Received: September 30, 2024. Revised: October 31, 2024.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

FGT-B Model for Botnet Attack Detection Using Hybrid Analysis with Data Filtering and Aggregation Data Based on Network Traffic Flows

Dandy Pramana Hostiadi[1]* I Made Darma Susila[2] Yohanes Priyo Atmojo[2]
Gede Angga Pradipta[1] Tohari Ahmad[3] Muhammad Aidiel Rachman Putra[3]

[1]*Department of Magister Information Systems, Instittut Teknologi dan Bisnis STIKOM Bali, Bali, Indonesia*
[2]*Department of Informatics and Computer, Institut Teknologi dan Bisnis STIKOM Bali, Bali, Indonesia*
[3]*Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia*
* Corresponding author's Email: dandy@stikom-bali.ac.id

Abstract: Cyber-attacks are a serious threat and require proper anticipation. The currently developing attacks use botnets to attack networks with activities such as phishing, identity theft, Distributed Denial of Service (DDoS), spamming, and personal information theft. The development of a botnet detection model is needed to analyze, identify, accurately detect, and anticipate attack activities to reduce the risk of system damage, data theft, and more severe information. Previous studies have introduced botnet attack detection models using anomaly-based, signature, and mining-based approaches. However, the detection results only show the existence of attacks, have less than optimal detection accuracy, and require complex techniques to analyze attacks based on huge network traffic data. In fact, a botnet detection model design is needed to detect accurately and realistically and recognize different types of malware attacks with different characteristics. This paper proposes a botnet attack detection model with a data aggregation approach obtained by extracting attack behavior on frequency analysis, behavior graphs, and activity time on network traffic flows. The proposed research aims to design an accurate botnet attack detection model through a new approach, using data aggregation techniques based on attack characteristic analysis to detect it accurately and precisely. The novelty of the proposed model is the analysis approach with data filtering and aggregation techniques by combining botnet attack characteristics, namely activity frequency, behavior graph, and activity time segmentation on network traffic flow data, to increase accuracy and optimize computational processing. Three different datasets, namely CTU-13, NCC-1, and NCC-2, were used in the experiment and showed that the proposed model obtained high-performance detection in detecting botnet attacks in three different datasets with an average detection accuracy above 92%, precision above 86.72%, recall above 92%, and F1-Score above 88.89%. The best computation time on the NCC dataset was 39.0617 seconds. The proposed model can help network administrators analyze and determine handling steps on the network when a botnet attack occurs.

Keywords: Data agregation, Botnet attack detection, Network security, Hybrid analysis, Network infrastructure.

1. Introduction

The development of technology use in the digital era impacts increasing cyber cases [1] and requires serious handling [2]. The increase in dangerous cyber cases such as ransomware attacks, data breaches, Advanced Persistent Threat (APT) attacks, phishing, cryptojacking, Distributed Denial of Service Attack (DDoS), Remote Desktop Protocol (RDP) attacks, social engineering, web defacement, Artificial Intelligent and Internet of Things (IoT) cybercrime [3-5]. Of the several types of attacks that occur, the use of malware is the most intense, has the potential to cause system damage, and requires proper handling [6, 7]. The types of malware that are damaging and can damage the system massively are Trojan horses, viruses, worms, spyware, and botnets [8]. In addition, dangerous malware attacks are currently widely found in IoT communications [9-11] and attack smartphone devices in the form of mobile applications [7]. Botnet is a type of malware that is quite well-known and is the most dangerous type of

attack today. It has an attack structure that has developed from a centralized type to a distributed one, requiring the right detection technique [12]. Some dangerous activities of botnet malware include personal data theft, Distributed Denial of Service (DDoS) attacks, phishing, spamming, session hijacking, Remote Desktop Protocol attacks, and other dangerous activities that can cause financial losses [13]. Botnet malware consists of bot masters and bot clients [14, 15]. Bot masters instruct bot clients to infect and attack targets through Command and Control services. Bot masters can activate botnet attacks and quickly infect victim computers, so some intrusion detection system applications or antivirus applications cannot detect malware attacks accurately [16].

Several previous studies proposed a botnet attack detection using mining-based, signature, anomaly, and case-based approaches [10, 17]. Besides, analysis techniques based on botnet attack characteristics, such as using node-based graphs [18], time analysis [2, 19], and activity frequency approach [20], were introduced and were able to detect botnet attacks accurately. Machine learning techniques often use optimization in feature selection techniques and algorithm selection [21-23]. The results of optimization through feature selection can improve detection accuracy and reduce the use of irrelevant features [24]. Meanwhile, using the right machine learning algorithm can improve and reduce the computation time in the detection model in the analysis of huge network traffic data [4]. Some classification models used in botnet detection models include k-NN, Decision Tree, SVM-brf and Naïve Bayes [25, 26]. However, machine learning requires high computing resources and computing time in big data analysis.

A botnet attack detection model can be developed using private and public data. Private data can be collected by recording and simulating using several traffic recording tools such as Wireshark [23, 27, 28]. However, it requires validation of traffic data and involves experts to label each attack category and normal category. Several previous studies used public datasets such as CTU-13 [29], NCC [30], NCC-2 [31], and UNSW-NB 15 [32-34]. This dataset is data that has a large amount of traffic, consists of several activity scenarios, a number of different attacking botnets and different types of bots. Thus, in order to detect it, the right approach is needed to detect botnet attacks accurately and realtime, so it can reduce the impact of more severe system damage. This study proposes a botnet attack detection model using a combination of graph behavior analysis, distribution, and analysis time. The results of the analysis become features and are selected at the feature selection stage to improve the performance of the machine learning model. The purpose of the study is to improve the accuracy and computation time of the botnet detection model. The novelty of the proposed model is:

- Developing an analysis model with three approaches to activity frequency analysis, botnet attack behavior graphs, and statistical activity time of attacks that occur.
- Adopting data aggregation techniques to reduce network traffic data contained in network traffic data so as to reduce computing resources and computing time.
- Adopting dynamic threshold techniques as traffic data filtering based on the original characteristics of the attack based on the analysis of attack characteristics.

This paper uses three different datasets, the NCC, CTU-13, and NCC2 datasets, to detect botnet attack activity. These datasets contain diverse attack characteristics and different activity scenarios. The proposed model can help system administrators to facilitate analysis, anticipation and handling when a botnet attack occurs in a crucial computer network. Besides, the model in this article can help administrators and provide considerations in handling botnet attacks and can be used to improve the performance of intrusion detection or antimalware systems.

This paper is organized into several sections. Section 2 explains the related research that correlates in this paper. Section 3 describes the details of the proposed model, including its novelty. Section 4 presents the experiment results and discussion, highlighting the model's effectiveness. Finally, Section 5 presents the research conclusions and future research development.

2. Related work

Several previous studies proposed a botnet attack detection with anomaly-based approaches [35], signature [17], and mining [2, 3, 28, 36]. Several mining methods that are often used to detect botnet attacks are based on machine learning, such as classification in [35, 37, 38], clustering in [39, 40], similarity [41], and correlation analysis [38]. Research in [18] introduced a botnet detection model with characteristic analysis approaches by developing graph node-based analysis. The graph node-based technique represents attackers to targets as nodes and edges as flows and the number of activities. The representation of the number of activities from attackers to targets is identified as

outbound, response activities as inbound, the intensity of attack activities as outbound degree, and the intensity of response activities from targets as inbound degree. This technique has been successfully developed in [18]. It can detect botnets in every scenario on the CTU-13 dataset with a True Positive value of 100%. Still, it does not show the other matrix evaluations, such as false positive, false negative, and true negative values produced, which makes it unable to measure the model's performance to evaluate accuracy, precision, and recall.

In [41], a detection model involving analysis of attack activity time was introduced. Botnet attack analysis begins with classification using several classification models such as Random Forest, Decision tree, Naïve Baye, Logistic regression, and k-NN. Then, the model with the best performance is used in the attack stage analysis process through time analysis contained in the time-stamp feature. The model results show good botnet detection performance and attack scenarios. The experimental results produced an average detection accuracy of 99.998% using two datasets, CTU-13 and NCC. Still, detection accuracy depends on detection accuracy in the classification model, and the best detection results were only in scenarios 9, 10, and 11, with more than one bot attacker. This research is improved in [2], where botnet attack detection is done by analyzing the attack gap time between bots and targets. Detection begins by segmenting traffic against time for 1 hour. Then, a sequential analysis is carried out by grouping traffic to each attacking actor through source IP address grouping. For each repeated occurrence of the suspected source IP address of the botnet attacker, the model will group it as a series of different activities and measure the activity gap time. The analysis results show good performance detection but low accuracy, precision, and recall values. The test results with the CTU-13 dataset produced a detection accuracy of 96.73%, precision of 0.08%, and recall of 91.03%. Testing on the NCC dataset resulted in a detection accuracy of 99.19%, a precision of 1.34%, and a recall of 96.15%. While testing on the NCC-2 dataset resulted in a detection accuracy of 97.22%, a precision of 0.08%, and a recall of 96.08%. In addition, the influence of time segmentation for 1 hour affects the accuracy of the detection model.

The detection model using frequency analysis is carried out in [20], where the basic concept of calculating the number of activities that occur is based on forming a time window with a time intensity of every 15 minutes. Then, the data is collected and traced for its appearance in different segments. Experiments with the KDD99 dataset showed detection accuracy of 98.03%, precision of 98.79%, and recall of 97.26%. This research was developed in research [42], where the intensity of attacks appearing in different segments can form similarities in activity and become causal activities. Every activity that occurs periodically with similarities between segments indicates a bot group attack. This study has high detection results with a detection accuracy of 97.93% and recall of 97.81% using the CTU-13 dataset, which was replayed and sampled for 5 hours. It can detect the type of bot group attack activity even though the precision value is still low, below 72.73%. This research was then adopted in research [21], where the detection model using machine learning classification is optimized using a combination of feature selection techniques with Univariate and ANOVA. Univariate optimization is carried out to see the distribution of data bias, and ANOVA is used to select features. The selected features have non-mandatory properties through feature engineering techniques with one-hot-encode techniques adopted in [41, 42]. The experiments showed that the detection model has increased detection in the CTU-13 dataset with an accuracy of 99.27%, precision of 98.68%, and recall of 99.27%. The model successfully detected the NCC dataset with an accuracy of 99.03%, precision of 98.26%, and recall of 98.96%. Meanwhile, the NCC-2 dataset was successful in detecting with an accuracy of 98.87%, precision of 97.90%, and recall of 98.87%. However, this technique requires complex analysis techniques and time, so it has challenges if applied in real-time.

The challenge of detecting botnet attacks lies in the level of analysis of attack characteristics. There are types of botnet attack activities that are sporadic in the CTU-13 dataset [29], intense and periodic in the NCC dataset [30], and simultaneous in the NCC-2 dataset [31]. The CTU-13 dataset comprises 13 scenarios from a real network environment featuring normal traffic, botnets, and background traffic recording in 2011 from CTU University, Czech Republic. The captured network traffic flow spans from 0.26 to 66.85 hours, leading to varying amounts of data. Thirteen scenarios are presented with different numbers of bot attackers. Thirteen scenarios are presented with varying numbers of bots. Scenarios 1 to 8 and 13 have one attacking bot. While scenarios 9 to 12 have more than one attacking bot with various types. Each scenario has a total traffic of up to millions of records. The NCC dataset, presented as a bidirectional flow file, focuses on botnet activity. In this dataset, periodic refers to bot activity patterns across different segments, while intensity indicates that bot activity is present in each segment. These

characteristics make the dataset well-suited for evaluating bot group detection using a time-segmentation method. The NCC dataset includes 12,896,345 flows, categorized into Normal flows, Normal host flows, and Bot flows. Across 13 scenarios lasting 8 hours, the dataset contains 536,000 Bot flows (90.33%) and 1,720,205 Normal host flows. This dataset was obtained by extracting bot attack characteristics from the CTU-13 dataset.

The NCC-2 dataset captures attack activities that occur concurrently and are detected across several sensors, a characteristic known as simultaneous botnet attack. The NCC-2 dataset is divided into three sub-datasets: Sensor Id-1, Sensor Id-2, and Sensor Id-3, featuring botnet malware types such as Rbot, Virut, Neris, NSIS.ay, and Sogo. Each sub-dataset consists of botnet attack activity ranging from 146,000 to 294,000 (2.983% to 7.566%), normal activity ranging between 4,749,758 and 3,591,792 (97.017% to 92.434%), and traffic capture over 8 hours. This dataset models botnet attack behavior based on the CTU-13 and NCC datasets. Previous studies have introduced detection models with various approaches to detect botnet attacks and produced accurate results. However, some models still need to be optimal, such as having low precision and recall values, having high False Positive values, and requiring complex analysis techniques to be tested on certain activities. In addition, the handling technique for analyzing huge data requires the right analysis technique to obtain accurate and fast detection results. So, it requires a model that can recognize the characteristics of various botnet attacks, such as activity time-based analysis techniques, graph node-based, and activity frequency analysis. In addition, data aggregation techniques and filtering techniques are needed to reduce data to improve detection performance and reduce computing time in the botnet attack detection model.

3. Methodology

This paper proposes a botnet attack detection model using a hybrid of frequency analysis, graph node-based, and time analysis. Data aggregation techniques are used to extract features based on the characteristics obtained and filtered based on the original characteristic analysis approach of the type of attack obtained from the network header. The proposed model divided into four main stages: input botnet dataset, pre-processing phase, Botnet detection, and Evaluation. The proposed model is shown in Fig. 1.

3.1 Botnet dataset

This study uses three botnet datasets: CTU- 13 [29], NCC [30] and NCC-2 [31]. The three datasets have different botnet activity scenarios, botnet types, and network traffic flow data records. Three datasets with bidirectional network flow (.binetflow) format represent different attack characteristics: periodic (NCC), and simultaneous (NCC-2) and sporadic (CTU-13). Table 1 provides detailed descriptions of the datasets.

