Cluster Ensemble with Link-Based Approach for Botnet Detection

Article *in* Journal of Network and Systems Management · July 2018

DOI: 10.1007/s10922-017-9436-x

2 authors, including:

Long Mai
Soongsil University
4 PUBLICATIONS **28** CITATIONS

SEE PROFILE

Noname manuscript No.
(will be inserted by the editor)

Cluster Ensemble with Link-based Approach for Botnet Detection

Long Mai · **Dong Kun Noh**

Received: date / Accepted: date

Abstract Botnet detection is one of the most imminent tasks for cyber security. Among popular botnet countermeasures, an intrusion detection system is the prominent mechanism. In the past, packet-based intrusion detection systems were popular. However, flow-based intrusion detection systems have been preferred in recent years due to their ability to adapt to modern high-speed networks. A collection of flows from an enterprise network usually contains both botnet traffic and normal traffic. To classify this traffic, supervised machine learning algorithms, i.e., classifications, have been applied and achieved a high accuracy. In an effort to improve the ability of intrusion detection systems against botnets, some studies have suggested partitioning flows into clusters before applying the classifications and this step could significantly reduce the complexity of a flow set. However, the instability of individual clustering algorithms is still a constraint for botnet detection.

To overcome this bottleneck, we propose a novel method that combines individual partitions to become a strong learner through the use of a link-based algorithm. Our experiments show that our cluster ensemble model outperforms existing botnet detection mechanisms with a high reliability. We also determine the balance between accuracy and computer resources for botnet detection, and thereby propose a range for the maximum duration time of flows in botnet research.

Keywords Cyber crime · Intrusion detection system · Network flow · Machine learning · Classification · Command and control

Long Mai
Department of Information Communication, Materials, and Chemistry Convergence, Technology,
Soongsil University, Seoul, Korea 06978
E-mail: longmaisg@ssu.ac.kr

Dong Kun Noh
Department of Software Convergence,
Soongsil University, Seoul, Korea 06978
E-mail: dnoh@ssu.ac.kr

1 INTRODUCTION

Nowadays, cybercrime has had an increasing impact and become a global disaster. Symantec [1], well-known security solutions provider, indicates that in 2014, attackers had intruded into 5 out of the 6 largest companies. The number of attacks has also increased by 40% year after year. Reports from the Intel Security Group (McAfee, [2]) argued that cybercrime and cyber espionage incurred large global costs. Among the different types of cyber threats, botnets result in some of the highest damage. A botnet is a huge network of compromised devices (bots) that are controlled by an attacker via the botmaster. The botmaster conquers bots to send spam, steal bank credentials, or launch denial of service (DoS) attacks. These operations violate the privacy of organizations, steal confidential information, and bring massive destruction to society. Therefore, botnet detection has emerged as an important task for cyber security.

Many studies have focused on detecting botnets, particularly for command and control (C&C) communication between bots and a botmaster. However, many challenges remain, due to the capabilities of the botnets to obfuscate and hide their C&C activities. A traditional approach, such as a rule-based IDS that identifies botnets using pre-defined signatures, cannot overcome such advanced evasion techniques [3], [4]. Hence, a machine learning (ML) approach, which can automatically recognize botnet communication patterns, is widely utilized. This approach applies a classification algorithm, a.k.a, binary classification, one type of supervised ML algorithm. It classifies traffic into botnet traffic and normal traffic by observing all packets through the IDS and analyzes the payload content of each packet (packet inspection). Besides the packet inspection, the ML classifications can also observe flows to detect botnet activities (flow inspection). A flow is a collection of packets within a time interval with common properties [5]. The statistic for all packet headers in a flow is the input of the classification algorithm. The flow inspection uses packet header information only; hence, it processes very quickly and should be a better choice than packet inspection, particularly in modern high-speed networks. Many studies have applied the ML classifications to detect botnet flows and have shown a high accuracy [6], [7].

However, in the flow inspection, we expect that each flow presents only a specific activity of the network, and this activity is distinct from the others. As the traffic increases, the flow set becomes more complex and includes many activities that are not much different. Consequently, the ML classification cannot be effectively performed for a big flow set because botnet activities can be confused with some abnormal activities that happen frequently in an enterprise network. Therefore, a data reduction technique should be deployed to divide flows into activity groups, which facilitates classification. Some studies have followed this approach to propose a hybrid model that includes two stages. In the first stage, they applied a clustering algorithm to generate a partition in which there were some clusters and each cluster covered flows that presented similar activities. In the second stage, the ML classification algorithms were applied to each cluster in order to separate botnet flows and benign flows [8].

Although the hybrid model performs well with a big flow set, it has a great weakness in the first stage. When a clustering algorithm generates a partition, it assigns flows to an appropriate cluster for which all flows have similar characteristics. Since

it could also assign flows to the wrong clusters, the classifications in the second stage cannot accurately separate botnet flows from normal flows when they are applied to those clusters. In other words, those wrong assignments negatively impact the overall performance of the botnet detection. This weakness is not trivial since clustering algorithms have been proven to be unstable, i.e., wrong assignments happen frequently [9].

In this paper, we propose a new botnet detection mechanism that overcomes the weakness of the hybrid model mentioned above. Our idea is to improve the precision of the first stage by using cluster ensemble approach, thereby increasing the overall accuracy of the hybrid model in botnet detection. At first, the proposed scheme generates some individual partitions. Then a link-based algorithm [10] is applied to combine them into a final partition in which the wrong flow assignment into inappropriate clusters decreases. In other words, the final partition has a higher precision and stability than each individual one. This improvement in the first stage results in an overall improvement in the accuracy of the hybrid model. To the best of our knowledge, this is the first work that uses this approach for botnet research. Fig. 1 illustrates the taxonomy of the proposed approach for botnet C&C activities detection. The dark branch represents our cluster ensemble approach.

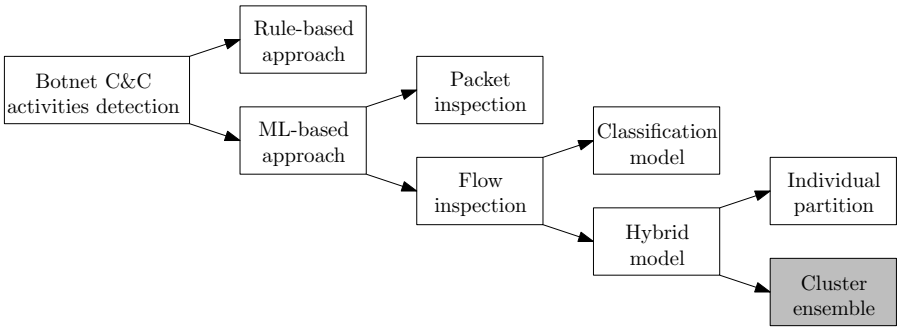

Fig. 1: Taxonomy of the proposed approach for botnet C&C activities detection

Moreover, we conduct a deep analysis of the traffic captured from both botnet C&C activities and benign activities to answer two questions that still remain in botnet research. The first question is, how many flows should we use to evaluate a detection method since different flow exporters generate different numbers of flows even with the same dataset? There should be a baseline for the minimum number of flows required for experiments. The second one is, how long should the time interval, i.e., maximum duration time, be to make a flow? If it is short, the botnet detection accuracy can improve. However, the number of flows also increases, and storage space and processing time are correspondingly consumed. When the time interval is long, the number of flows decreases. Since one flow covers many activities, it cannot accurately present a specific activity of a botnet. In other words, we need to determine an optimal duration time to balance between the accuracy and processing performance. The following is a summary of the main contributions of this paper.

