journal of Computer Science and Information Security, vol.4, no.1, pp.4063–4071, 2009.
[4] D.R. Patil and J.B. Patil, "Survey on Malicious Web pages Detection Techniques," International Journal of U- and E-Service, Science and Technology, vol.8, no.5, pp.195–206, 2015.
[5] P. Faruki, A. Bharmal, V. Laxmi, V. Gannoor, M.S. Gaur, M. Conti, and M. Rajarajan, "Android security: A survey of Issues, Malware Penetration, and Defenses," Commun. Surveys Tuts., vol.17, no.2, pp.998–1022, 2015.
[6] K. Ingham and S. Forrest, "A history and survey of network firewalls," University of New Mexico Tech. Rep. pp.1–42, 2002.
[7] F. Avolio, "Firewalls and Internet security, the second hundred (Internet) years," The Internet Protocol Journal, vol.2, no.2, pp.24–32, 1999.
[8] M.G. Gouda and A.X. Liu, "A model of stateful firewalls and its properties," Proc. Conf. on Dependable Systems and Networks, Yokohama, Japan, pp.128–137, June 2005.
[9] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood, "Deep packet inspection using parallel bloom filters," Proc. on High Performance Interconnects, Stanford, CA, USA, pp.44–51, Aug. 2003.
[10] F. Sabahi and A. Movaghar, "Intrusion detection: A survey," Proc. Conf. on Systems and Networks Communications, Boston, pp.23–26, Oct. 2008.
[11] M.F. Umer, M. Sher, and Y. Bi, "Flow-based intrusion detection: Techniques and challenges," Computers & Security, vol.70, pp.238–254, 2017.
[12] K.K.R. Kendall, "A database of computer attacks for the evaluation of intrusion detection systems," PhD thesis, Massachusetts Institute of Technology, 1999.
[13] M. Ahmed, A.N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," Journal of Network and Computer Applications, vol.60, pp.19–31, 2016.
[14] V. Vigneswaran, P. Barford, and J. Ulrich, "Internet intrusions: Global characteristics and prevalence," ACM SIGMETRICS Performance Evaluation Review, vol.31, no.1, pp.138–147, 2003.
[15] S. Stanford, J.A. Hoagland, and J.M. McAleney, "Practical automated detection of stealthy portscans," Journal of Computer Security, vol.10, no.1, pp.105–136, 2002.
[16] N. Hoque, M.H. Bhuyan, R.C. Baishya, D.K. Bhattacharrya, and J.K. Kalita, "Network attacks: Taxonomy, tools and systems," Journal of Network and Computer Applications, vol.40, pp.307–324, 2014.
[17] S.T. Zargar, J. Joshi, and D. Tipper, "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks," Commun. Surveys Tuts., vol.15, no.4, pp.2046–2059, 2014.
[18] O. Al-Jarrah and A. Arafat, “Network intrusion detection system using neural network classification of attack behavior,” Journal of Advances in Information Technology, vol.6, no.1, 2005.
[19] Snort. (2020) Snort - network intrusion detection & prevention system https://www.snort.org, accessed 9 Nov. 2020.
[20] N. Casarano, L. Ciminierra, and F. Risso, “Improving cost and accuracy of DPI traffic classifiers,” Proc. Conf. on Applied Computing, Sierre, Switzerland, pp.641–646, March 2010.
[21] W. Jiang, Y.-H.E. Yang, and V.K. Prasanna, “Scalable multi-pipeline architecture for high performance multi-pattern string matching,” Proc. Conf. on Parallel & Distributed, Atlanta, GA, USA, pp.1–12, April 2010.
[22] E. Gandotra, D. Bansal, and S. Sofat, “Malware analysis and classification: A survey,” Journal of Information Security, vol.6, no.2, pp.5–18, 2014.
[23] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, “Anomaly-based network intrusion detection: Techniques, systems and challenges,” Computers & Security, vol.28, no.1, pp.18–28, 2009.
[24] A. Sperotto, G. Schaffrath, R. Sadre, C. Morarit, A. Pras, and B. Stillier, “An overview of IP flow-based intrusion detection,” Commun. Surveys Tuts., vol.12, no.3, pp.343–356, 2010.
[25] R. Hofstede and A. Pras, “Real-time and resilient intrusion detection: A flow-based approach,” Proc. Conf. on Autonomous Infrastructure, Management and Security, Berlin, Heidelberg, pp.109–112, June 2012.
[26] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, “Flow monitoring explained: From packet capture to data analysis with netflow and ipfix,” Commun. Surveys Tuts., vol.16, no.4, pp.2037–2064, 2014.
[27] M.H. Bhuyan, D.K. Bhattacharrya, and J.K. Kalita, “Network Anomaly Detection: Methods, Systems and Tools,” Commun. Surveys Tuts., vol.16, no.1, pp.303–336, 2014.
[28] S. Faraiposo, P. Owezarski, and E. Monteiro, “Contribution of Anomalies Detection and Analysis on Traffic Engineering,” Proc. Conf. on Computer Communications, Barcelona, Spain, pp.1–2, April 2006.
[29] H.A. Nguyen, T. Van Nguyen, D.T. Kim, and D. Choi, “Network traffic anomalies detection and identification with flow monitoring,” Proc. Conf. on Wireless and Optical Communications Networks, Surabaya, Indonesia, pp.1–5, May 2008.
[30] A. Lakhina, M. Crovella, and C. Diot, “Mining anomalies using traffic feature distributions,” Proc. Int. Conf. Applications, Technologies, Architectures, and Protocols for Computer Systems, vol.35, no.4, pp.217–228, 2005.
[31] A.W. Moore and D. Zuev, “Internet traffic classification using bayesian analysis techniques,” Proc. Conf. on Performance Evaluation Review, Alberta, Canada, pp.50–60, June 2005.
[32] H. Altwaijry, “Bayesian Based Intrusion Detection System,” Transactions on Engineering Technologies, pp.29–44, 2012.
[33] J. François, C. Wagner, R. State, and T. Engel, “SAFEM: Scalable analysis of flows with entropic measures and SVM,” Proc. Conf. on Network Operations and Management Symposium, Maui, HI, USA, pp.510–513, April 2012.
[34] M.M. Najafabadi, T.M. Khoshgoftaar, C. Calvert, and C. Kemp, “Detection of SSH Brute Force Attacks Using Aggregated Netflow Data,” Proc. Conf. on Machine Learning and Applications, Miami, FL, USA, pp.283–288, Dec. 2015.
[35] P. Kalaivani and M. Vijaya, “Mining based detection of botnet traffic in network flow,” International Journal of Computer Science and Information Technology & Security, 2016.
[36] S. Zhao, M. Chandrahekhar, Y. Lee, and D. Medhi, “Real-time network anomaly detection system using machine learning,” Proc. Conf. on Design of Reliable Communication Networks, Kansas
1826