Each dataset has features and class labels. The NCC and CTU-13 datasets have 14 basic features. While the NCC-2 dataset has 17 basic features. Activity labels consist of botnet, normal and background class labels. The feature details of the three datasets are shown in Table 2.

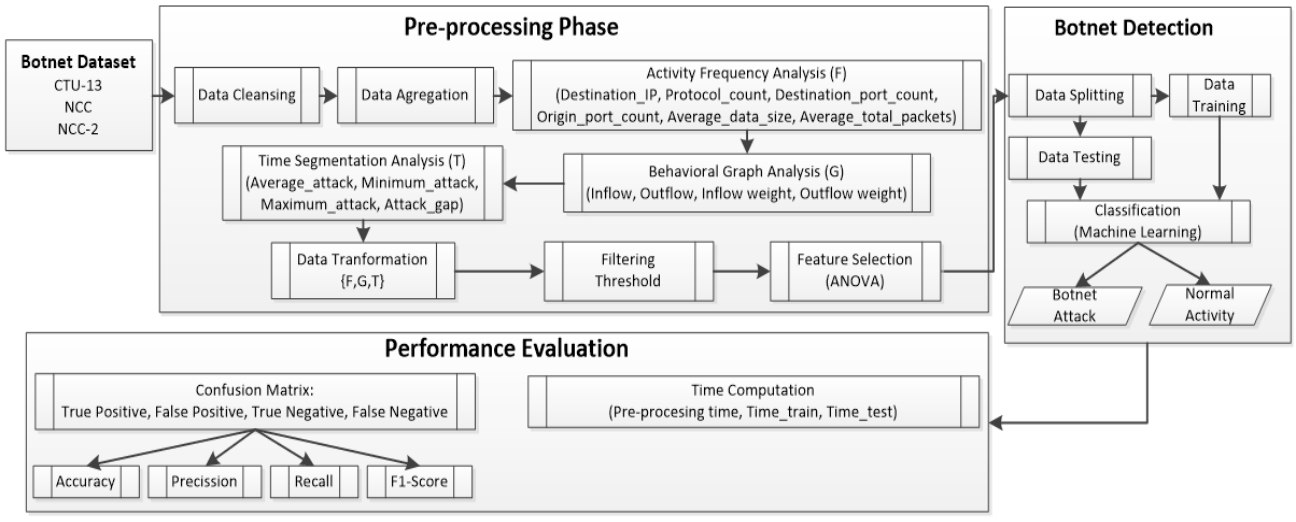


Figure. 1 Model Overview

Table 1. CTU-13 Dataset Description

Dataset	Scenario / Sensor ID	Duration (hour)	Network Flow		
			Total	Botnet	Normal
CTU -13 (sporadic)	1	6.15	2,824,636	40,961	2,783,675
	2	4.21	1,808,122	20,941	1,787,181
	3	66.85	4,710,638	26,822	4,683,816
	4	4.21	1,121,076	2,580	1,118,496
	5	11.63	129,832	901	128,931
	6	2	558,919	4,630	554,289
	7	0	114,077	4,630	114,014
	8	20	2,954,230	63 6,127	2,948,103
	9	5	2,087,508	184,987	1,902,521
	10	5	1,309,791	106,352	1,203,439
	11	0	107,251	8,164	99,087
	12	1	325,471	2,168	323,303
	13	16	1,925,149	40,003	1,885,146
NCC (periodic)	1	8	2,112,224	23,000	2,089,224
	2	8	1,465,182	24,000	1,441,182
	3	8	2,905,611	2,000	2,903,611
	4	8	724,388	11,000	713,388
	5	8	92,917	19,000	73,917
	6	8	512,021	6,000	506,021
	7	8	83,473	9,000	74,473
	8	8	2,871,217	14,000	2,857,217
	9	8	1,573,304	220,000	1,353,304
	10	8	984,369	60,000	924,369
	11	8	30,964	12,000	18,964
	12	8	274,186	9,000	265,186
	13	8	1,876,489	19,000	1,857,489
NCC-2 (simultaneous)	1	8	4,895,158	146,000	4,749,158
	2	8	5,998,133	364,000	5,634,133
	3	8	3,885,792	294,000	3,591,792

Table 2. Detail Feature on NCC, CTU-13 and NCC-2 Dataset

Dataset	Number	Original Feature
CTU-13	14	StartTime, DstAddr, TotBytes, SrcAddr, Proto, sTos, TotPkts, Dur, State, Dir, Sport, Dport, dTos, SrcBytes,
NCC	14	Sport, State, sTos, StartTime, Proto, Dport, TotPkts, Dir, SrcBytes, DstAddr, dTos, TotBytes, SrcAddr, Dur,.
NCC-2	17	SensorId, TotPkts, BotnetName, State, SrcAddr, Dur, TotBytes, Sport, Proto, DstAddr, dTos, StartTime, Dir, Dport, ActivityLabel, SrcBytes, sTos.

3.2 Preprocessing

The pre-processing stage is carried out before the detection process and begins with data cleansing, data aggregation, feature extraction based on activity frequency analysis, feature extraction based on behavioral graph analysis, feature extraction based on activity time analysis, data transformation, filtering traffic data using a dynamic threshold approach, and feature selection.

The data cleansing process is the process of cleaning traffic record data that has empty

valuescontained in each feature. In addition, the data cleansing process converts values from categorical to numeric, adopting techniques [28, 44]. Then, the data aggregation process combines several types of traffic based on the IP source address feature.

If network traffic is denoted as T, each traffic has a set of network header features NH, where there are n basic features for each dataset, then $T = f(NH)$, where $NH = (nh_{i,n}), i = 1,2,\dots,n$. IP Sources are $nh_{SrcAddr}$ collected into $UniqueAddresses$, a set containing all unique source addresses ($SrcAddr$) extracted from the network flow dataset (T). The result of the data aggregation is a group of activities from each IP source address and is sorted based on the activity time obtained from $nh_{StartTime}$ and then continued with analyzing behavioral characteristics into analysis features.

The first feature extraction is done by analyzing the activity frequency from each group of aggregated IP Source Addresses. This analysis approach adopts the technique in [41]. At this stage, feature extraction results are obtained, such as counts of unique protocols, destination ports, source ports, mean total packets, and mean source bytes. flows associated with the current source address ($addr$). $UniqueDportSet$ is a set that tracks the distinct destination ports ($Dport$) used in the flows associated with the current source address ($addr$). $UniqueSportSet$ is a set used to track the distinct source ports ($Sport$) used in the flows associated with the current source address ($addr$). $TotalPktsSum$ is a variable that stores the sum of the total number of packets ($TotPkts$) across all flows for the current source address ($addr$). $SrcBytesSum$ is a variable that stores the sum of the source bytes ($SrcBytes$) across all flows for the current source address ($addr$). $ActualClassSum$ is a variable that accumulates the sum of the $ActualClass$ for each flow, derived from the Label field in T. $FlowCount$ is a counter that tracks the number of flows ($rows$) in T associated with the current source address ($addr$). $UniqueProtoCount$ is the count of unique protocols ($Proto$) used by the flows for the current source address ($addr$). $UniqueDstPortCount$ is the count of unique destination ports ($Dport$) used by the flows for the current source address ($addr$). $UniqueSrcPortCount$ is the count of unique source ports ($Sport$) used by the flows for the current source address ($addr$). $MeanTotPkts$ is the mean ($average$) of the total number of packets ($TotPkts$) sent by the current source address ($addr$). $MeanSrcBytes$ is the mean ($average$) of the source bytes ($SrcBytes$) sent by the current source address ($addr$). The result of frequency feature extraction is expressed as $f(T)$, the output matrix, a summarized dataset where each row corresponds to a unique source address ($SrcAddr$) with aggregated features. The feature extraction process based on frequency analysis is shown in Algorithm 1.

Algorithm 1. Activity Frequency Analysis

Input: T
Output: $f(T)$
T : Network flow dataset, with 7 features (N × 7)
N ($LENGTH(T)$) : Number of network flow in dataset
$T[i]$: i-th row of T ($SrcAddr$, $Proto$, $Dport$, $Sport$, $TotPkts$, $SrcBytes$, $ActualClass$)
$T[i][j]$: value of feature with index j in i-th row of T
So, $T[i][0]$ is the value of feature $SrcAddr$ (feature with index 0) i-th row of T
$\{T[i][j]\}$: set of j-th feature value (collection of unique value from feature j)
$LENGTH(x)$: number of elements in x set

Step 1: Initialize an empty set for unique addresses
$UniqueAddresses = \emptyset$

Step 2: Determine unique source addresses
FOR i FROM 0 TO $LENGTH(T)$ - 1:
  $UniqueAddresses = UniqueAddresses \cup \{T[i][0]\}$

Step 3: Initialize the result matrix
$f(T) = EMPTY\ MATRIX$

Step 4: Iterate through each unique source address
FOR each $addr$ IN UniqueAddresses:
  Step 4.1: Initialize variables
  $sub_T = \emptyset$
  $UniqueProtoSet = \emptyset$
  $UniqueDportSet = \emptyset$
  $UniqueSportSet = \emptyset$
  $TotalPktsSum = 0$
  $SrcBytesSum = 0$
  $ActualClassSum = 0$
  $FlowCount = 0$

  Step 4.2: Filter flows where SrcAddr equals addr
  FOR i FROM 0 TO $LENGTH(T)$ - 1:
    IF $T[i][0] = addr$ THEN:
      $sub\_T = sub\_T \cup \{T[i]\}$

$UniqueProtoSet = UniqueProtoSet \cup \{T[i][1]\}$
$UniqueDportSet = UniqueDportSet \cup \{T[i][2]\}$
$UniqueSportSet = UniqueSportSet \cup \{T[i][3]\}$
$TotalPktsSum = TotalPktsSum + T[i][4]$
$SrcBytesSum = SrcBytesSum + T[i][5]$
$ActualClassSum = ActualClassSum + T[i][6]$
$FlowCount = FlowCount + 1$

Step 4.3: Compute unique counts and means

$UniqueProtoCount = LENGTH(UniqueProtoSet)$
$UniqueDstPortCount = LENGTH(UniqueDportSet)$
$UniqueSrcPortCount = LENGTH(UniqueSportSet)$
$MeanTotPkts = TotalPktsSum / FlowCount$
$MeanSrcBytes = SrcBytesSum / FlowCount$

Step 4.4: Determine the label for this group

IF $ActualClassSum > 0$ THEN:
  $Label = 'botnet'$
ELSE:
  $Label = 'normal'$

Step 5: Append new row to $f(T)$

$f(T) = f(T) \cup \{[addr, UniqueProtoCount, UniqueDstPortCount UniqueSrcPortCount, MeanTotPkts, MeanSrcBytes, Label]\}$

The second feature extraction based on the behavior graph, adopting the research [18]. The analysis is done by analyzing each $nh_{SrcAddr}$ to $nh_{DstAddr}$. The attack carried out by the botnet is illustrated through the communication graph node, where bots interact as vertex and communication flows as edges. Each bot is considered a vertex in a network, connected by edges. A vertex, representing a suspected bot, has two kinds of flows: indegree and outdegree. Inbound flows are those that enter the vertex, while outbound flows are those that exit it. These flows can be further classified into weighted indegree and weighted outdegree. The result of feature extraction at this stage is the production of indegree, outdegree, weighted indegree, and weighted outdegree features. $InDegree$ is the count of unique incoming addresses associated with each address in $g(T)$. $OutDegree$ is the count of unique outgoing addresses associated with each address in $g(T)$. $WeightedInDegree$ is the total count of incoming addresses associated with each address in $g(T)$. $WeightedOutDegree$ is the total count of outgoing addresses associated with each address in $g(T)$. An illustration of node-based graph analysis is shown in Fig. 2, where the bot attacker has an outdegree value of 3 and an indegree value of 3. For instance, the bot initiates three repeated flows to target host 1, 4 flows to target 2, and 2 flows to target 3. Meanwhile, the bot received two repeated flows from target host 2. As a result, the bot's total weighted outdegree is 9, while its weighted indegree is 4. The model analyzes by initializing the $OutgoingFlows$ is a dataset created from T, where the source address ($SrcAddr$) is renamed to $Addr$, the destination address ($DstAddr$) is renamed to $OtherAddr$, and a new column $Direction$ is added with the value 'out'. $IncomingFlows$ is a dataset created from T, where the source address ($SrcAddr$) is renamed to $OtherAddr$, the destination address ($DstAddr$) is renamed to $Addr$, and a new column $Direction$ is added with the value 'in'. T_concat is the concatenated T that combines $OutgoingFlows$ and $IncomingFlows$, resulting in a unified dataset with outgoing and incoming flow information. Eq. (1) calculates the unique flow directions. Meanwhile, Eq. (2) calculates the total access for all flow directions.

$$UniqueCount(Addr, Direction) = |\{OtherAddr \mid (Addr, OtherAddr, Direction) \in Tconcat\}| \quad (1)$$

$$TotalCount(Addr, Direction) = \sum(Addr, OtherAddr, Direction) \in Tconcat \quad (2)$$

The result is a grouped dataset where flows are aggregated by $Addr$ and $Direction$, calculating the number of unique addresses ($UniqueCount$) and the total count of addresses ($TotalCount$) for each direction. $g(T)$is the reshaped output matrix derived from the result, where missing values are filled with 0, and the index is reset, providing the final summarized dataset.