- We design a novel botnet detection mechanism that outperforms existing methods and reduces the risk of instability when deploying clustering algorithms. We compare our model with 4 other schemes to point out its prominence to detect various types of botnet.
- We conduct comprehensive experiments to determine the minimum number of flows required to evaluate a detection method. We suggest that 100,000 flows should be the baseline for a performance evaluation in botnet detection.
- We determine how to choose the maximum duration time of flows, where there is a balance between accuracy and processing performance for botnet detection. Previous studies [11], [8] mentioned this, but lacked a deep analysis and explanation regarding their values.

The rest of this paper is organized as follows. Section 2 introduces the machine learning techniques that have been proposed to detect botnets and the unsolved issues. Section 3 is the motivation of our approach. In section 4, we present the framework that applies our link-based approach to improve the botnet detection mechanism. Section 5 shows the experiments conducted in our study with the corresponding results and discussion. Finally, Section 6 provides the conclusion and future work pointing out potential improvements to this study.

2 RELATED WORK

2.1 Botnet Detection

Bots, or zombies, are computers that have been compromised by malicious softwares (malwares). These computers are controlled by a botmaster and are interconnected to form a network (botnet) without the users knowledge. Typically, a botnet life cycle includes four phases: (1) *spread phase*: exploit vulnerabilities to deliver malwares; (2) *infection phase*: stealthily install malwares and obfuscate its presence; (3) *command and control (C&C) phase*: compose peer-to-peer communication between bots or bots to the botmaster; (4) *attack phase*: perform remote destruction. Preventing any of the phases in this intrusion chain leads to disabling the entire botnet threats. For practical reasons, it is nearly unfeasible to resist all types of malware during the spreading phase and infection phase in modern systems. In the attack phase, it might be too late to prevent an attack when all crimeware has been deployed. Hence, it is logical to concentrate on botnet detection efforts during C&C phase, when malicious software is still removable and the massive attacks have not yet happened.

Different methods have been proposed to detect botnet C&C communication. Among these, the machine learning approach is particularly prominent because of its ability to automatically recognize botnet C&C patterns, i.e., the ability to adapt to various types of botnets [12]. ML algorithms can discovery C&C activities by packets or by flows. A flow is a collection of packets from the same source and the same destination. Statistics of all packet headers in a flow (flow record) are used in flow-based botnet detection techniques. Since the flow-based approach only catches information of the packet headers, it can be processed very quickly. In contrast, packet-based approach belongs to the payload content and hence utilizes the resources and time in

a remarkably manner. Accordingly, flow-based approach can be adapted to modern high-speed networks, and it would rather be a better choice than packet inspection.

Applying ML algorithms to flow-based IDS necessitates two materials. The first material is a good traffic dataset that covers both botnet flows and real-world flows, with ground-true labels for training and testing. Some studies have provided datasets that combine of several existing public resources [13], [14], or collections obtained from their own traffic [15]. Such works bring the opportunity to botnet research community for comparing their studies. The second material is a tool to export flows from the network traffic. A flow exporter can capture flows from live traffic or datasets and can also derive flow records. Haddadi, et al., [16] compared the functionality of five flow exporters, with two of them collecting bi-directional flows and three collect uni-directional flows. Due to the differences in the mechanisms or predefined factors, these tools generate flow records with various specific features. This variety provides the chance to choose a tool that effectively generates flows toward the requirements of each study.

It is also important to note that when utilizing a flow exporter, we need to determine the maximum time duration of a flow, which is also called the time window (TW). Zhao, et al., [11] compared the botnet detection accuracy with TW strides from 10 seconds to 300 seconds. They showed that the performance is heavily influenced by the TW, and accordingly 300 seconds is the best choice. However, another study indicates that the length of the TW does not have much effects on botnet detection [6]. This contradiction reveals a gap in current studies and gives room to researchers to determine an optimal solution.

In the first time since applying to intrusion detection, ML techniques with flow-based approach exhibited a large potential to address various types of cyber attacks, particularly botnets [5]. Furthermore, Stevanovic, et al., [6] applied eight supervised ML algorithms to botnet detection and the input in these is the set of selected features. They experimented that the tree-related classifiers (Decision Tree, Random Forest, Boosted Decision Tree) performed well and had the potential to identify malicious traffic, with the Random Forest classifier performing best among them. However, when the size of the dataset increases, the complexity of the traffic can overwhelm the classifiers. In the study by Al and Omar [8], a combination consisting of an unsupervised clustering algorithm and classification was proposed. They first used K-means algorithm to divide the flows into some groups with similar characteristics. Then, they identified malicious flows or normal flows by applying ML classifications to each group. This hybrid model significantly reduces the heterogeneity of the flow set and brings advantages to the classifications. It is worth noting that there is not much of a difference between the definitions of a supervised ML algorithm and a classification algorithm. Hence, we use them interchangeably in this paper.

2.2 Cluster Ensemble

K-means is one of the most popular and effective clustering algorithms. However, K-means tends to converge on a local minimum [9], [17]. Thus, it might generate an unstable partition and bring confusion to the classification. Moreover, there is no such

individual clustering algorithm that could provide the best results for all datasets [18]. Consequently, it is very difficult to choose an appropriate unsupervised clustering algorithm for the hybrid model in botnet detection research.

To overcome these weaknesses of the clustering algorithm, particularly the instability, some researchers have introduced cluster ensemble that combines several clustering algorithms to a strong learner [19], [20], [21], [22]. Basically, the cluster ensemble is a combination of many individual clusterings to obtain a superior accuracy [23]. However, most papers have commonly suggested their own methods without carefully qualifying or have only presented data point relations. In [10], the authors proposed a new approach for cluster ensemble using a link-based algorithm. This study focuses on the relations between both data points and clusters inside the different individual partitions. Their experiments, in various well-known datasets, showed promising results over a wide range of areas. Therefore, we employ a link-based method in this paper to overcome the existing constraints of the botnet detection problem, especially the bottleneck in clustering algorithms.

3 MOTIVATION OF OUR APPROACH

As mentioned in Section 2, the hybrid model in botnet detection includes two stages. The first stage is a clustering algorithm and the second stage is classification. The clustering algorithm generates a partition that includes clusters of flows with similar features. Then the classification is applied to each cluster to separate botnet flows and normal flows. This hybrid model has shown potential to improve the accuracy of the botnet classification. However, previous works have only utilized an individual clustering algorithm and this leads to producing an unstable partition in which a flow could be assigned to a wrong cluster. This wrongness confuses the classification, and as a result, it reduces the overall accuracy of the botnet detection.

In our method, we overcome this weakness by using a cluster ensemble approach. Instead of generating one partition with a large number of clusters, we apply clustering algorithms to generate some partitions with a small number of clusters. Next, we utilize a link-based algorithm to combine these partitions, i.e., cluster ensemble, to a final partition. In this approach, in case one botnet flow is assigned to a wrong cluster in a partition, it also has a high ability of being assigned to the right clusters in other partitions. Hence, the wrongness in any partitions is mitigated and in the final partition, all flows could be assigned to the right clusters. Consequently, the classifications that are applied to these clusters could provide higher accuracy than when applied to clusters in a partition generated by an individual clustering algorithm. In other words, the role of the cluster ensemble in a hybrid model is to improve the precision of the first stage, i.e., increasing the overall performance of the botnet detection correspondingly.