City, MO, USA, pp.267–270, March 2015.

[37] S. Zanero and S.M. Savaresi, “Unsupervised learning techniques for an intrusion detection system,” Proc. Conf. on Applied computing, Nicosia, Cyprus, pp.412–419, March 2004.

[38] J. Höchst, L. Baumgärtner, M. Hollick, and B. Freisleben, “Unsupervised Traffic Flow Classification Using a Neural Autoencoder,” Proc. Conf. on Local Computer Networks, Singapore, pp.523–526, Oct. 2017.

[39] J. Liu and T.G. Xu, “A semi-supervised clustering algorithm for application protocol recognition based on NetFlow,” Computer Technology and Development, vol.7, no.4, 2010.

[40] R.A.R. Ashfaq, X.-Z. Wang, J.Z. Huang, H. Abbas, and Y.-L. He, “Fuzziness based semi-supervised learning approach for intrusion detection system,” Information Sciences, vol.378, pp.484–497, 2017.

[41] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, “Handling imbalanced datasets: A review,” International Transactions on Computer Science and Engineering, vol.30, no.1, pp.25–36, 2006.

[42] Y. Sun, A.K. Wong, and M.S. Kamel, “Classification of imbalanced data: A review,” International Journal of Pattern Recognition and Artificial Intelligence, vol.23, no.4, pp.687–719, 2009.

[43] D. Ramychitra and P. Manikandan, “Imbalanced dataset classification and solutions: A review,” International Journal of Computing and Business Research, vol.5, no.4, 2014.

[44] F. Charte, A.J. Rivera, M.J. Jesus, and F. Herrera, “Addressing imbalance in multilabel classification: Measures and random resampling algorithms,” Neurocomputing, vol.163, pp.3–16, 2015.

[45] D.A. Cieslak, N.V. Chawla, and A. Striegel, “Combating imbalance in network intrusion datasets,” Proc. Conf. on Granular Computing, Atlanta, Georgia, USA, pp.732–737, May 2006.