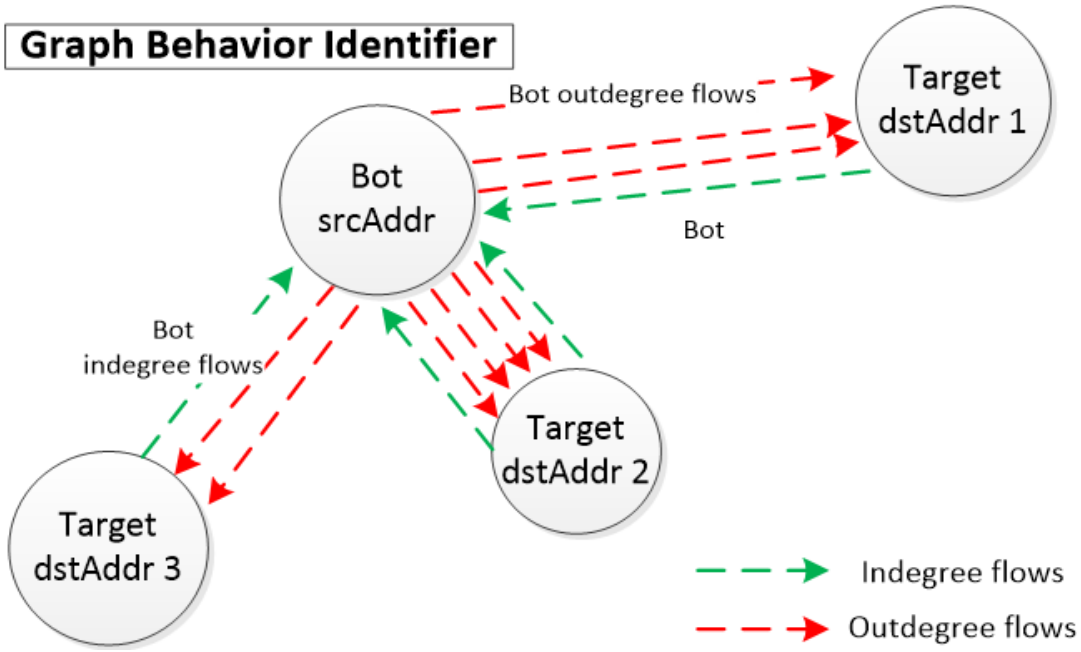


Figure. 2 Graph node-based Analysis

The node-based graph analysis process is shown in algorithm 2.

Algorithm 2. Graph Behavior Analysis

Input: T
Output: $g(T)$
$Addr$: The unique address involved in the network flow.
$|\cdot|$: The number of elements in the set (cardinality).
$OutgoingFlows$: network flow matrix that records outbound activities
$IncomingFlows$: network flow matrix that records inbound activities

Step 1: Create a dataset for outgoing flows

$$OutgoingFlows \leftarrow \{ (Addr, OtherAddr, Direction) \mid (Addr = SrcAddr, \; OtherAddr = DstAddr, \; Direction = 'out') \}$$

Step 2: Create a dataset for incoming flows

$$IncomingFlows \leftarrow \{ (Addr, OtherAddr, Direction) \mid (Addr = DstAddr, \; OtherAddr = SrcAddr, \; Direction = 'in') \}$$

Step 3: Concatenate outgoing and incoming flows

$$T\_concat \leftarrow OutgoingFlows \cup IncomingFlows$$

Step 4: Group by address and direction
Count the number of unique addresses with Eq. (1)
Calculate the total number of OtherAddrs connected to Addr using Eq. (2)

$$result \leftarrow \{(Addr, Direction, UniqueCount, TotalCount)$$

Step 5: Reshape the result

$$g(T) \leftarrow unstack(result)$$

Step 6: Rename columns in g(T)

$$g(T).columns \leftarrow ['Addr', \; 'InDegree', \; 'OutDegree',$$

The third feature extraction process involves a detailed analysis based on activity time. Activity time is calculated based on the repeated occurrence of each group of IP source addresses. This study adopts the study [19]. Where $seg(T)$ time is obtained from $nh_{StartTime}$, calculated activity interval from $nh_{StartTime\_end} - nh_{StartTime\_start}$. $seg(T)$ is the output matrix that contains a summarized dataset where each row corresponds to a unique source address with aggregated features, including mean, minimum, and maximum durations and time gaps. The results of feature extraction regarding activity

time are MeanDur, MinDur, MaxDur, MeanTimeGap, MinTimeGap, and MaxTimeGap. $MeanDur$ is the mean ($average$) duration of flows associated with each source address in $seg(T)$. $MinDur$ is the minimum duration of flows associated with each source address in $seg(T)$. $MaxDur$ is the maximum duration of flows associated with each source address in $seg(T)$. $MeanTimeGap$ is the mean ($average$) time gap between consecutive flows associated with each source address in $seg(T)$. $MinTimeGap$ is the minimum time gap between consecutive flows associated with each source address in $seg(T)$. $MaxTimeGap$ is the maximum time gap between consecutive flows associated with each source address in $seg(T)$. Eq. (3) is used to trace the Time gap between consecutive flows from the same source address. The process of extracting the activity time feature is shown in Algorithm 3.

$$\Delta t_i \begin{cases} (T[i][1] - T[i-1][1]); \\ \quad if\ T[i][0] = T[i-1][0] \\ 0; else \end{cases} \quad (3)$$

Algorithm 3. Time Segmentation Analysis

Input: T
Output: $seg(T)$
$T[i]$: Row i in the dataset T.
$SrcAddr$: Source address in a network flow.
$StartTime$: Start time of the network flow, to be converted into datetime format.
Dur: Duration of the network flow (column 2), in seconds.
Δt_i: Time gap between consecutive flows from the same source address.
$mean(x)$: mean value from x set
$min(x)$: the lower value from x set
$max(x)$: the highest value from x set

Step 1: Convert StartTime to datetime format
FOR i FROM 0 TO $LENGTH(T) - 1$:
$T[i][1] \leftarrow convert\_to\_datetime(T[i][1])$

Step 2: Sort dataset by SrcAddr and StartTime
$T \leftarrow sort(T, by = [SrcAddr, StartTime])$

Step 3: Calculate time gaps
FOR i FROM 0 TO $LENGTH(T) - 1$:
Calculate the time gap between consecutive flows from the same source address with Eq. (3)

Step 4: Group by SrcAddr and aggregate
Calculate $mean(Dur)$, $min(Dur)$, $max(Dur)$, $mean(TimeGap)$, $min(TimeGap)$ and $max(TimeGap)$

$seg(T) \leftarrow \{($
$SrcAddr, mean(Dur), min(Dur),$
$max(Dur), mean(TimeGap), min(TimeGap),$
$max(TimeGap))\}$

Step 5: Flatten the column names
$seg(T).columns$
$\leftarrow$ ['_'.$\boldsymbol{join}(col).\boldsymbol{strip}('\_')$ $\boldsymbol{for}$ col $\boldsymbol{in}$ $seg(T).co$

Step 6: Rename columns in $seg(T)$
$seg(T).columns$
$\leftarrow seg(T).columns.replace('SrcAddr','Addr')$

In the data transformation stage, the extraction results $f(T), g(T), and\ seg(T)$ are combined into an aggregate data (A), where $A = \{f_{(T)}, g_{(T)}, seg_{(T)}\}$. Merging is done to enlarge the feature dimension and obtain behavioral patterns according to the characteristics of the activities that have been extracted. The data transformation process is shown in Algorithm 4.

Algorithm 4. Data Transformation

Input: $f(T), g(T), seg(T)$
Output: A
$f(T)$: Address aggregation from $f(T)$
$g(T)$: Address behavior aggregation from $g(T)$
$seg(T)$: Time analysis aggregation from $seg(T)$
$FIND\_ROW(x, y)$: check if element y exist in x
$EMPTY\ MATRIX$: empty matrix to store the processed data

Step 1: Initialize matrices
$A_1 \leftarrow f(T)$
$A_2 \leftarrow g(T)$
$A_3 \leftarrow seg(T)$

Step 2: Perform outer join (union) on Addr for A_1 and A_2
$merged(A_1, A_2) = EMPTY\ MATRIX$

FOR each row_{a1} IN A_1:
$merged\_row \leftarrow row_{a1}$
$matching\_row_{a2} \leftarrow$
$FIND\_ROW(A_2, row\_a1[0])$
IF $matching\_row_{a2}$ EXISTS THEN:
$merged\_row \leftarrow merged\_row + matching\_row_{a2}[1:]$
ELSE:
$merged\_row \leftarrow merged\_row + [NA, NA, NA, NA]$

$merged(A_1, A_2)$
$\leftarrow merged(A_1, A_2)$
$\cup \{merged\_row\}$

FOR each row_{a2} IN A_2:
$check \leftarrow$
$FIND\_ROW(merged(A_1, A_2), row\_a2[0])$
IF $check$ NOT EXIST THEN:
$merged\_row \leftarrow [row\_a2[0]]$
$+ row\_a2[1:] + [NA, NA]$
$merged(A_1, A_2)$
$\leftarrow merged(A_1, A_2)$
$\cup \{merged\_row\}$
ELSE
DO NOTHING

Step 3: Perform outer join (union) with A_3 on Addr
$A \leftarrow EMPTY\ MATRIX$

FOR each row_merged IN $merged(A_1, A_2)$:
$merged\_row \leftarrow row\_merged$
$matching\_row_{a3}$
$\leftarrow FIND\_ROW(A_3, row\_merged[0])$
IF $matching\_row_{a3}$ EXISTS THEN:
$merged\_row \leftarrow merged\_row$
$+ matching\_row\_a3[1:]$
ELSE:
$merged\_row \leftarrow merged\_row$
$+ [NA, NA, NA]$
$A \leftarrow A \cup \{merged\_row\}$

Step 4: Replace empty values with NA
$A \leftarrow A.replace(["", " ", None], NA)$

Step 5: Fill missing values with 0
$A \leftarrow A.fillna(0)$

The results of the aggregated data (A) continue to the threshold filtering stage, which filters the data that appears based on the indication of the type of bot attack that occurs. The threshold analysis approach is dynamically adjusted based on the value determined in the recognition of the type of attack. The filters include outdegree, weighted outdegree, unique Proto Count, Unique DstPort Count, Unique SrcPort Count, Time Gap Mean, MeanToTkts, and MeanSrcBytes. For example, a bot attack is a type of Denial of Service (DoS) Attack, namely the attacker seeks to make a system, network, or service unavailable to its intended users by overwhelming it with excessive traffic or sending information that triggers a crash. This flood of incoming requests prevents legitimate users from accessing the service or resources, causing disruptions, downtime, or slow performance. It indicated that if analyzed based on the $g(T)$ analysis, it will have an outdegree of more than 1 activity, is repetitive where the weighted outdegree will be more than 1 time, and attacks several known ports such as HTTP, FTP or SSH ports [43-46]. Meanwhile, research [2, 41] states that the minimum active time of bot attacks is an average of 13 seconds until the next attack appears. The characteristics of DoS attacks have a relatively small but consistent data size, so the Bot attack filter has an average total packet and source byte value above 0 bytes [47, 48]. The data filtering process of aggregation data is shown in Algorithm 5 and the measurement of data reduction results is calculated by Eq. (4).

$$reduction\_ratio \begin{cases} 0; if\ N_{before} = 0 \\ \frac{(N_{before} - N_{after})}{N_{before}} \times 100; else \end{cases} \quad (4)$$

Algorithm 5. Aggregated Network Flow Filtering
Input: A
Output: $filter(A)$

Step 1: Apply filtering criteria
FOR each i IN A:
$filter(A) \leftarrow \{A[i] \mid$
$OutDegree_i > 1,$
$WeightedOutDegree_i > 1,$
$UniqueProtoCount_i \geq 3,$
$UniqueDstPortCount_i \geq 3,$
$UniqueSrcPortCount_i \geq 3,$
$TimeGap_{mean_i} \leq 13,$
$MeanTotPkts_i \neq 0,$
$MeanSrcBytes_i \neq 0\}$

Step 2: Count the number of rows before and after filtering
$countBefore \leftarrow NUMBER\_OF\_ROWS(A)$
$countAfter$
$\leftarrow NUMBER\_OF\_ROWS(filter(A))$

Step 3: Calculate the reduction ratio
execute Eq. (4)

Step 4: Return $filter(A)$ and $reduction\_ratio$

After the data filtering process, an optimization technique is carried out through the feature selection process. Feature selection is carried out to improve the performance of the classification model. In this study, the feature selection model was carried out using ANOVA, which has been used and showed good optimization results in research [9, 21]. In this

study, feature selection selects 75% of the best features used in the classification model at the botnet detection stage. The feature selection process with ANOVA is shown in Algorithm 6.

Algorithm 6. Feature Selection -ANOVA

INPUT: X, Y
OUTPUT: $Fval$
$Fval$: a set of f_values
X : matrix representing input from data $(m \times n)$
Y : vector containing the feature target $(n \times 1)$
n : columns number in data for matrix X
m : rows number in data for matrix X
i : row index used in loop
j : column index used in loop
x_{ij} : value at the i-th row and j-th column in matrix x
$f\mu$: mean value of a feature
$y\mu$: mean value of the target feature
$bVar$: variance between groups
$wVar$: variance within groups
$sumY$: variable to store the sum of all vectors in Y
sum : variable to store the sum of all data in feature
bdf : degrees of freedom in each group
wdf : degrees of freedom within groups
bms : between group mean square
wms : mean square between groups
$fval$: f-value for the feature.