We might have concerns over the complexity of the new method. Without loss of generality, we assume that a K-means cluster algorithm is utilized in the hybrid model. Let us then analyze the process of the K-means algorithm to generate a partition. Initially, we need to determine the number of clusters in a partition. After that, we randomly pick up a center point for each cluster and then follow the steps below:

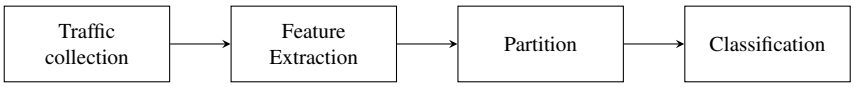

Fig. 2: Block diagram of new botnet detection model

– Calculate the distances from one point to all centers and assign this point to a cluster that has the nearest center. Repeat the calculation to all points and assign them to their appropriate clusters.
– Choose center points again, a new center point is a point with the shortest distance to all points in a cluster.
– Calculate the distance of a point to the new centers and assign it to a cluster with the the nearest center. Repeat for all points. Some of them can be assigned to new clusters.
– Iterate the process until all points do not change their clusters.

When the number of clusters is large, the computation cost increases accordingly. Since the number of clusters in each partition of the cluster ensemble method is small, the total calculation steps for all partitions approximates that of an individual partition with a large number of clusters, i.e., the cluster ensemble method does not increase the computational cost. Section 5 also provides further details to prove that there is not much of a change in the complexity when we apply the new botnet detection method.

4 PROPOSED FRAMEWORK FOR BOTNET DETECTION

In this section, we design a comprehensive framework for botnet detection in IDS. Our framework focuses on enhancing the robustness of the hybrid model by using a cluster ensemble approach. It includes 4 blocks. The first block is to choose a traffic collection that contains both botnet activities and normal activities. In the second block, we present the process to export flows and derive the flow records from traffic collected. We carefully analyze the features of the flow records and remove the features that could confuse machine learning algorithms. In the third block, we generate some partitions, and then utilize a link-based algorithm to combine them into a final partition. In the fourth block, we apply machine learning classification to each cluster in the final partition to separate the normal flows and malicious flows[1]. Fig. 2 illustrates our process.

4.1 Traffic collection

Traffic collection, which is also called traffic dataset, is used to qualify improvements in a new botnet detection method. These datasets need to satisfy common requirements such as including real botnet activities with precise labels, exhibiting heterogeneity, and being publicly available to the research community. There are only a few

[1] Two phrases *malicious flow* and *botnet flow* are used interchangeably in this paper.

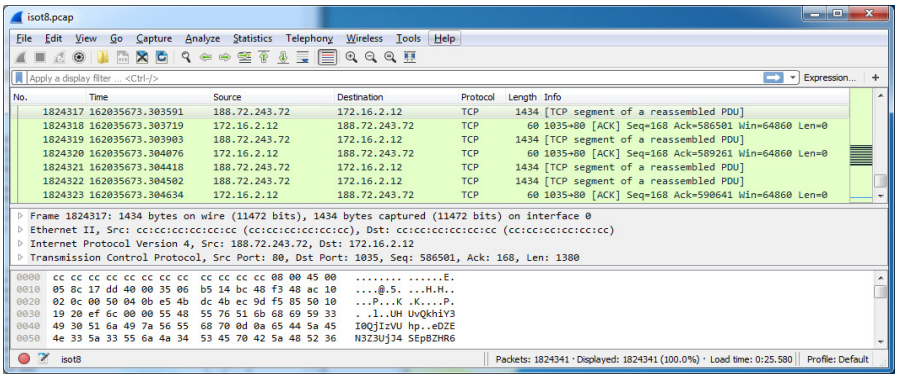

Fig. 3: A part of ISOT dataset (stored as pcap file)

botnet datasets that adhere to these conditions. For example, in [15], authors published a new dataset by capturing traffic from their university campus and creating 13 folders for different botnets scenarios. Although this case matches most requirements, packet captured (pcap) files are not provided and this absence is an obstacle to researchers who wish to conduct deep analysis. Beigi, et al., [24] also provided a combination of some available datasets, but it contains 43.92% malicious traffic and this proportion might not be realistic for actual networks.

Among public botnet datasets, ISOT is one of the most prominent resources [13]. Malicious traffic and normal traffic were obtained from some separate locations and were merged using the TcpReplay tool on the same network interface. The malicious traffic is obtain from the French chapter of the Honeynet Project [25], whereas the normal traffic is a combination of two different datasets from the Traffic Lab at Ericsson Research in Hungary [26] and from the Lawrence Berkeley National Laboratory (LBNL) (2005) [27]. The ISOT dataset involves some of the most prevalent botnets. In addition, its normal traffic includes a variety of applications, including HTTP web browsing, email, streaming media, Warcraft game, and Bittorrent. It can be used to represent everyday usage traffic in an enterprise network. Therefore, the ISOT dataset is appropriate to investigate the behaviors of botnets that have infected a running system. In this paper, we utilize the ISOT dataset both for training and to evaluate our new botnet detection method. Fig. 4 illustrates the process used to generate a traffic dataset, for which all traffic from normal devices (light color) and infected devices (dark color) has been collected by a router and stored as a pcap file. A flow exporter is deployed to generate flows using this pcap file, i.e., a traffic dataset. This process is clearly described in Section 4.2. Fig. 3 provides a part of ISOT dataset. It is worth noting that the start times for the packets is in UNIX time, which is easy to manipulate in the following sections.

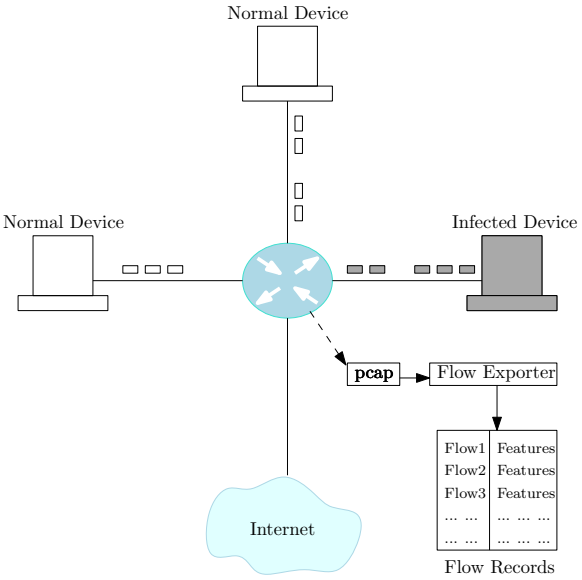

Fig. 4: Traffic collection
Packets from normal devices (light color) and infected devices (dark color) are collected by a router and stored as a pcap file. The flow exporter uses this pcap file to generate flows and stores flow features in the flow records.

4.2 Feature Extraction

In brief, a flow is a collection of packets that pass through a flow exporter during a time interval. All packets in a flow have a set of common properties: source IP address, destination IP address, source port, destination port and communication protocol [28], [29]. In Fig. 5, we present an example of flows that follow this definition. In this example, we assume that a flow is generated by a flow exporter (as in Fig. 4) during a time interval of 10 seconds. Packets (denoted by $P1$, $P2$, ...) are sent from IP1 (light color) and IP2 (dark color) to IP3. Let us analyze $Flow1$ in further detail. $Flow1$ is from IP1 to IP3 and hence, starts from packet $P1$ and covers packets $P1$, $P2$, $P3$. The time interval from the start point of $Flow1$, i.e., the start point of packet $P1$, to the start point of packet $P5$ exceeds 10 seconds, and hence, packet $P5$ is not covered by $Flow1$. Similarly, 6 flows are exported in this example. $Flow1$ ($P1$, $P2$, $P3$), $Flow3$ ($P5$, $P8$), $Flow4$ ($P9$), $Flow6$ ($P13$, $P14$, $P15$, $P16$) are from IP1 to IP3. $Flow2$ ($P4$, $P6$, $P7$) and $Flow5$ ($P10$, $P11$, $P12$) are from IP2 to IP3.