[46] G.E. Batista, R.C. Prati, and M.C. Monard, “A study of the behavior of several methods for balancing machine learning training data,” ACM SIGKDD Explorations Newsletter, vol.6, no.1, pp.20–29, 2004.

[47] N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” Journal of Artificial Intelligence Research, vol.16, pp.321–357, 2002.

[48] T. Jo and N. Japkowicz, “Class imbalances versus small disjuncts,” ACM SIGKDD Explorations Newsletter, vol.6, no.1, pp.40–49, 2004.

[49] X.-Y. Tao, H.-B. Ji, and Y.-X. Xie, “A Modified PSVM and its Application to Unbalanced Data Classification,” Proc. Conf. on Natural Computation, Haikou, China, pp.488–490, Nov. 2007.

[50] D. Yao, J. Yang, and X. Zhan, “An improved random forest algorithm for class-imbalanced data classification and its application in PAD risk factors analysis,” The Open Electrical & Electronic Engineering Journal, vol.7, no.1, 2013.

[51] X.-W. Chen and M. Wasikowski, “FAST: a roc-based feature selection metric for small samples and imbalanced data classification problems,” Proc. Conf. on Knowledge discovery and data mining, Las Vegas, Nevada, USA, pp.124–132, Aug. 2008.

[52] J. Van Hulse, T.M. Khoshgoftaar, A. Napolitano, and R. Wald, “Feature Selection with High-Dimensional Imbalanced Data,” Proc. Conf. on Data Mining, Miami, FL, USA, pp.507–514, Dec. 2009.

[53] L. Zhu and X. Wang, “Entropy-based matrix learning machine for imbalanced data sets,” Pattern Recognition Letters, vol.88, pp.72–80, 2017.

[54] P. Cerda, G. Varoquaux, and B. Kégl, “Similarity encoding for learning with dirty categorical variables,” Machine Learning, vol.107, no.8, pp.1477–1494, 2018.

[55] P. Haag, NFDUMP, https://github.com/phag/nfdump, accessed Nov. 9. 2020.

[56] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” Proc. Conf. on Knowledge Discovery and Data Mining, San Francisco, California, USA, pp.785–794, Aug. 2016.

[57] T. Cover and P. Hart, “Nearest neighbor pattern classification,” IEEE Trans. Inf. Theory, vol.13, no.1, pp.21–27, 1967.

Cheng-Chung Kuo received the B.S. degree in Computer Science and Engineering from National Sun Yat Sen University and the M.S. degree in Computer Science and Information Engineering from Chang Gang University, Taiwan. His research interests include network security, malware analysis and network management.

Ding-Kai Tseng received the B.S. degree in Computer Science and Information Engineering from National Cheng Kung University and the M.S. degree in Institute of Computer Science and Communication Engineering at National Cheng Kung University, Taiwan. His research interests include network security, intrusion detection and machine learning.
Chu-Wei Tsai received the Ph.D. degree in Computer Science and Engineering from National Sun Yat Sen University, Kaohsiung, Taiwan, in 2009. He was a postdoctoral fellow with the Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan before joining the faculty of the Applied Geoinformatics and the Information Technology, Chia Nan University of Pharmacy & Science, Tainan, Taiwan in 2010 and 2012, respectively. He joined the faculty of the Department of Computer Science and Information Engineering, National Ilan University, Yilan, Taiwan, in 2014, the Department of Computer Science and Engineering, National Chung-Hsing University, Taichung, Taiwan, in 2017, and then the Department of Computer Science and Engineering, National Sun Yatsen University, Kaohsiung, Taiwan, in 2019, where he is currently an Assistant Professor. He has served as the Senior Associate Editor for the Journal of Internet Technology since 2014 and the Associate Editors for the IEEE Access, IET Networks, and IEEE Internet of Things Journal since 2017, 2018, and 2020. He has also been a member of the Editorial Board of the International Journal of Internet Technology and Secured Transactions (IJITST) and the Journal of Network and Computer Applications (JNCA) since 2016 and 2017.

Chu-Sing Yang is a Professor of Electrical Engineering in the Institute of Computer and Communication Engineering at National Cheng Kung University. He joined the faculty of the Department of Electrical Engineering at NCKU in 2006. He participated in the design and deployment of TaiWan Advance Research and Education Network (TWAREN), served as the Deputy Director of the National Center for High-Performance Computing from 2007 to 2008 and Director of Taiwan Information Security Center at National Cheng Kung University (TWISC@NCKU) from to 2010 to 2017. His research interests include software-defined networking, network management, cloud computing, and information security.