Step 1: loop all columns for matrix X
$Fval \leftarrow []$
for $j \leftarrow 1$ to n **do**
Step 2: calculate group and total means value
$sumX \leftarrow 0$
$sum \leftarrow 0$
for $i \leftarrow 1$ to m **do**
$sumY \leftarrow sumY + Y_{ji}$
$sum \leftarrow sum + x_{ji}$
end for
$y\mu \leftarrow sumY/m$
$f\mu \leftarrow sum/m$
Step 3: calculate the between and within group variance
$bVar \leftarrow (f\mu - y\mu)^2$
$wVar \leftarrow 0$
for $i \leftarrow 1$ to m **do**
$wVar \leftarrow wVar + \left((y_{ji} - f\mu)^2\right)$
end for
Step 4: degrees calculation for free and mean squares values
$bdf \leftarrow 1$
$wdf \leftarrow m - 1$
$bms \leftarrow bVar/bdf$
$wms \leftarrow wVar/wdf$
$fval \leftarrow bms/wms$
$Fval_i \leftarrow fval$
end for
Step 5: return $Fval$

3.3 Botnet detection

At this stage, botnet detection is carried out using machine learning classification. In this paper, the model used four classification algorithms: decision tree, k-NN, SVM, and random forest. The four classification algorithms are selected because they have high-performance detection and have been used in previous studies in [21, 26, 37, 41, 49, 50]. The classification process starts by splitting the dataset into two sections: a training set and a test set. The training set is used to build the classification model, while the test set is reserved for evaluation. Typically, 70% of the data from each class is assigned to the training set, and the remaining 30% is allocated for testing.

3.4 Evaluation

Evaluation In this paper, model evaluation is carried out from analyzing classification model performance and computing time. In the classification model, evaluation is carried out by tracing the confusion matrix values, namely True Positive (TP), True Negative (TN), False Negative (FN), and False Positive (FN). TP refers to the number of botnet attack instances correctly identified as attacks. FN occurs when botnet attack data is mistakenly classified as normal activity. TN indicates the correct identification of normal activity as normal, while FP refers to normal data that is incorrectly classified as a botnet attack. Then, the accuracy (acc), recall (rec), Precision (prec), and F1-Score (F1) measurements were carried out.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (5)$$

$$Precision = \frac{TP}{TP+FP} \quad (6)$$

$$Recall = \frac{TP}{TP+FN} \quad (7)$$

$$F1 - Score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \quad (8)$$

Computational time is evaluated from the pre-processing stage to the detection stage. At the data pre-processing stage, the time measured is the data cleansing process, data aggregation, activity frequency analysis, node-based graph analysis, activity time analysis, data transformation, filtering threshold, and feature selection. At the detection stage, the data training and data testing processes are measured.

4. Result and discussion

This paper proposes a botnet activity detection model through hybrid analysis by analyzing large-scale network traffic flows. In this research, the computer has a specification with a 2.5 GHz dual-core Intel Core i7 processor, 16 GB of RAM, 1 TB of SSD storage, and Python programming to develop the botnet model detection.

4.1 Experiment result

This study uses three different datasets, namely CTU-13, NCC, and NCC-2. The model performs the data cleansing process in the initial stage. The results of the data cleansing process show a reduction in the number of data records due to the deletion of records with empty values in nh. In addition, several features successfully transformed into numeric data are $nh_{SrcAddr}$, $nh_{\mathbf{DstAddr}}$, $nh_{\mathbf{Proto}}$, $nh_{\mathbf{Dir}}$ and $nh_{\mathbf{Dur}}$. The average data reduction data from the data cleansing results on the CTU-13 dataset is 0.08%, on the NCC dataset is 0.56% and at 0.04%. $nh_{SrcPort}$ and $nh_{DstPort}$ are the empty network headers because they are related to nh_{Proto} if there is an ICMP protocol type, where ICMP does not have a source port or destination port value recorded in network traffic flows. However, SPAM and DoS attacks can be carried out by botnets by utilizing the ICMP protocol as part of the attack stage. The reduction results from the data cleansing stage are shown in Table 3.

After data cleansing, the model performs the data aggregation process. In this stage, the aggregation process is carried out by grouping the activity sets based on $nh_{SrcAddr}$ and sorting them by $nh_{StartTime}$. The results of data aggregation, shown in Table 4, show the number of unique source IPs in each scenario dataset.

After aggregation, the model performs feature extraction based on activity characteristics. Activity frequency analysis $f(T)$ extracts feature such as counts of unique protocols, destination ports, source ports, mean total packets, and mean source bytes from each $nh_{SrcAddr}$. Table 5 shows an example of feature extraction results based on activity frequency.

Table 3. Result of Data Cleansing

Scen/ Sensor ID	Number of records in data cleansing								
	CTU-13 Dataset			NCC			NCC-2		
	before	after	Reduction (%)	before	after	Reduction (%)	before	After	Reduction (%)
1	2,112,224	2,111,773	0.005	2,112,224	2,094,904	0.82	4,895,158	4,894,668	0.01
2	1,465,182	1,464,977	0.01	1,465,182	1,459,321	0.4	5,998,133	5,994,894	0.054
3	2,905,611	2,904,993	0.009	2,905,611	2,891,373	0.49	3,885,792	3,883,460	0.06
4	724,388	724,188	0.024	724,388	718,231	0.85			
5	92,917	92,310	0.257	92,917	92,137	0.84			
6	512,021	511,443	0.038	512,021	507,874	0.81			
7	83,473	83,278	0.138	83,473	83,097	0.45			
8	2,871,217	2,870,890	0.007	2,871,217	2,865,187	0.21			
9	1,573,304	1,572,813	0.015	1,573,304	1,563,707	0.61			
10	984,369	983,909	0.028	984,369	983,582	0.08			
11	30,964	30,772	0.439	30,964	30,695	0.87			
12	274,186	273,954	0.041	274,186	273,912	0.1			
13	1,876,489	1,876,194	0.007	1,876,489	1,863,166	0.71			

Table 4. Result of Data Agregation

Scenario/Sensor ID	Number of Unique Host in Dataset after Agregation		
	CTU-13	NCC	NCC-2
1	542,093	342,741	723,635
2	392,301	252,364	861,588
3	359,987	240,781	448,309
4	131,177	66,014	
5	28,301	10,347	
6	72,345	46,628	
7	23,057	9,599	
8	333,816	252,163	
9	313,766	180,559	
10	151,256	89,919	
11	29,157	3,732	
12	59,552	33,622	
13	277,486	209,886	

Table 5. Example of Feature Extraction Results Based on Activity Frequency

Source IP Address	Unique ProtoCount	UniqueDst PortCount	UniqueSrc PortCount	MeanTotPkts	MeanSrcBytes	Label
147.32.84.165	3	23	3,913	4.674	543.164	botnet
147.32.84.166	2	6	403	21,157.764	436,167.440	normal
147.32.84.168	2	3	45	35.920	1,590.140	normal
147.32.84.189	3	9	2,093	19.604	866.633	normal
147.32.84.19	3	84	2,632	6.251	3,079.534	normal
147.32.84.191	3	24	3,917	6.829	1,317.690	botnet
147.32.84.192	2	18	3,930	5.623	857.245	botnet
147.32.84.193	2	19	3,867	6.958	1,212.571	botnet
147.32.84.194	3	437	1,717	83.790	5,824.008	normal
147.32.84.199	1	188	3	1.299	354.851	normal
147.32.84.206	3	19	3,889	6.914	465.313	botnet
147.32.84.207	2	19	3,863	12.906	1,569.953	botnet
…	…	…	…	…	…	…

The second feature extraction analyzes the communication flow between hosts represented in the form of a node-based graph $g(T)$ obtained from tracing the communication of each $nh_{SrcAddr}$ to $nh_{DstAddr}$. Feature extraction produces four features: indegree, outdegree, weighted indegree, and weighted outdegree. Table 6 shows an example of the results of feature extraction based on node-based graph analysis.

The third feature extraction is to perform analysis based on the activity time of each host $seg(T)$, which is done based on the grouping of $nh_{SrcAddr}$ and sorted by activity time at $nh_{StartTime}$. The feature extraction results against activity time are MeanDur, MinDur, MaxDur, MeanTimeGap, MinTimeGap, and MaxTimeGap. Table 7 shows an example of the results of feature extraction based on activity time analysis.

The results of the feature extraction $f(T)$, $g(T)$, and $seg(T)$ are aggregated into aggregate data (A). This data forms a new data record that combines each feature from the previous extraction process. The results of the aggregation obtained based on $nh_{SrcAddr}$, produce 15 new features, namely UniqueProtoCount, UniqueDstPortCount, UniqueSrcPortCount, MeanTotPkts, MeanSrcBytes, InDegree, OutDegree, WeightedInDegree, WeightedOutDegree, Dur_mean, Dur_min,

Table 6. Example of Feature Extraction Results Based on Node-Based Graph Analysis

Source IP Address	InDegree	OutDegree	WeightedInDegree	WeightedOutDegree	Label
147.32.84.162	9	23	44	1,337	normal
147.32.84.164	19	139	45	3,526	normal
147.32.84.165	24	1,080	68	22,000	botnet
147.32.84.166	4	18	7	550	normal
147.32.84.168	4	8	10	32	normal
147.32.84.170	6	77	15	8,213	normal
147.32.84.191	20	1,178	29	22,002	botnet
147.32.84.192	28	974	64	22,000	botnet
147.32.84.193	26	1,196	46	22,000	botnet
147.32.84.194	17	199	39	1,304	normal
147.32.84.199	8	3	17	149	normal
…	…	…	…	…	…

Table 7. Example of Feature Extraction Results Based on Activity Time Analysis

Source IP Address	Dur_ mean	Dur_ min	Dur_ max	TimeGap_ mean	TimeGap_ min	TimeGap_ max	Label
147.32.84.172	0.00	0.00	0.00	2,536.67	0.00	6,151.00	normal
147.32.84.174	2,613.72	0.00	3,598.91	1,430.65	0.00	4,638.00	normal
147.32.84.19	51.41	0.00	3,564.16	15.11	0.00	235.00	normal
147.32.84.191	89.68	0.00	3,596.91	1.31	0.00	22.00	botnet
147.32.84.192	354.80	0.00	3,597.08	1.31	0.00	23.00	botnet
147.32.84.193	276.01	0.00	3,597.05	1.31	0.00	19.00	botnet
147.32.84.194	285.71	0.00	3,514.88	22.08	0.00	251.00	normal
147.32.84.199	18.48	0.00	2,753.42	178.10	0.00	1,622.00	normal
147.32.84.206	300.85	0.00	3,597.20	1.31	0.00	23.00	botnet
147.32.84.207	299.16	0.00	3,597.05	1.31	0.00	24.00	botnet
147.32.84.208	265.81	0.00	3,597.35	1.31	0.00	27.00	botnet
147.32.84.209	357.62	0.00	3,597.01	1.31	0.00	20.00	botnet
…	…	…	…	…	…	…	…

Table 8. Threshold Value for Data Filtering

Feature Name	Filtering Value	Feature Name	Filtering Value
Outdegree	> 3	Unique SrcPort Count	≥ 3
Weighted outdegree	> 1	Time Gap Mean	≤ 13
Unique Proto Count	≥ 3	MeanToTkts	$\neq 0$
Unique DstPort Count	≥ 3	MeanSrcBytes	$\neq 0$

Dur_max, TimeGap_mean, TimeGap_min, TimeGap_max and there is one feature as class label. Then, the data filtering process is carried out with the filter values used shown in Algorithm 5, namely filtering the values on the outdegree, weighted outdegree, unique Proto Count, Unique DstPort Count, Unique SrcPort Count, Time Gap Mean, MeanToTkts and MeanSrcBytes features. The specified filtration values are shown in Table 8.

The results of data filtering produce quite significant traffic reductions. The average data reduction reaches 87% on the CTU-13 dataset, 89% on the NCC dataset, and 94% on the NCC-2 dataset. Details of the reduction results in each dataset scenario are shown in Table 9. In this paper, the feature selection technique aims to optimize the machine learning model's performance by selecting the right features. The feature selection method used is ANOVA, with a reduction value of 25%. So, 11 features were used in the classification stage. The results of the feature selection are shown in Table 10.