A flow record gives statistical information of all packet headers in a flow, i.e., the specific features of each flow. With more valuable features, ML algorithms could detect botnet flows with a higher accuracy. As shown in the the comparison in [16], Tranalyzer exporter can extract up to 93 meaningful features, and it overcomes other tools accordingly. Tranalyzer is lightweight and was designed for simplicity and scalability [30]. Thus, it adapts to high-speed networks. For these reasons, we use Tran-

Table 1: Flows exported by Tranalyzer

Flow	Start time	Duration	Src IP	Dst IP	Src Port	Dst Port	Prot
$Flow1$	10:22:03	20.5456	172.16.2.11	172.16.0.2	80	59451	6 (TCP)
$Flow2$	10:25:12	1.9789	172.16.0.12	172.16.2.2	32245	27343	17 (UDP)
$Flow3$	10:30:20	150.5559	172.16.2.3	172.16.2.13	22	1352	6 (TCP)
...	...	...	...	...	...	...	...
...	...	...	..	...	...	...	...

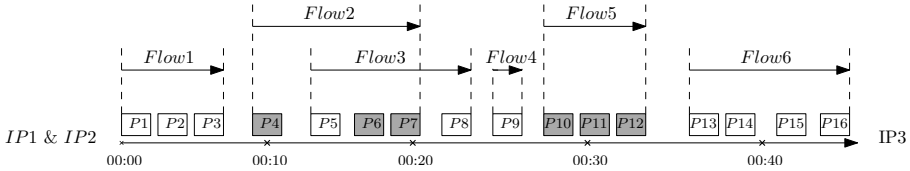

Fig. 5: Example of flows

Packets are sent from IP1 (light color) and IP2 (dark color) to IP3. Flow is exported during a time interval 10 seconds. There are total 6 flows in this example. $Flow1$, $Flow3$, $Flow4$, $Flow6$ are from IP1 to IP3. $Flow2$ and $Flow5$ are from IP2 to IP3.

alyzer to export flows and extract features for the ML algorithms in our framework (Fig. 4).

At first, we need to determine the time window (TW) for the exporter to generate a flow (maximum duration time of a flow). The authors in [11] recommend a TW of 300 seconds, because if the TW is too short, we might eliminate the unique characteristics of the botnet C&C, which can only be observed over a long term. However, if the TW is too long, the characteristics of a flow could become the average of many flow statistics. By following this instruction, we generate the flows with a maximum duration time of 300 seconds. Table 1 shows an example of the flows exported by Tranalyzer, with Src IP, Dst IP, and Prot as the abbreviations for Source IP, Destination IP, and Protocol, respectively.

From the set of features in the flow records derived with Tranalyzer, we need to remove the features that are irrelevant or not independent because they can negatively impact the classification step. The trend in botnet detection research is to improve performance with as few features as possible because fewer features reduces the complexity and storage space needed. From the group of 11 features suggested in [8], we apply a backward elimination technique [24], i.e., at each loop, we try to remove one feature that does not contribute to the overall performance. After all, we choose 7 features as input for the next block as follows:

- Source port
- Destination port
- Layer 4 protocol
- Average packet load ratio
- # of packets transferred per second
- # of bytes transferred per second
- # of connections between source IP and destination IP

Table 2: An example of packets in a flow (named as flow A) and flow features

(a) Packets in flow A

Packets	Start time	End time	# of bytes
*Packet*1	00:00.00	00:00.60	10
*Packet*2	00:00.60	00:20.38	245
*Packet*3	00:25.00	00:30.40	127
*Packet*4	00:30.40	00:33.70	92
*Packet*5	00:55.00	00:60.00	156

(b) Features of flow A

Feature	1	2	3	4	5	6	7
Value	80	20	6	126	0.083	10.5	3

Among these, the first 6 features represent unique characteristics for each flow. Specifically, the last feature could improve the botnet detection accuracy by unveiling the hidden techniques that advanced botnets use to bypass anomaly detection systems. Intuitively, we can explain the role of this feature by considering the number of connections from a host to an outside IP address. In a normal activity, this number is quite low due to the long packet payloads. In an infected host, by contrast, the bot continuously sends requests until they reach the botmaster. In addition, as a result of the defense systems, the connections to the botmaster are often dropped and the bot needs to repeatedly send requests. Consequently, the number of connections between a bot and its botmaster are extremely high in a short time interval.

In the following example, we describe how to derive the features of a flow in further detail. Assume that all 5 flows listed in Table 2 are in the same flow (named as flow A). Obviously, a real flow might have more packets than flow A. The duration time of flow A, from the start time of *Packet*1 to the end time of *Packet*5, is 60 seconds. There are a total of 630 bytes. Hence, we can calculate the features of flow A (with the assumption that we already know the Source port, Destination port, and Layer 4 Protocol) as below.

- Source Port = 80
- Destination port = 20
- Layer 4 protocol = 6 (TCP)
- Average packet load ratio = # of bytes / # of packets = 630/5 = 126
- # of packets transferred per second = 5/60 = 0.083
- # of bytes transferred per second = 630/60 = 10.5
- # of connections between the source IP and destination IP: *Packet*1 and *Packet*2 are sent continuously. After that, the Source IP and Destination IP reconnect,

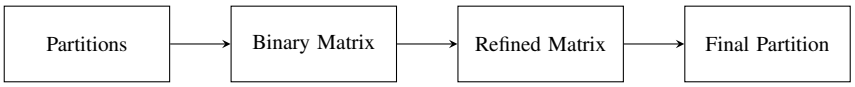

Fig. 6: Steps of Consensus Function to generate the final partition

*Packet*3 and *Packet*4 are sent. Then, Source IP and Destination IP reconnect again, and *Packet*5 is sent. So there are three connections.

We handle these values as the feature vector of flow A. The role of the features is the input of the ML algorithms. To describe this role in further detail, let us assume that a new partition is generated in which there are two clusters. Cluster 1 has center point 1 and cluster 2 has center point 2, with their feature vectors listed below.

$P1$ = [80, 20, 6, 120, 0.08, 10.0, 5].

$P2$ = [80, 20, 6, 130, 0.09, 12.5, 10].

To assign flow A to an appropriate cluster in this partition, we calculate the distances between flow A to all center points, and assign flow A to the cluster with the nearest distance. The following equation calculates the distance between two flows, and each flow has k features.

$$d = \sum_{i=1}^{k} \|x_i - \mu_i\|^2, \quad \text{with} \quad x_i, \mu_i : feature\, i \tag{1}$$

In our case, $k = 7$. The distances between flow A to center point 1 and center point 2 are 40.25 and 69, respectively. Therefore, flow A is assigned to cluster 1, which has the nearest distance. The method described in this example is also the basic step of K-means clustering algorithm, as mentioned in Section 3 and Section 5.

4.3 Partition

In this section, we first apply the clustering algorithm to create a set of individual partitions. Second, we utilize a consensus function to combine them into a final partition. Algorithm 1 and Fig. 7 summarize the outline of our cluster ensemble process. We formulate the consensus function to two steps: (1) derive a binary matrix from a set of partitions and employ a link-based algorithm to create a refined matrix (RM), and (2) generate the final partition by applying a clustering algorithm to that refined matrix (Fig. 6).

Without loss of generality, we assume that two partitions (i.e., two learners L_1 and L_2) are generated when we apply clustering algorithms to 5 input flows. In the figures below, a flow is illustrated as a dot for easy viewing, e.g., dot X_1 illustrates flow $F1$. Now, we intend to combine these individual learners to a strong learner L. (In this paper, the words *clustering*, *partition*, and *learner* are used interchangeably to mention about the product of a clustering algorithm.) L_1 is a set of 2 clusters, C_1^1 and C_2^1. L_2 covers C_1^2, C_2^2, C_3^2. The illustration of two learners L_1 and L_2 is in Fig. 8.

Now we arrange input flows and corresponding learners as in Table 3(a). From this result, we have a binary matrix (BM) that represents a crisp association between

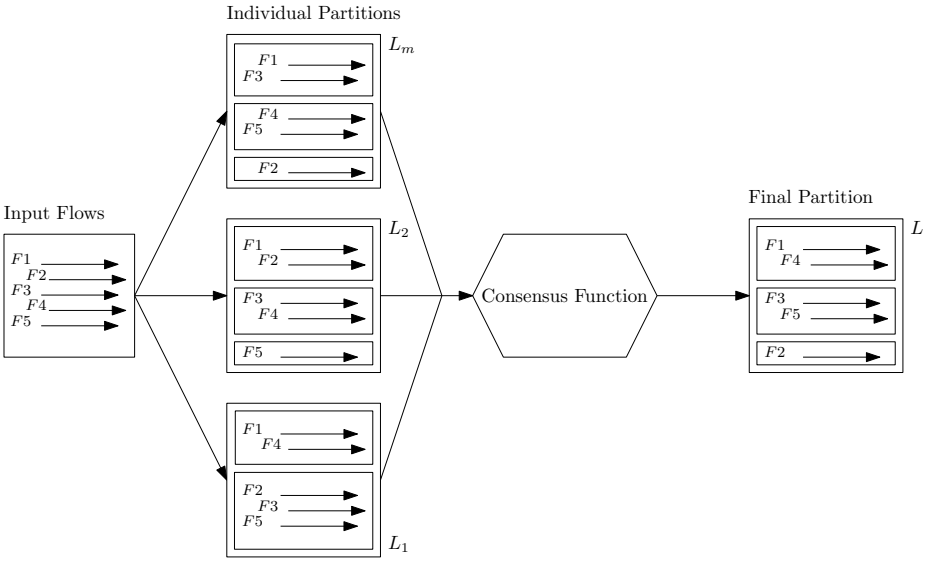

Fig. 7: Cluster ensemble approach

$L_1, L_2, ..., L_m$ are individual partitions.These are generated when we apply clustering algorithms to input flows. Each partition contains some clusters of flows. L is the final partition, generated from the consensus function.

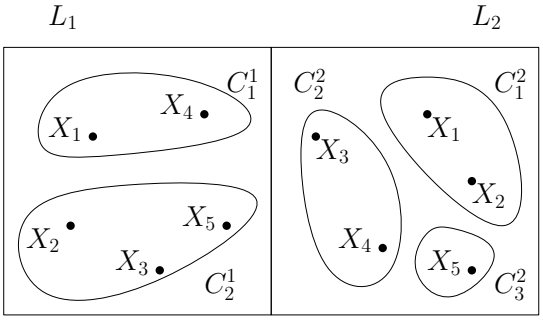

Fig. 8: Illustration of 2 learners L_1 and L_2

all inputs X_i and cluster $ct \in L$, with $BM(X_i, ct) \in \{0,1\}$ (Table 3(b)). For example, learner L_1 assigns flow X_3 to cluster C_2^1, and L_2 assigns X_3 to C_2^2. The binary matrix that reflects the row X_3 is:

$$BM(X_3) = \begin{cases} 1, & if column = C_1^1, C_2^2 \\ 0, & otherwise \end{cases}$$

The underlying problem is to explore the relationship between the base clusterings, in this case L_1 and L_2. This is tackled by refining the BM matrix, using a link-based approach. For each clustering L_h, $h = 1, ..., m$, there is a collection of clusters

Algorithm 1 Cluster ensemble with Link-based approach

Training

Given: training dataset of n flows

- m : number of partitions
- c_i : number of clusters in each partition, ▷ $i = 1, ..., m$
- $c = \sum c_i$: total number of clusters
- c_f : number of clusters in final partition
- C_j : classifier for cluster j, ▷ $j = 1, ..., c_f$

Output: Trained partitions and classifiers

Start

- Generate c_i clusters for partition i, ▷ $i = 1 \rightarrow m$
- Construct binary matrix $m \times c$
- Calculate refined matrix, ▷ rows: refined feature vectors
- Generate c_f clusters for final partition
- Train C_j using cluster j in final partition, ▷ $j = 1 \rightarrow c_f$

Stop

Testing

Given: Flow f_A

Output: Prediction of f_A as botnet flow or normal flow

Start

- Assign f_A to a cluster in partition i, ▷ $i = 1 \rightarrow m$
- Construct binary vector $1 \times c$
- Calculate refined feature vector f_{rA}
- Assign f_{rA} to a cluster j in final partition
- Classify f_A by corresponding classifier C_j.

Stop

Table 3: Clusters of inputs

(a) Assignments of inputs

	L_1	L_2
X_1	C_1^1	C_1^2
X_2	C_2^1	C_1^2
X_3	C_2^1	C_2^2
X_4	C_1^1	C_2^2
X_5	C_2^1	C_3^2

(b) Binary matrix

	C_1^1	C_2^1	C_1^2	C_2^2	C_3^2
X_1	1	0	1	0	0
X_2	0	1	1	0	0
X_3	0	1	0	1	0
X_4	1	0	0	1	0
X_5	0	1	0	0	1

C_*^h. To acquire the related information, a refined matrix ($RM \in \{0,1\}$) is created by the similarity function within the clusters:

$$RM(x_i, ct) = \begin{cases} 1, & if\ ct = C_*^h(x_i) \\ sim\left(ct, C_*^h(x_i)\right), & otherwise \end{cases} \quad (2)$$

or $BM(x_i, ct) = 1 \rightarrow RM(x_i, ct) = 1$ (note that the vise versa reflection $BM(x_i, ct) = 0 \rightarrow RM(x_i, ct) = 0$ is not true). With the cluster ensemble L of all input X_i, the similarities can be constructed using the weight edges between the clusters, according to

the proportion of overlap between the members within two clusters. It is also important to realize that the similarity of 2 clusters represents the closeness of them, in this case their shared members.

$$\left|S_{xy}\right| = \frac{\left|C_x \cap C_y\right|}{\left|C_x \cup C_y\right|} \tag{3}$$

For example:

$$With\, C_1^1, C_1^2 : \left|S_{xy}\right| = \frac{\{X_1, X_4\} \cap \{X_1, X_2\}}{\{X_1, X_4\} \cup \{X_1, X_2\}} = 0.33$$

Fig. 9 is the weighted network of clusters after we derive all existing edges, noting that it is an undirected graph.