In this paper, botnet activity detection uses a

Table 9. Result of Data Filtering

Scenario/ Sensor ID	Number of data record in Data Filtering by Threshold values								
	CTU-13			NCC			NCC-2		
	before	after	reduction (%)	before	after	reduction (%)	before	after	reduction (%)
1	542,093	108,419	80	342,741	41,129	88	723,635	28,945	96
2	392,301	82,383	79	252,364	20,189	92	861,588	60,311	93
3	359,987	50,398	86	240,781	24,078	90	448,309	31,382	93
4	131,177	10,494	92	66,014	9,242	86			
5	28,301	1,981	93	10,347	1,449	86			
6	72,345	15,916	78	46,628	5,595	88			
7	23,057	3,689	84	9,599	1,152	88			
8	333,816	40,058	88	252,163	17,651	93			
9	313,766	25,101	92	180,559	10,834	94			
10	151,256	19,663	87	89,919	9,891	89			
11	29,157	4,665	84	3,732	485	87			
12	59,552	5,955	90	33,622	4,707	86			
13	277,486	19,424	93	209,886	18,890	91			

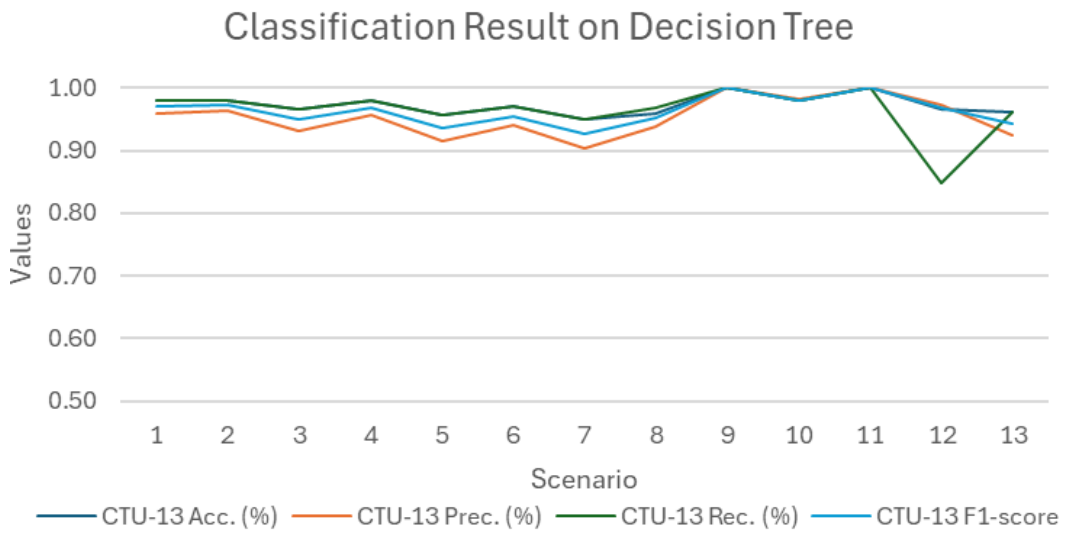
Table 10. Result of ANOVA Feature Selection

Dataset	Selection Feature using ANOVA			
	before	feature	after	feature
CTU-13 NCC NCC-2	15	UniqueProtoCount, UniqueDstPortCount, UniqueSrcPortCount, MeanTotPkts, MeanSrcBytes, InDegree, OutDegree, WeightedInDegree, WeightedOutDegree, Dur_mean, Dur_min, Dur_max, TimeGap_mean, TimeGap_min, TimeGap_max	11	MeanTotPkts, TimeGap_mean, Dur_mean, MeanSrcBytes, UniqueSrcPortCount, Dur_max, UniqueProtoCount, UniqueDstPortCount, Dur_min, WeightedInDegree, InDegree,

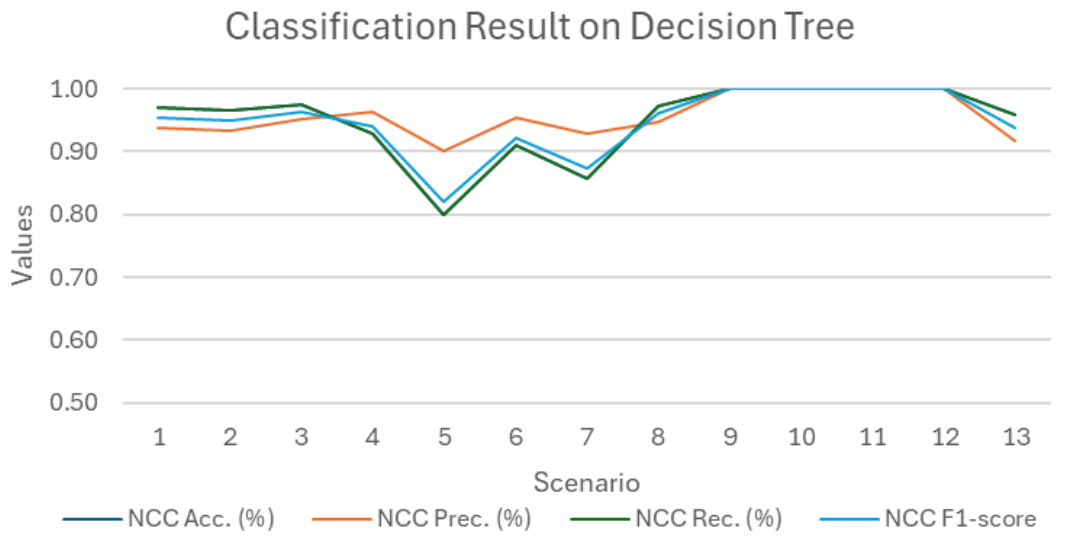
machine learning-based classification model. Four classification models were used: decision tree, k-NN, SVM, and random forest. The Decision Tree (DT) model classification results produce an average detection accuracy on the CTU-13 dataset of 0.9732, precision of 0.9532, recall of 0.9649, and F1-score value of 0.9631. The NCC dataset produces an average detection accuracy of 0.9489, precision of 0.9565, recall of 0.9489, and F1-score of 0.9475. The NCC2 dataset produces an average detection accuracy of 0.9923, precision of 0.9933, recall of recall of 0.9933, and F1-score of 0.9927. The results of the Decision tree model detection are shown in Fig. 3.

The results of the k-NN model classification produced an average detection accuracy on the CTU-13 dataset of 0.9545, precision of 0.9312, recall of 0.9306, and F1-score value of 0.9252. On the NCC dataset, it produced an average detection accuracy of 0.92, precision of 0.8672, recall of 0.92 and F1-score value of 0.8898. On the NCC2 dataset, it produced an average detection accuracy of 0.9813, precision of 0.9803, recall of 0.9780, and F1-score of 0.9770. The results of the k-NN model detection are shown in Fig. 4. The Support Vector Machine (SVM) model classification results produced an average detection accuracy on the CTU-13 dataset of 0.9611, precision of 0.9262, recall of 0.9588, and F1-score value of 0.9429. The NCC dataset produced an average detection accuracy of 0.9522, precision of 0.9190, recall of 0.9522, and F1-score value of 0.9355. While on the NCC2 dataset, it produced an average detection accuracy of 0.9753, precision of 0.9590, recall of 0.9753 and F1-score of 0.9753. The detection results of the Support Vector Machine (SVM) model are shown in Fig. 5. The classification results of the Random Forest (RF) model produce an average detection accuracy on the CTU-13 dataset of 0.9878, a precision of 0.9785, a recall of 0.9855 and an F1-score value of 0.9816.

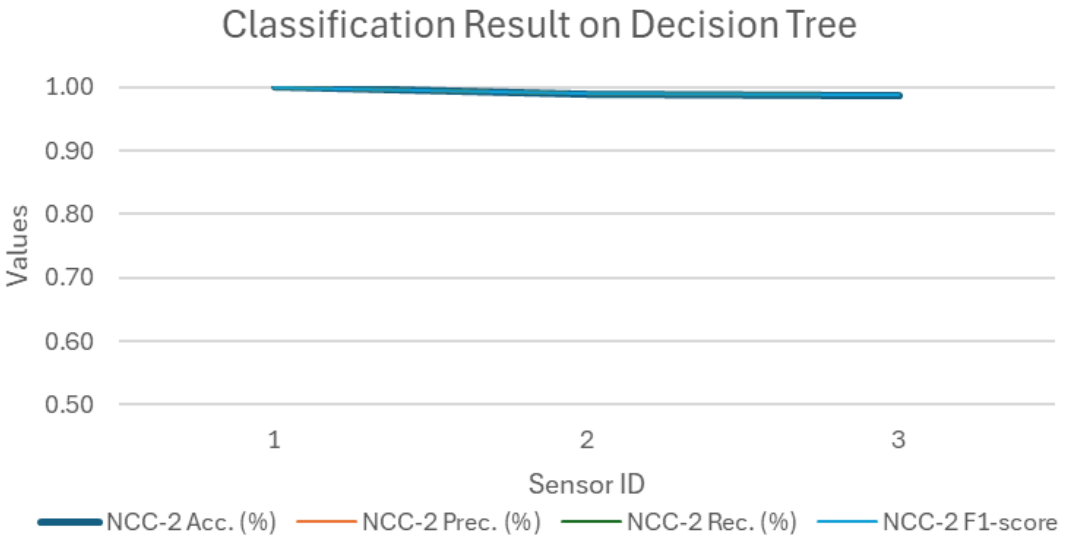
The NCC dataset produces an average detection accuracy of 0.9503, a precision of 0.9091, a recall of 0.9503, and an F1-score value of 0.9280. The NCC2 dataset produces an average detection accuracy of 0.9991, a precision of 0.9993, a recall of 0.9991, and an F1-score of 0.9992. The detection results of the Random Forest model are shown in Fig. 6. The results of the classification performance analysis on



(a)

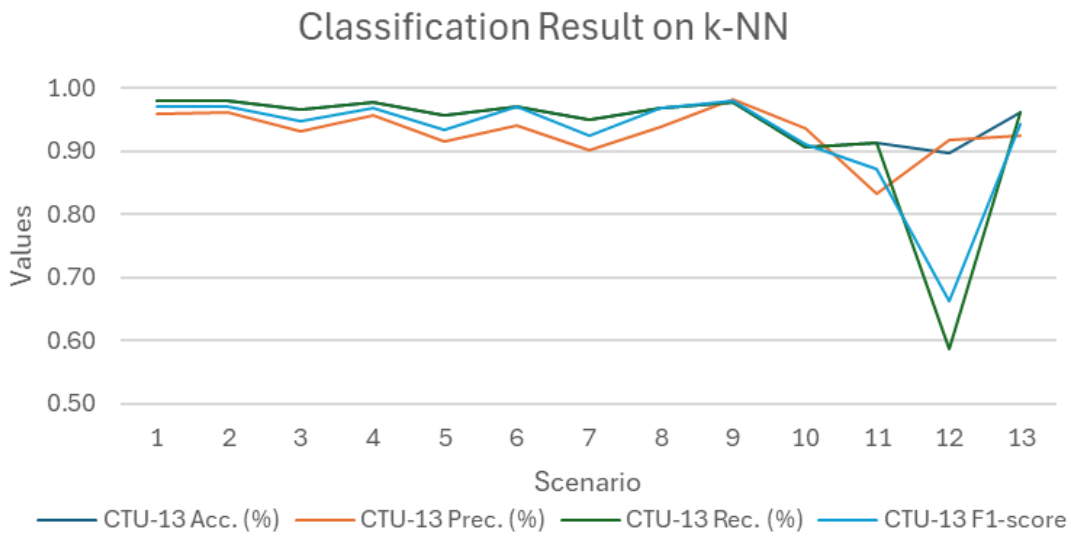


(b)

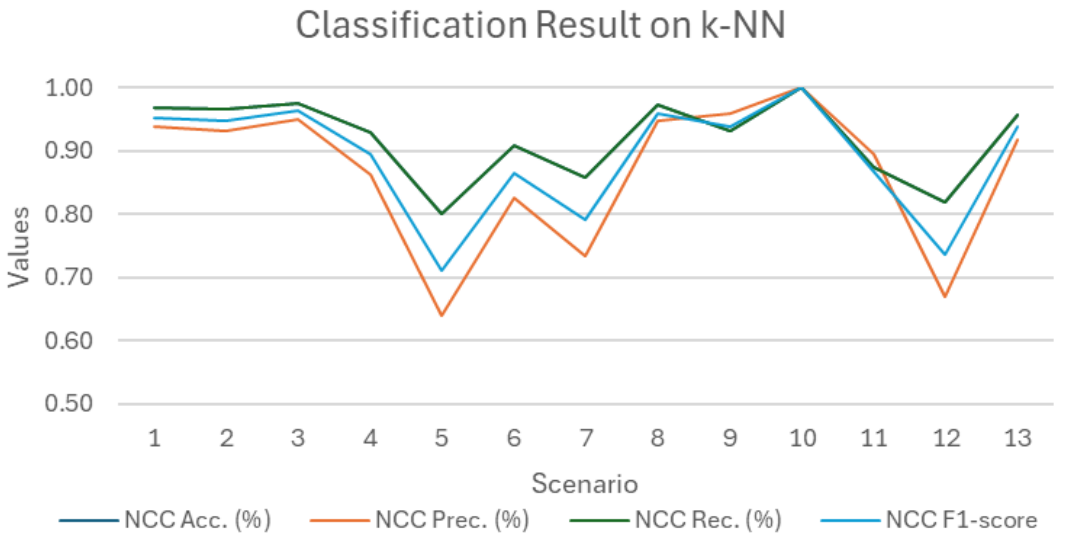


(c)

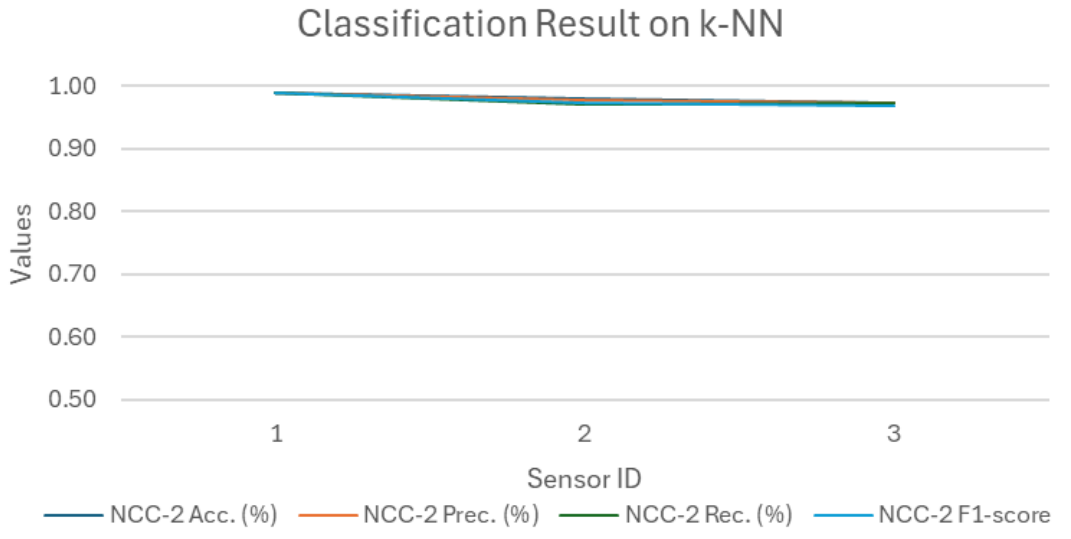
Figure. 3 Decision Tree Classification Result: (a) CTU-13 Dataset, (b) NCC Dataset, and (c) NCC-2 Dataset