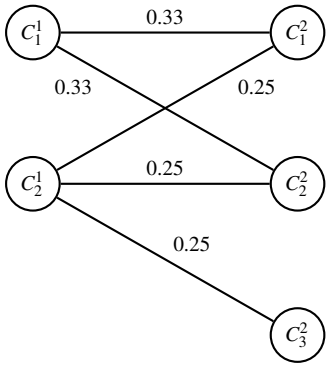

Fig. 9: Graph of weighted nodes

In Fig. 9, the graph only represents the basic relations between the clusters. We now apply a method called Link-Triple (LT) to develop the unweighted edges in the network. The initial technique identifies and counts the subgraphs that each vertex belongs to. We can thereby measure the weight of two arbitrary vertices v_x, v_y in the graph, which have relations with an existing vertex v_z:

$$LT_{xy}^z = min(\left|S_{xz}\right|, \left|S_{yz}\right|) \tag{4}$$

The weight between the two vertices is calculated by the summation of all link-triples:

$$LT_{xy} = \sum_{z=1}^{all} LT_{xy}^z \tag{5}$$

Finally, the similarity of v_x and v_y, i.e., the similarity of the two clusters C_x and C_y, which is denoted as $SIM(C_x, C_y)$, is defined using Equation (6), with LT_{max} in the denominator reflecting $S \in \{0,1\}$:

$$SIM(v_x, v_y) = \frac{LT_{xy}}{LT_{max}} \times CL \tag{6}$$

where $CL \in \{0,1\}$ is the confident level of similarity between two-nonidentical clusters. CL is normally set to 0.9 in the experiments [10]. Following the results in Equation (6), the refined matrix RM represents the associations discovered using the Link-based algorithm. This is shown in Table 4.

Table 4: Refined matrix

	C_1^1	C_2^1	C_1^2	C_2^2	C_3^2
X_1	1	0.78	1	0.9	0.39
X_2	0.78	1	1	0.9	0.39
X_3	0.78	1	0.9	1	0.39
X_4	1	0.78	0.9	1	0.39
X_5	0.78	1	0.39	0.39	1

In the refined matrix, we could consider each row as a refined feature vector of a flow. Now, we generate a final partition by applying a clustering algorithm to all rows of that matrix, i.e., assign all input flows to the clusters in the final partition. This last process of cluster ensemble covers the relations of the individual clusterings that we have explored using the link-based algorithm.

4.4 Classification

A clustering algorithm is unsupervised learning, hence in the final partition, there might be a mixture between normal flows and malicious flows inside each cluster. To accomplish categorization, we intend to employ a supervised classification algorithm. The study in [31] shows that Decision Tree is suitable for flow classification in high speed networks with high accuracy and reasonable training time. The study in [6] also achieved promising results using tree-related classifiers for botnet detection. The authors indicated that Random Forest prevails over other methods. However, we intentionally enhance clustering algorithm in this paper, and thus compound classifications have been omitted.

For the sake of simplicity, we apply Decision Tree classifier [32] in this framework to categorize botnet flows. Decision Tree represents a divide-and-conquer approach to the pattern recognition problem. Each tree is built from nodes and leaves. The nodes in a decision tree involve considering prominent features. From the nodes, trees are routed down to the leaves by learning from the inputs. This decision process guarantees that when a leaf is reached, a feature is classified and assigned appropriately. An example of Decision Tree classifier is described in Fig. 10. Let us utilize this example to classify the 3 flows listed in Table 3. In the first node, the Average Packet Sizes of all flows are larger than 10. Hence, these flows pass the first node and go down by the left branch. In the second node, Destination Ports of 3 flows are

obviously larger than 100 and hence, continuously go down by the right branch. In the third node, Protocol of $Flow2$ is UDP and therefore is classified as a botnet flow. The remaining flows ($Flow1$ and $Flow3$) have Protocol TCP and are consequently classified as normal flows.

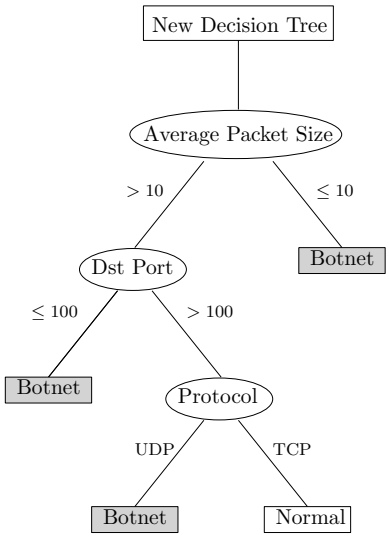

Fig. 10: Example of Decision Tree classifier

Various decision tree programs have been introduced. The C4.5 decision tree was devised by Ross Quinlan [33], and it has become one of the most popular ML algorithms in academia. Its successor C5.0 was also unveiled for commercial use. However, C4.5 is statistical-based and works with a shaky assumption. Sometimes the C4.5 method adds unnecessary structure to the final tree [34], and this could lead to missing botnet flows, or wrong alarms, which annoy network administrators. To resolve the botnet detection problem, Classification and Regression Trees (CART) learning system is more useful. CART can produce smaller trees than C4.5 with more accuracy [9]. In this study, we imply that CART algorithm is utilized in Decision Tree classifiers.

5 PERFORMANCE EVALUATION

When we apply ML algorithms to botnet detection, more input flows lead to a higher accuracy. However, in our observations, the accuracy is implemented only slightly when the number of input flows is over 100.000 (see Fig. 11). Therefore, we randomly choose 100.000 flows for our experiments.

In this section we use 10-folder cross-validation model to evaluate the performance. We randomly divide all input flows into 10 folders and then choose nine for

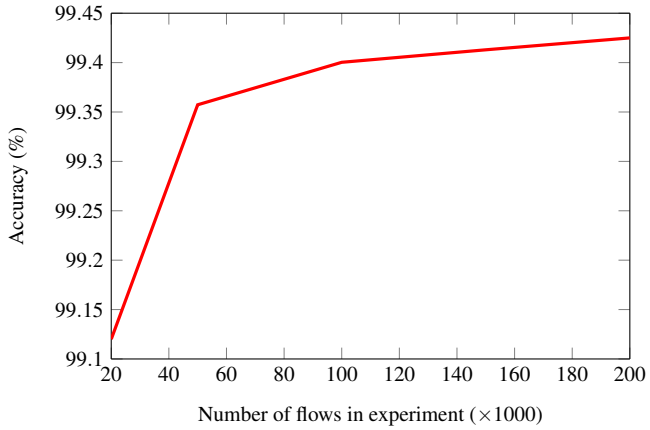

Fig. 11: Accuracy versus number of flows

training and one for testing. We iterate this replacement until all folders have been tested. To enhance the precision, we iterate this process 50 times, and calculate the average of the results. We also compare our new method with 4 existing ones, including: Decision Tree (DT), Gradient-Boosted Trees (GBT), Random Forest (RF), and the hybrid model proposed in [8]. (We consider that it is a combination of K-means and Random Forest). The metrics for comparisons include Accuracy (Acc), Detection Rate (DR), False Alarm Rate (FAR), and F-Measure.

$$Acc = \frac{TP+TN}{TP+TN+FP+FN} \tag{7}$$

$$DR = \frac{TP}{TP+FN} \tag{8}$$

$$FAR = \frac{FP}{FP+TN} \tag{9}$$

$$F-Measure = \frac{2TP}{2TP+FP+FN} \tag{10}$$

TP is the number of malicious flows that are correctly identified; *FP* is the number of normal flows that are incorrectly identified as malicious flows; *TN* is the number of normal flows that are correctly identified; *FN* is the number of malicious flows that are incorrectly identified as normal flows.

In all experiments, we utilize the K-means clustering algorithm in the cluster ensemble model because of its advantages and simplicity. To generate a set of clusterings using K-means, choosing a fixed number of clusters (k value) for all k-means clusterings provides the best results compared to random allocation [10]. Hence, we choose 3,4,5, and 6 as the number of clusters in a set of 4 clusterings. These 4 base clusterings are combined in the cluster ensemble. In the ISOT dataset, there are 3 types of botnets: Storm, Waledac and Zeus. Around 7.16% of the flows collected are

malicious. Table 5, Table 6, and Table 7 show a comparison between our cluster ensemble method and the other four methods (the proportion of Zeus flows is quite low, 0.002%, thus it is not evaluated separately).