(a)

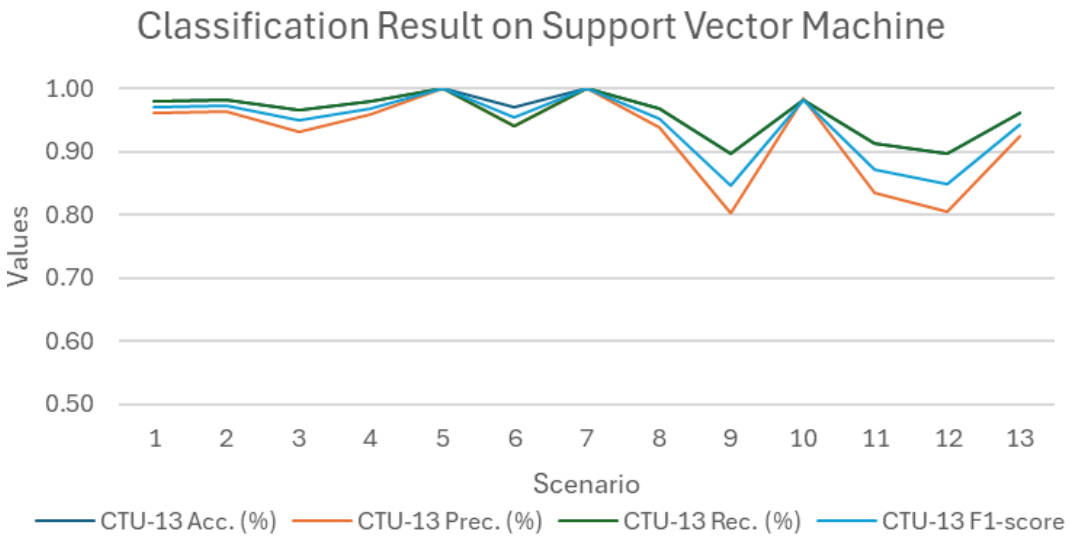


(b)

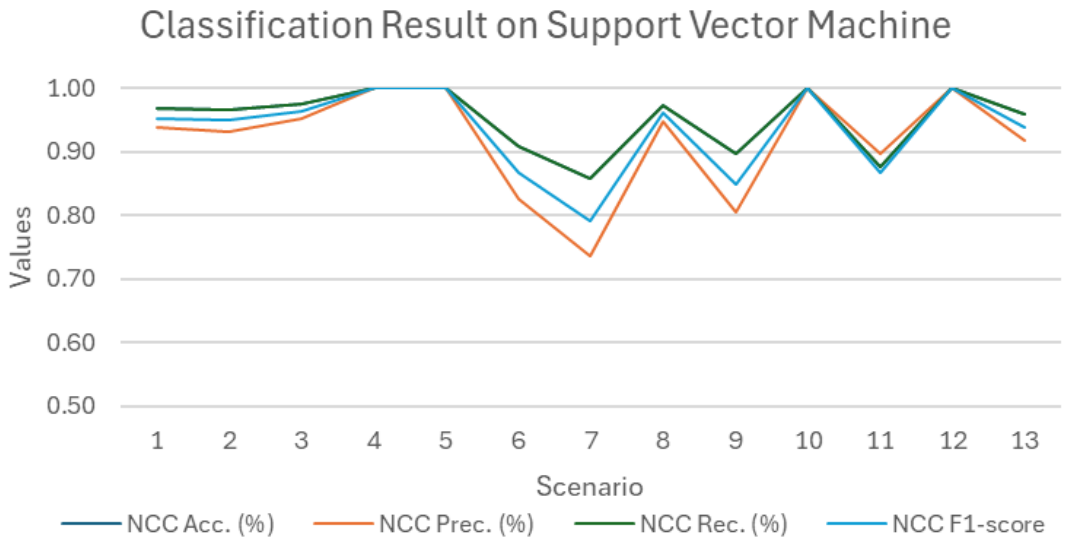


(c)

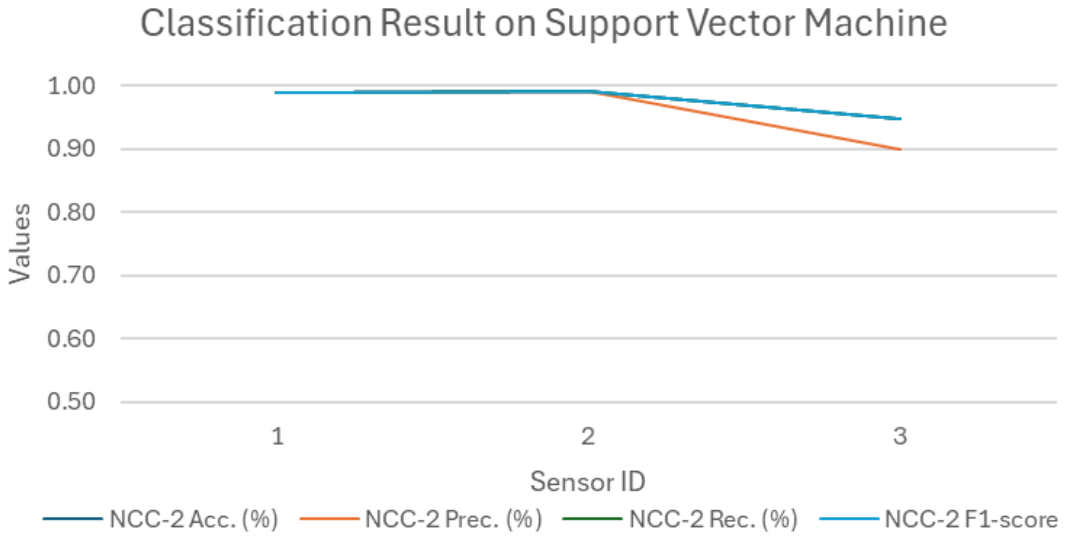
Figure. 4 k-NN Classification Result: (a) CTU-13 Dataset, (b) NCC Dataset, and (c) NCC-2 Dataset



(a)

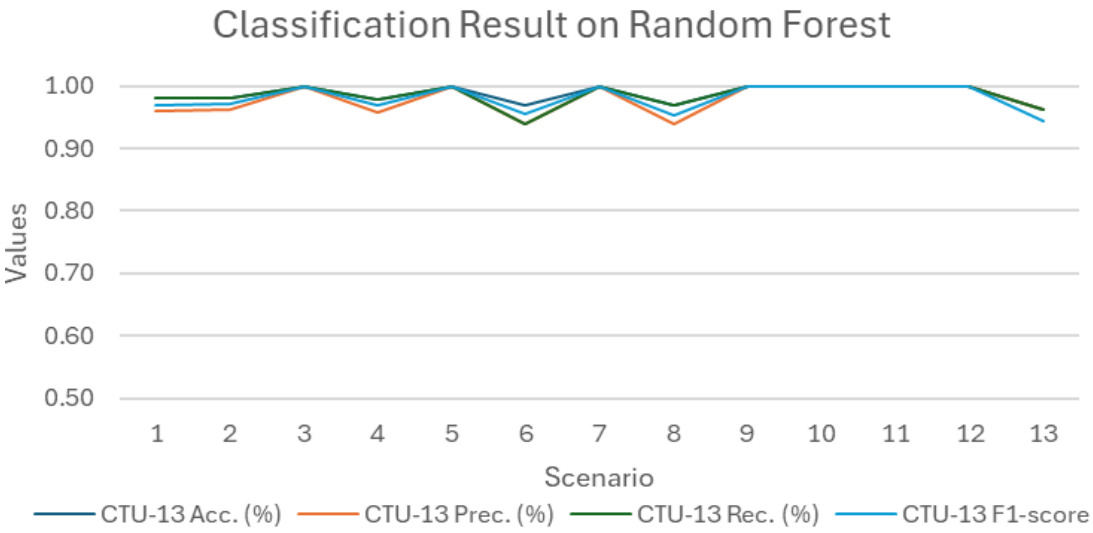


(b)

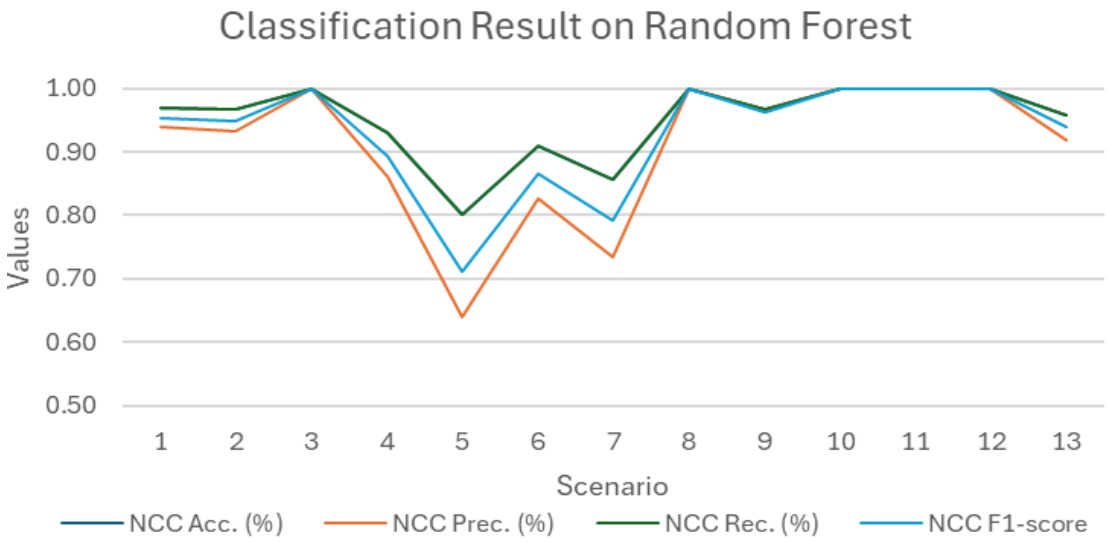


(c)

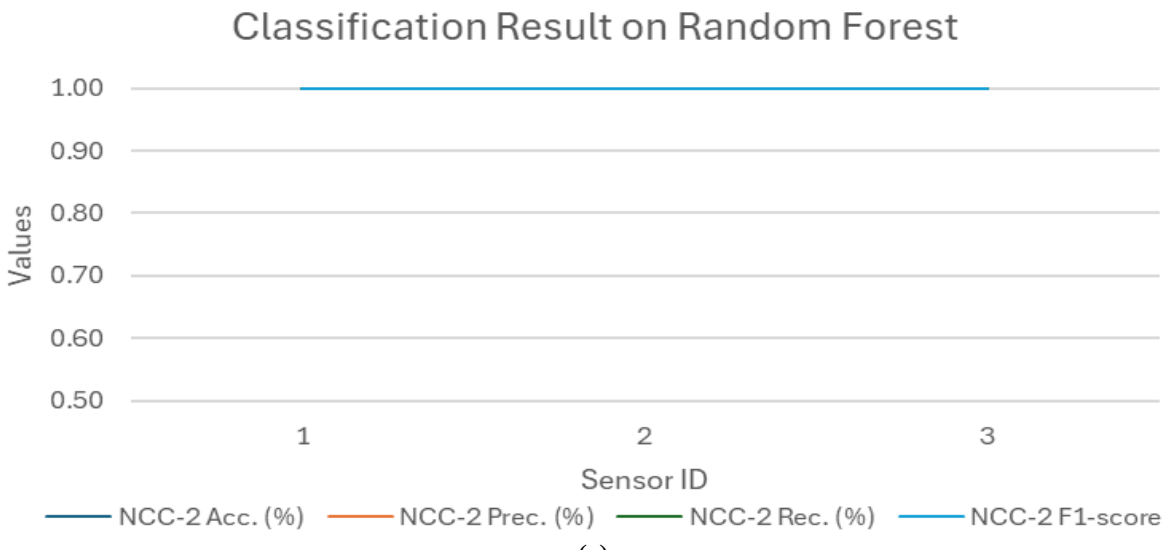
Figure. 5 Support Vector Machine Classification Result: (a) CTU-13 Dataset, (b) NCC Dataset, and (c) NCC-2 Dataset



(a)



(b)



(c)

Figure. 6 Random Forest Classification Result: (a) CTU-13 Dataset, (b) NCC Dataset, and (c) NCC-2 Dataset