Table 5: Comparison between detection algorithms and all botnets

Metrics	Acc	DR	FAR	F-Measure
DT	99.345	96.437	0.0026	95.308
RF	99.364	96.825	0.0023	95.475
GBT	99.373	**97.278**	**0.002**	95.556
Hybrid	99.376	96.844	0.0023	95.527
LB	**99.412**	95.985	0.0031	**95.935**

Table 6: Comparison between detection algorithms and Storm botnet

Metrics	Acc	DR	FAR	F-Measure
DT	99.631	94.783	0.0017	94.302
RF	99.644	96.65	**0.0012**	94.271
GBT	99.649	96.079	0.0013	94.645
Hybrid	99.647	96.752	0.0012	94.552
LB	**99.678**	**95.756**	0.0015	**95.209**

Table 7: Comparison between detection algorithms and Waledac botnet

Metrics	Acc	DR	FAR	F-Measure
DT	99.68	95.185	0.002	95.899
RF	99.687	97.304	0.0012	95.916
GBT	99.706	**98.362**	**0.0006**	96.083
Hybrid	99.678	96.814	0.0012	95.647
LB	**99.708**	96.458	0.0014	**96.145**

Our new model outperforms the other methods in terms of Accuracy and F-Measure, with all the types of botnets we analyzed. These results show that our model not only improves the classification of the botnet traffic and normal traffic but also achieves higher reliability. Random Forest classifier might maintain false alarm rate low while Gradient-Boosted Trees achieves a high detection rate. Fig. 12 and Fig. 13 illustrate the accuracy and F-Measure when we analyze all types of botnets.

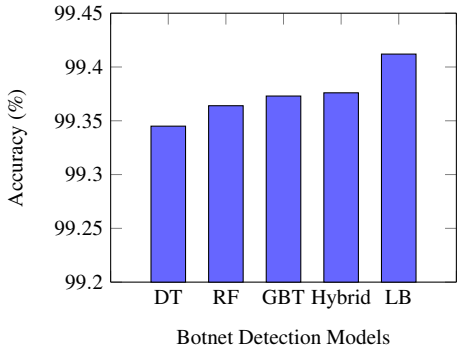

Fig. 12: Comparison of Accuracy

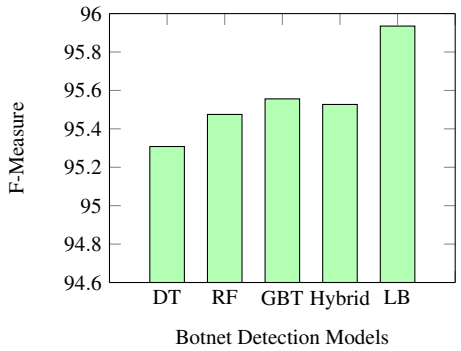

Fig. 13: Comparison of F-Measure

5.1 Determine maximum duration time of flows

In this section, we discuss the maximum duration time of a flow, or time window (TW) to derive the best results for botnet detection. Zhao, et al., [11] proposed that TW should be 300 seconds because the performance improves when TW increases. However, when we set TW at 300 seconds and take a deep analysis, almost all of flows are seen to have durations of less than 5 seconds, as shown in Fig. 14.

One flow is expired when it reaches the maximum duration time (active timeout), or the idle status is over the idle timeout. Ideally, within the 300-second time interval, there might be only one flow. But what happens if there are some very short flows that go through the flow exporter but with idle times that are also short enough to be combined into a unique flow? Unintentionally, the flow record covers the average features of many short flows but does not separate them as we would expect. This results in a reduction of the accuracy because the statistics do not cover the specific characteristics of each flow, which is the source that recognizes the unique activities of a botnet. The relation between the accuracy and TW is illustrated in Fig. 15, with 100.000 flows utilized.

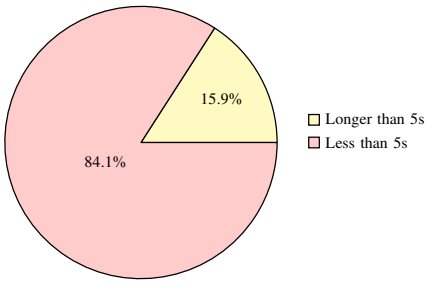

Fig. 14: Analysis of flow duration

On the other hand, if TW is set to 10 seconds or less, the total number of flows generated from the ISOT dataset is almost double than the amount in 300 seconds (see Fig. 16). However, in modern networks, flow records need to be stored for a long time for various purposes in accordance with the network administration policies. Clearly, it takes significant resources to store a double amount of flow records. In addition, time consuming for a botnet detection algorithm also increases with the number of flows. Thus, there is a trade-off between the performance and resource constraints.

To provide a balance between the accuracy and computer resources, we observe that when TW lengthens (see Fig. 15 and Fig. 16), either the number of flows or the accuracy are reduced. However, the number of flows reduces remarkably if TW is shorter than 150 seconds, i.e., too many flows are generated if TW is too short. For this reason, we should choose a TW greater than 150 seconds. Moreover, we observe that with two time intervals of TW, from 150 seconds to 300 seconds versus greater than 300 seconds, the accuracy is reduced with similar speed in both intervals. However, we get a higher benefit in the first interval because the number of flows is lower with a higher speed compared to the second one. Therefore, the range between 150 seconds and 300 seconds should be the best option for the maximum duration time of the flows. This range satisfies the balance between the botnet detection accuracy and acceptable resources for flow record storage.

5.2 Time consuming analysis

Instead of counting the execution time, which depends on computational resources, algorithm implementation, and program languages, we analyze the complexity of our method here. The time requirement for a Link-based algorithm is $O(C^2L+NC)$, where N is the number of flows, C is the number of all clusters, and L is the average number of links per cluster [10]. Since C and L are quite small, we can consider that Link-Based algorithm is a linear time consuming algorithm with approximately $O(NC)$. With the hybrid model in [8], K-means has time consuming of $O(Nk)$ with k as the number of clusters. In our experiments, the total clusters are $(3+4+5+6=18)$ for cluster ensemble, and 10 for K-means hybrid model. Accordingly, the complexity of the cluster ensemble is almost double of an individual K-means, in similar conditions of features and iterations. However, K-means leads to sub-optimal

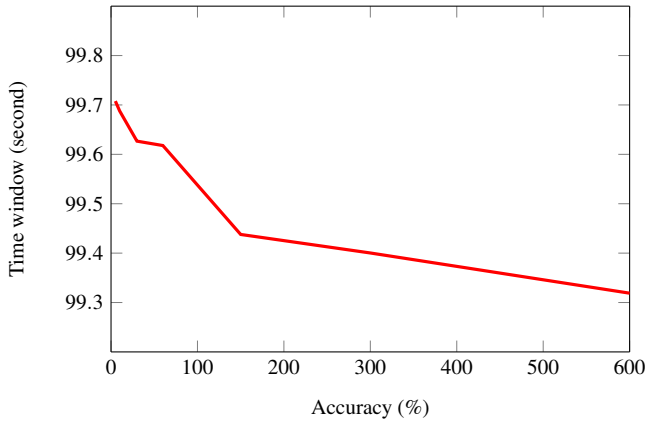

Fig. 15: Accuracy versus Time window (TW)

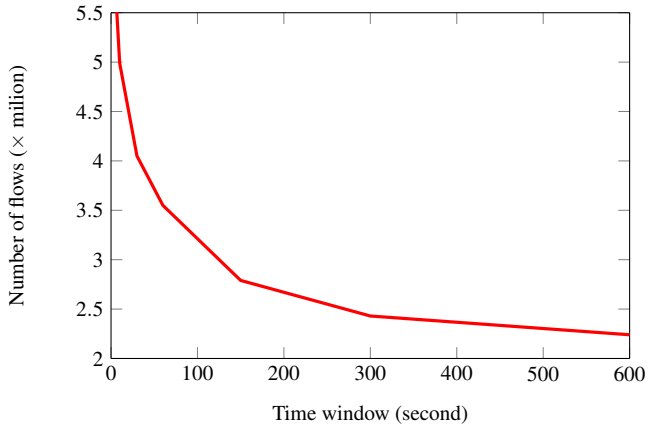

Fig. 16: Number of flows versus Time window (TW)

points [17], and hence, choosing good parameters is a difficult problem. In contrast, developers can easily choose some simple clusterings in a cluster ensemble with less effort but obtain more robust results. Consequently, the development time to build a botnet detection model is reduced, and the performance is significantly enhanced.

6 CONCLUSION

In this paper, we present a new method to overcome the constraints of a hybrid model in botnet detection. Our experiments show that our cluster ensemble with a link-based approach outperforms other botnet detection algorithms in intrusion detection systems. In particular, with a higher F-Measure, our method achieves better reliability in resolving the botnet detection problem. We also determine how to choose

the maximum duration time of flows in order to achieve a balance between accuracy and computer resource usage. We suggest that in the future, researchers could choose the duration time in the range from 150 seconds to 300 seconds. Our study applies Decision Tree classifier for flow categorization. However, individual classification usually implies errors. For this reason, future studies could combine multiple classifiers to improve the final results. Another point for further consideration is to compare Link-based algorithm with other cluster ensemble approaches. Such work gives room to researchers to improve botnet detection accuracy in network intrusion detection systems.

Acknowledgements This research was supported by the Ministry of Science, ICT and Future Planning (MSIP), Korea, under the Information Technology Research Center (ITRC) support program (IITP-2017-2012-0-00646) supervised by the Institute for Information & communications Technology Promotion (IITP).

References

1. http://www.symantec.com/.
2. http://www.mcafee.com/.
3. Guofei Gu, Phillip A Porras, Vinod Yegneswaran, Martin W Fong, and Wenke Lee. Bothunter: Detecting malware infection through ids-driven dialog correlation. In *Usenix Security*, volume 7, pages 1–16, 2007.
4. Martin Roesch et al. Snort: Lightweight intrusion detection for networks. In *LISA*, volume 99, pages 229–238, 1999.
5. Anna Sperotto, Gregor Schaffrath, Ramin Sadre, Cristian Morariu, Aiko Pras, and Burkhard Stiller. An overview of ip flow-based intrusion detection. *IEEE communications surveys & tutorials*, 12(3):343–356, 2010.
6. Matija Stevanovic and Jens Myrup Pedersen. An efficient flow-based botnet detection using supervised machine learning. In *Computing, Networking and Communications (ICNC), 2014 International Conference on*, pages 797–801. IEEE, 2014.
7. Fariba Haddadi and A Nur Zincir-Heywood. Botnet detection system analysis on the effect of botnet evolution and feature representation. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 893–900. ACM, 2015.
8. Omar Y Al-Jarrah, Omar Alhussein, Paul D Yoo, Sami Muhaidat, Kamal Taha, and Kwangjo Kim. Data randomization and cluster-based partitioning for botnet intrusion detection. 2015.
9. Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.
10. Natthakan Iam-On, Tossapon Boongoen, Simon Garrett, and Chris Price. A link-based approach to the cluster ensemble problem. *IEEE transactions on pattern analysis and machine intelligence*, 33(12):2396–2409, 2011.
11. David Zhao, Issa Traore, Bassam Sayed, Wei Lu, Sherif Saad, Ali Ghorbani, and Dan Garant. Botnet detection based on traffic behavior analysis and flow intervals. *Computers & Security*, 39:2–16, 2013.
12. SéRgio SC Silva, Rodrigo MP Silva, Raquel CG Pinto, and Ronaldo M Salles. Botnets: A survey. *Computer Networks*, 57(2):378–403, 2013.
13. Sherif Saad, Issa Traore, Ali Ghorbani, Bassam Sayed, David Zhao, Wei Lu, John Felix, and Payman Hakimian. Detecting p2p botnets through network behavior analysis and machine learning. In *Privacy, Security and Trust (PST), 2011 Ninth Annual International Conference on*, pages 174–180. IEEE, 2011.
14. Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3):357–374, 2012.
15. Sebastian Garcia, Martin Grill, Jan Stiborek, and Alejandro Zunino. An empirical comparison of botnet detection methods. *computers & security*, 45:100–123, 2014.

16. Fariba Haddadi and A Nur Zincir-Heywood. Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification. 2014.
17. Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.
18. Ludmila I Kuncheva and Stefan Todorov Hadjitodorov. Using diversity in cluster ensembles. In *Systems, man and cybernetics, 2004 IEEE international conference on*, volume 2, pages 1214–1219. IEEE, 2004.
19. Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.
20. Alexander Topchy, Anil K Jain, and William Punch. Clustering ensembles: Models of consensus and weak partitions. *IEEE Transactions on pattern analysis and machine intelligence*, 27(12):1866–1881, 2005.
21. Ludmila I Kuncheva and Dmitry P Vetrov. Evaluation of stability of k-means cluster ensembles with respect to random initialization. *IEEE transactions on pattern analysis and machine intelligence*, 28(11):1798–1808, 2006.
22. Xiaoli Z Fern and Wei Lin. Cluster ensemble selection. *Statistical Analysis and Data Mining*, 1(3):128–141, 2008.
23. Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.
24. Elaheh Biglar Beigi, Hossein Hadian Jazi, Natalia Stakhanova, and Ali A Ghorbani. Towards effective feature selection in machine learning-based botnet detection approaches. In *Communications and Network Security (CNS), 2014 IEEE Conference on*, pages 247–255. IEEE, 2014.
25. The honeynet project. french chapter [online], http://www.honeynet.org/chapters/france, 2011.
26. Géza Szabó, Dániel Orincsay, Szabolcs Malomsoky, and István Szabó. On the validation of traffic classification algorithms. In *International Conference on Passive and Active Network Measurement*, pages 72–81. Springer, 2008.
27. Lawrence berkeley national laboratory and icsi. lbnl/icsi enterprise tracing project. lbnl enterprise trace repository. [online], http://www.icir.org/enterprise-tracing; 2005.
28. Benoit Claise. Specification of the ip flow information export (ipfix) protocol for the exchange of ip traffic flow information. Technical report, 2008.
29. Ganesh Sadasivan. Architecture for ip flow information export. *Architecture*, 2009.
30. https://sourceforge.net/projects/tranalyzer/.
31. Murat Soysal and Ece Guran Schmidt. Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison. *Performance Evaluation*, 67(6):451–467, 2010.
32. Wei-Yin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.
33. J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.
34. Lior Rokach and Oded Maimon. *Data mining with decision trees: theory and applications*. World scientific, 2014.

Long Mai is a PhD candidate at the Department of Information Communication, Materials, and Chemistry Convergence Technology, Soongsil University. He received M.Sc. in Information and Communication Technology from Soongsil University in 2017. His research interests include applied machine learning and security.

Dong Kun Noh received the Ph.D. degree in Computer Science and Engineering from Seoul National University in 2007. He is currently associate professor in Department of Software Convergence at Soongsil University. His primary research interests include embedded system, mobile computing, and ubiquitous sensor network.