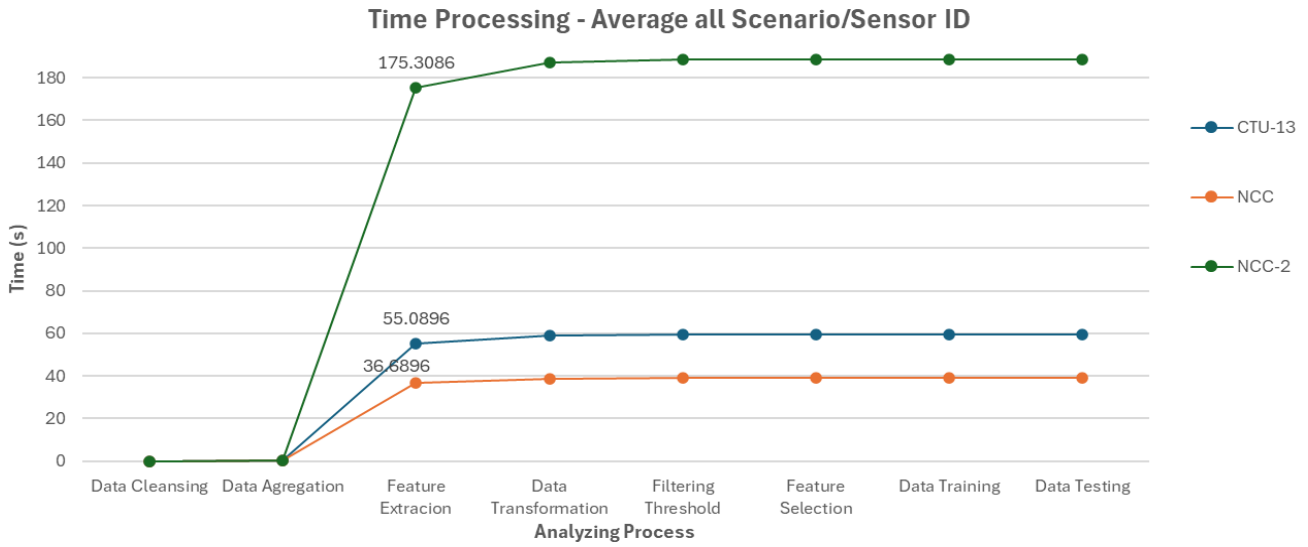


Figure. 7 Analysis of Time Processing

the CTU-13 dataset show that the decision tree classification method can achieve the highest accuracy, precision, recall, and F1-score values of 1.0 in scenarios 9 and 11. The *k*-NN classification method can achieve the highest results with an accuracy value of 0.9810, precision of 0.9630, recall of 0.9810, and F1-score of 0.9720 in scenario 2. The SVM method can achieve the highest accuracy, precision, recall, and F1-score values of 1.0 in scenarios 5 and 7. While the Random Forest classification method can achieve the highest accuracy, precision, recall and F1-score values of 1.0 in scenarios 3, 5, 7, 9, 10, 11 and 12. The results of the classification performance analysis on the NCC dataset show that the decision tree classification method can achieve the highest accuracy, precision, recall, and F1-score values of 1.0 in scenarios 9, 10, 11, and 12. The *k*-NN classification method can achieve the highest accuracy, precision, recall and F1-score values of 1.0 in scenarios 9, 10, 11, and 12 The SVM method can achieve the highest accuracy, precision, recall, and F1-score of 1.0 in scenarios 4, 5, 10, and 12. While the Random Forest classification method can achieve the highest accuracy, precision, recall and F1-score of 1.0 in scenarios 3,8,10, 11 and 12. The results of the classification performance analysis on the NCC-2 dataset show that the decision tree classification method can achieve the highest accuracy, precision, recall, and F1-score of 1.0 on Sensor ID 1. The *k*-NN classification method can achieve the highest results with an accuracy value of 0.9890, precision of 0.9890, recall of 0.9890, and F1-score of 0.9890 on Sensor ID 1. The SVM method can achieve the highest results with an accuracy value of 0.99, precision of 0.99, recall of 0.99, and F1-score of 0.99 on Sensor ID 2. Meanwhile, the Random Forest classification method can achieve an accuracy value of 0.9991, precision of 0.9993, recall of 0.9991, and F1-score of 0.9992 on all Sensor IDs. In this study, computational time analysis was carried out to see how fast the processing time of each stage is. The results of the detection time analysis show that the detection process on the NCC dataset with an average total time of 13 scenarios is 39.0617 seconds, the CTU-13 dataset is 59.5944 seconds, and the NCC-2 dataset is 188.6164 seconds. The results of the time analysis of each process in each dataset scenario are shown in Table 11. From the results of the computational time analysis, the feature extraction stage is the longest processing time, where in the CTU-13 dataset, it reaches 55.0896, NCC 36.6869 seconds, and NCC-2 175.3086 seconds. The total computation time of the three datasets is shown in Fig. 7.

4.2 Discussion

In this paper, the dynamic thresholding analysis technique as an approach to filtering data is one of the research contributions. In Table 8, the threshold value filters traffic data and successfully reduces traffic in each dataset scenario. The selection of the outdegree filter value is to see the characteristics of the botnet attack. In the attack activity, the botnet communicates with the bot server and client connected to the C&C network. In the attack, the botnet will send more than three hosts to infect new targets or when attacking command from the bot master. This botnet attack technique was introduced in the study [51]. Attacks carried out by botnets can only be done once or repeatedly. This is why attacks, such as spam or DoS, can be periodic and intense to one target.

Table 11. Time Analysis Results of Each Process in Each Dataset Scenario

Data set	Eval. (s)	Scenario / Sensor-ID												
		1	2	3	4	5	6	7	8	9	10	11	12	13
CTU-13	DC	0.0820	0.0946	0.0400	0.07111	0.0330	0.0953	0.0962	0.0962	0.0083	0.0380	0.0346	0.0603	0.0257
	DA	0.9420	0.3800	0.4970	0.50600	0.7230	0.1430	0.6140	0.1180	0.4550	0.9380	0.9220	0.4060	0.1440
	FE	139.3603	101.6179	103.5670	32.73351	6.2776	17.6541	5.3208	87.5842	81.0833	39.9948	8.2250	13.8959	71.2855
	DAT	9.4984	6.9264	6.9378	2.76008	0.5933	1.5860	0.5123	5.8531	5.9111	2.9300	0.5653	1.3273	5.0303
	FTH	1.4940	1.0104	0.9523	0.44911	0.1198	0.2254	0.1065	0.8547	0.6688	0.4039	0.0942	0.1990	0.6065
	FS	0.0379	0.0838	0.0685	0.08944	0.0564	0.0660	0.0686	0.0634	0.0387	0.0501	0.0495	0.0644	0.0965
	DTR	0.0068	0.0055	0.0048	0.00700	0.0053	0.0083	0.0088	0.0041	0.0053	0.0055	0.0050	0.0073	0.0070
	DTT	0.0025	0.0020	0.0020	0.00200	0.0018	0.0045	0.0033	0.0025	0.0023	0.0018	0.0023	0.0040	0.0015
	TT	151.4239	110.1206	112.0693	36.62605	7.8101	19.7826	6.7305	94.5762	88.1727	44.3621	9.8979	15.9642	77.1971
NCC	DC	0.0053	0.0878	0.0596	0.02378	0.0475	0.0187	0.0072	0.0354	0.0399	0.0788	0.0719	0.0642	0.0736
	DA	0.2020	0.0060	0.9860	0.84300	0.5770	0.3700	0.3510	0.0520	0.2550	0.4180	0.7830	0.6670	0.1320
	FE	93.5829	65.5015	70.6454	17.24764	2.5654	11.9237	2.3473	70.1898	46.8587	22.9093	1.0691	9.5344	56.3357
	DAT	4.0945	3.1427	3.2858	1.05770	0.1986	0.7375	0.2644	3.9857	3.3671	1.8567	0.2120	0.8415	3.8750
	FTH	0.5360	0.3839	0.3740	0.13955	0.0371	0.1336	0.0310	0.3826	0.2879	0.1642	0.0187	0.0864	0.3087
	FS	0.0882	0.0956	0.0926	0.00155	0.0367	0.0059	0.0985	0.0498	0.0610	0.0603	0.0915	0.0520	0.0926
	DTR	0.0065	0.0085	0.0083	0.00500	0.0048	0.0078	0.0053	0.0378	0.0048	0.0055	0.0075	0.0043	0.0045
	DTT	0.0060	0.0020	0.0028	0.00155	0.0483	0.0070	0.0065	0.0023	0.0018	0.0020	0.0075	0.0078	0.0020
	TT	98.5214	69.2281	75.4544	19.31777	3.5153	13.2042	3.1112	74.7353	50.8762	25.4947	2.2613	11.2574	60.8242
NCC-2	DC	0.0117	0.0703	0.0292										

	DA	0.5960	0.9540	0.1510
	FE	188.7097	222.0181	113.3857
	DAT	13.1282	15.2785	7.6653
	FTH	1.1138	1.5434	1.0414
	FS	0.0366	0.0144	0.0662
	DTR	0.0103	0.0075	0.0103
	DTT	0.0030	0.0028	0.0018
	TT	203.6092	239.8890	122.3509

Data Cleansing (DC); Data Agregation (DA); Feature Extraction (FE); Data Transformation (DAT); Filtering Threshold (FTH); Feature Selection (FS); Data Training (DTR); Data Testing (DTT); Total Time (TT)

This characteristic was introduced in the study [37]. So, for the Weighted outdegree feature, the value is determined more than once for one target. Botnets can perform several activities based on the use of communication protocols. In [52], was stated that the stages of botnet attacks can be carried out from 3 to 7 stages with different activities. So, the determination of the threshold value on the Unique Proto count, Unique DstPort Count, and Unique SrcPort Count features is ≥ 3. Periodically, botnet attacks have a time gap in activity used to repeat or carry out different attacks. Research in [37] found that the time gap between botnet attacks is not more than 13 seconds on SPAM and DoS attacks. So, this paper uses a value of ≤ 13 seconds in the Time Gap Mean feature. Research [48] states that botnet attacks have a number of packages and sizes in bytes of data. These packages and bytes of data are used to run illegal applications that can infect computer targets or attack instructions given by the bot master to the bot client. In this study, the threshold value used in the MeanToTkts and MeanSrcBytes features is $\neq 0$. The model proposed in this paper is a development of research [2, 21, 37], where previous research had weaknesses in precision performance results. The proposed model in this paper shows an increase in the performance of the detection model from precision measurements. The comparative results of previous studies are shown in Table 12.

The results of the comparison with previous studies, the proposed model has the highest results in testing the CTU-13 Dataset using the Random Forest classification algorithm with an accuracy of 98.78%, higher than the studies [2, 19, 20, 49, 50], but still lower than the studies [21, 37, 53]. The highest precision performance evaluation of the proposed model is on the Random Forest algorithm, which is 97.85%, higher than the studies [2, 20, 37], but lower than the studies [19, 21, 53]. The recall value achieved in this study is on the Random Forest classification algorithm with a value of 98.55%, higher than the studies [2, 19, 20] but lower than the studies [21, 37, 53]. Testing on the NCC dataset, the proposed model has the best performance value in the Support Vector Machine (SVM) classification algorithm with an accuracy of 95.22%, lower than the studies [2, 21, 37]. The SVM classification algorithm's precision value can reach 91.90%, higher than [2, 37] but lower than the study [21]. The evaluation of the recall value that the SVM algorithm can achieve is 95.22%, higher than [2, 37] but lower than the study [21]. On the NCC-2 dataset, the proposed model can achieve the highest value with the Random Forest classification algorithm with an accuracy of 99.91%, precision of 99.93%, and recall of 99.91%, higher than the study [2, 21, 37].

Research in [49] had lower performance than ours because the detection technique did not use optimization techniques in the feature selection section or at the classification stage. In research [53], it performed better than the proposed model in this paper because it only used nine basic features without tracing the basic behavior of botnet variant attacks. In addition, computational time and resources were not carried out. Research [20] had lower results than the proposed model because it only used eight manually selected features. Inappropriate selection can reduce the performance of the detection model.

Table 12. Comparison Proposed Model with Previous Studies

Authors	Dataset	Classification / Approach	Accuracy (%)	Precision (%)	Recall (%)	Computation time (s)
Khan et al. [49]	CTU-13	Naive Bayes	75.50	-	-	-
		ANN	93.80	-	-	
		Decision Tree	94.40	-	-	
Joshi, Ranjan and Bharti [53]	CTU-13	ANN	99.94	99.92	99.96	-
Letteri, Penna and Caianiello [20]	CTU-13	Decision Tree	97.54	97.75	97.26	-
Mathur et al. [50]	CTU-13 and ISOT	Logistic Regression	98.40	-	-	-
		Random Subspace	97.50	-	-	
		Randomizable Filtered	97.70	-	-	
		Multiclass Classifier	98.40	-	-	
		Random Committee	95.30	-	-	
Putra, MAR et.al [2]	CTU-13	Sequential Pattern Mining	96.73	0.08	91.03	461.69
	NCC		99.19	1.34	96.15	398.87
	NCC-2		97.22	0.08	96.08	1542.25
Naseri, Abidin and Eslahi [19]	CTU-13	C4.5	98.20	98.20	98.20	-
		Random Forest	98.20	98.20	98.20	-
		Naïve Bayes	97.00	97.00	97.00	-
		SVM	98.40	98.40	98.40	-
		Feedforward Neural Network (FNN)	98.50	98.50	98.50	-
Putra, MAR et al. [37]	CTU-13	XGB	99.93	8.77	100.00	-
		DT	99.93	8.93	100.00	
		RF	99.93	9.01	100.00	
		NB	99.98	11.11	40.00	-
		LR	99.99	0.00	0.00	
		k-NN	99.97	18.18	100.00	
		SVC	99.99	-	0.00	-
	NCC	XGB	99.99	57.14	80.00	
		DT	99.99	61.54	80.00	
		RF	99.99	58.33	70.00	-
		NB	99.98	32.00	80.00	
		LR	99.99	-	0.00	
		k-NN	99.96	-	0.00	-
		SVC	99.96	-	0.00	
	NCC-2	XGB	99.99	20.69	60.00	
		DT	100.00	70.00	70.00	-
		RF	100.00	63.64	70.00	
		NB	99.99	9.09	40.00	
		LR	100.00	-	0.00	-
		k-NN	100.00	45.45	50.00	
		SVC	99.99	-	0.00	
Hostiadi, et al. [21]	CTU-13	Decision Tree	99.27	98.68	99.27	28.8523
	NCC		99.03	98.26	98.96	17.7299

	NCC-2		98.87	97.90	98.87	44.3419
Proposed method	CTU-13	Decision Tree	97.32	95.32	96.49	59.5944
		k-NN	95.45	93.12	93.06	
		SVM	96.11	92.62	95.88	
		Random Forest	98.78	97.85	98.55	
	NCC	Decision Tree	94.89	95.65	94.89	39.0617
		k-NN	92.00	86.72	92.00	
		SVM	95.22	91.90	95.22	
		Random Forest	95.03	90.91	95.03	
	NCC-2	Decision Tree	99.23	99.33	99.23	188.6164
		k-NN	98.13	98.03	97.80	
		SVM	97.53	95.90	97.53	
		Random Forest	99.91	99.93	99.91	

In addition, the limitation of detection analysis is that it only analyzes botnet attacks in HTTP attacks. In fact, botnets can utilize many communication protocols based on the type of attack. Research in [50] has lower detection performance than the proposed model because it combines two different types of datasets and uses the same features in both datasets. The feature selection method used is CfsSubsetEval, which adopts the correlation measurement technique between features. However, the relationship between features and targets is linear. If the relationship between the independent variable (feature) and the dependent variable (target) is non-linear, this method may not be able to identify important features non-linearly. As a result, important features can be ignored, and the selected feature subset may not be optimal.

Research in [2] performs better than the proposed model based on accuracy, but the precision value is very low. This is because the analysis stage involves the analysis of pre-defined knowledge, which has been determined at the thresholding analysis stage of the botnet attack time. So, every botnet activity can be detected precisely and accurately. However, based on the determination of the activity time, many normal activities are categorized as botnet attacks, resulting in high False Positive and impacting low precision value obtained. In addition, the low precision value is caused by the characteristics of the imbalanced dataset.

Research in [19] has lower performance than the proposed model. This is because the introduced model uses histogram analysis, a frequency analysis that relies on the activity time segmentation process for 1 hour. In addition, the type of attack detected is only the type of attack that utilizes the HTTP protocol. Each suspected attack is compared, the correlation is measured, and each correlated activity is declared an attack. The idea of frequency analysis is adopted in the proposed model but combined with other analyses to improve detection performance. Research [37] has better detection results than the proposed model in this paper but has a very low precision value detection performance. This is due to graph-based analysis techniques combined with activity time analysis. Graph-based and activity time analysis can produce high accuracy, above 99% for the three datasets. However, the model has shortcomings for analysis in large amounts of data and is imbalanced, which impacts high False Positive detection results and precision values below 70%. Graph-based and activity time analysis techniques are adopted in the proposed model in this paper.

In [21] has detection results with higher accuracy but lower precision and recall values than the proposed model in this paper. Higher accuracy is influenced by two feature selection techniques, namely univariate as a measure of data bias and ANOVA as a feature ranking. In addition, feature selection based on importance, namely mandatory features and non-mandatory, is used for the feature selection process. Determining mandatory features manually causes features not to be selected using the ANOVA method. Not all mandatory features affect improving detection performance. In addition, using two feature selection methods causes a higher processing time than this paper's proposed model. Overall, the proposed model's experiment result shows good performance, with an average accuracy, precision, and recall value above 91% tested on three public datasets. However, in some comparisons, the classification model has a lower value.

In this paper, the proposed model uses data aggregation and dynamic filtering techniques to

optimize the detection process and become a research contribution. The method can reduce data significantly and show more efficiency in each process than research [2, 37]. This paper has the advantage of analyzing the presentation of computational time analysis of each process in detail, which has not been done in research [19, 20, 37, 49, 50, 53], and comparing the performance of several classification models carried out on three different datasets. Research [21] has lower computational results for the classification process but has not analyzed the time from the beginning of the data pre-processing process. In this paper, the model has the longest processing time at the feature extraction stage, using three analysis approaches: frequency, behavior graph, and activity time. In the feature extraction process, the highest memory used is between 90% and 96%, with the computer memory specifications being 16 GB. At the same time, the average memory usage in the classification process is 5% to 8%.

5. Conclusions

Research on botnet detection has challenges for researchers and requires appropriate techniques. This paper proposes a botnet activity detection model through hybrid analysis, namely analyzing bot attack characteristics such as activity frequency, node-based graph, and activity time. Data aggregation techniques are carried out to obtain optimal analysis by analyzing activity groups based on source IP addresses and combining feature extraction results. Dynamic filtering techniques, which determine threshold values according to attack characteristics, are carried out to reduce data and computing resources. In addition, dynamic filtering techniques can accelerate the detection process in classification algorithms. To detect, four classification algorithms, Decision Tree, *k*-NN, Support Vector Machine, and Random Forest, are used in the botnet attack detection model. The experiment result showed that the proposed model was able to detect botnet attacks well in all three datasets with an average detection accuracy above 92%, precision above 86.72%, recall above 92%, F1-Score above 88.89%, and the best computing time on the NCC dataset, which was 39.0617 seconds. It was tested on three datasets of botnet attacks, which have specific types and characteristics, namely NCC-2, CTU-13, and NCC datasets. The best classification algorithm in detecting botnet attacks is Random Forest on the NCC-2 dataset with a detection accuracy of 99.91%, precision of 99.93%, recall of 99.91%, and F1-score of 99.92%. The best computation time is on the NCC dataset, with an overall detection computation time of 39.0617 seconds. Comparison with previous studies shows that the proposed model has the best performance on the CTU-13 dataset using the Random Forest algorithm with an accuracy of 98.78%, higher than several previous studies such as [2, 19, 20, 49, 50], but still lower than [2, 21, 37]. On the NCC dataset, the SVM algorithm provides the highest accuracy of 95.22%, while on the NCC-2 dataset, the Random Forest algorithm achieves the highest accuracy, precision, and recall of 99.91%, 99.93%, and 99.91% respectively, higher than the studies [2, 22, 39]. Regarding computing time, the proposed model performs better than research [2, 37].

Data aggregation techniques in large data analysis on real-time detection models have challenges. In the future, the research will improve by analyzing data aggregation techniques with time series data based on network traffic flows. An example is combining data aggregation analysis with data sliding techniques. This improvement can find the best time segment to analyze in the adaptive segment area. This analysis technique can help system security administrators streamline analysis time and can be the basis for developing an intrusion detection or anti-malware model.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization DPH, TA and MARP; writing—review and editing DPH, TA and MARP; writing—original draft preparation DPH, YPA and IMDS; methodology DPH, MARP and TA; software DPH, IMDS and GAP; analysis DPH, YPA, GAP and MARP; investigation DPH, TA, and MARP; validation DPH and YPA; visualization DPH.

Acknowledgments

[1] R. Alguliyev, R. Aliguliyev, and L. Sukhostat, “Radon transform based malware classification in cyber-physical system using deep learning”, *Results Control Optim.*, Vol. 14, No. January, p. 100382, 2024, doi: 10.1016/j.rico.2024.100382.

[2] M. A. R. Putra, T. Ahmad, D. P. Hostiadi, and R. M. Ijtihadie, “Botnet sequential activity detection with hybrid analysis”, *Egypt. Informatics J.*, Vol. 25, No. January, p. 100440, 2024, doi: 10.1016/j.eij.2024.100440.

[3] S. Ali, R. Ghazal, N. Qadeer, O. Saidani, F. Alhayan, A. Masood, R. Saleem, M. A. Khan, and D. Gupta, “A novel approach of botnet detection using hybrid deep learning for

enhancing security in IoT networks", *Alexandria Eng. J.*, Vol. 103, No. June, pp. 88-97, 2024, doi: 10.1016/j.aej.2024.05.113.

[4] G. Chandana Swathi, G. Kishor Kumar, and A. P. Siva Kumar, "Ensemble classification to predict botnet and its impact on IoT networks", *Meas. Sensors*, Vol. 33, No. March, p. 101130, 2024, doi: 10.1016/j.measen.2024.101130.

[5] J. Song, S. Choi, J. Kim, K. Park, C. Park, J. Kim, and I. Kim, "A study of the relationship of malware detection mechanisms using Artificial Intelligence", *ICT Express*, Vol. 10, No. 3, pp. 632-649, 2024, doi: 10.1016/j.icte.2024.03.005.

[6] K. E. Mwangi, S. Masupe, and J. Mandu, "Modelling malware propagation on the internet of things using an agent-based approach on complex networks", *Jordanian J. Comput. Inf. Technol.*, Vol. 6, No. 1, pp. 26-40, 2020, doi: 10.5455/jjcit.71-1568145650.

[7] M. Alazab, "Automated malware detection in mobile app stores based on robust feature generation", *Electron.*, Vol. 9, No. 3, 2020, doi: 10.3390/electronics9030435.

[8] Y. Xing, H. Shu, H. Zhao, D. Li, and L. Guo, "Survey on Botnet Detection Techniques: Classification, Methods, and Evaluation", *Math. Probl. Eng.*, Vol. 2021, 2021, doi: 10.1155/2021/6640499.

[9] M. Al-Sarem, F. Saeed, E. H. Alkhammash, and N. S. Alghamdi, "An aggregated mutual information based feature selection with machine learning methods for enhancing iot botnet attack detection", *Sensors*, Vol. 22, No. 1, 2022, doi: 10.3390/s22010185.

[10] S. Li, Y. Cao, S. Liu, Y. Lai, Y. Zhu, and N. Ahmad, "HDA-IDS: A Hybrid DoS Attacks Intrusion Detection System for IoT by using semi-supervised CL-GAN", *Expert Syst. Appl.*, Vol. 238, No. PF, p. 122198, 2024, doi: 10.1016/j.eswa.2023.122198.

[11] A. Nazir, J. He, N. Zhu, A. Wajahat, X. Ma, F. Ullah, S. Qureshi, and M. S. Pathan, "Advancing IoT security: A systematic review of machine learning approaches for the detection of IoT botnets", *J. King Saud Univ. - Comput. Inf. Sci.*, Vol. 35, No. 10, p. 101820, 2023, doi: 10.1016/j.jksuci.2023.101820.

[12] I. Apostol, A. D. Tica, and V. V. Patriciu, "Design and implementation of a novel hybrid botnet", In: *Proc. of 2022 14th Int. Conf. Electron. Comput. Artif. Intell. ECAI 2022*, 2022, doi: 10.1109/ECAI54874.2022.9847442.

[13] H. Y. Alshaeaa and Z. M. Ghadhban, "Developing a hybrid feature selection method to detect botnet attacks in IoT devices", *Kuwait J. Sci.*, Vol. 51, No. 3, p. 100222, 2024, doi: 10.1016/j.kjs.2024.100222.

[14] O. A.M. and J. R.G., "Towards Building an Improved Botnet Detection Model in Highly Imbalance Towards Building an Improved Botnet Detection Model in Highly Imbalance Botnet Dataset-A Methodological Framework", *Mejast.Com*, Vol. 3, No. March, pp. 33-38, 2020.

[15] M. Asadi, M. A. Jabraeil Jamali, S. Parsa, and V. Majidnezhad, "Detecting botnet by using particle swarm optimization algorithm based on voting system", *Futur. Gener. Comput. Syst.*, Vol. 107, pp. 95-111, 2020, doi: 10.1016/j.future.2020.01.055.

[16] T. S. Naseri and F. S. Gharehchopogh, "A Feature Selection Based on the Farmland Fertility Algorithm for Improved Intrusion Detection Systems", *J. Netw. Syst. Manag.*, Vol. 30, No. 3, pp. 1-27, 2022, doi: 10.1007/s10922-022-09653-9.

[17] A. Affinito, S. Zinno, G. Stanco, A. Botta, and G. Ventre, "The evolution of Mirai botnet scans over a six-year period", *J. Inf. Secur. Appl.*, Vol. 79, No. October, p. 103629, 2023, doi: 10.1016/j.jisa.2023.103629.

[18] S. Chowdhury, M. Khanzadeh, R. Akula, F. Zhang, S. Zhang, H. Medal, M. Marufuzzaman, and L. Bian, "Botnet detection using graph-based feature clustering", *J. Big Data*, Vol. 4, No. 1, 2017, doi: 10.1186/s40537-017-0074-7.

[19] M. Eslahi, W. Z. Abidin, and M. V. Naseri, "Correlation-based HTTP Botnet detection using network communication histogram analysis", In: *Proc. of 2017 IEEE Conf. Appl. Inf. Netw. Secur. AINS 2017*, Vol. 2018-Janua, pp. 7-12, 2017, doi: 10.1109/AINS.2017.8270416.

[20] I. Letteri, G. Della Penna, and P. Caianiello, "Feature selection strategies for http botnet traffic detection", In: *Proc. of 4th IEEE Eur. Symp. Secur. Priv. Work. EUROS PW 2019*, pp. 202-210, 2019, doi: 10.1109/EuroSPW.2019.00029.

[21] D. P. Hostiadi, T. Ahmad, M. A. R. Putra, G. A. Pradipta, P. D. W. Ayu, and M. Liandana, "A New Approach of Botnet Activity Detection Models Using Combination of Univariate and ANOVA Feature Selection Techniques", *Int. J. Intell. Eng. Syst.*, Vol. 17, No. 3, pp. 485-502, 2024, doi: 10.22266/ijies2024.0630.38.

[22] R. Abrantes, P. Mestre, and A. Cunha, "Exploring Dataset Manipulation via Machine Learning for Botnet Traffic", *Procedia Comput. Sci.*, Vol. 196, No. 2021, pp. 133-141, 2021, doi: 10.1016/j.procs.2021.11.082.

[23] Z. Ismail, A. Jantan, M. N. Yusoff, and M. U.

Kiru, "The effects of feature selection on the classification of encrypted botnet", *J. Comput. Virol. Hacking Tech.*, Vol. 17, No. 1, pp. 61-74, 2021, doi: 10.1007/s11416-020-00367-7.

[24] R. Nair and A. Bhagat, "Feature selection method to improve the accuracy of classification algorithm", *Int. J. Innov. Technol. Explor. Eng.*, Vol. 8, No. 6, pp. 124-127, 2019.

[25] M. Alshamkhany, W. Alshamkhany, M. Mansour, M. Khan, S. Dhou, and F. Aloul, "Botnet Attack Detection using Machine Learning", In: *Proc. of 2020 14th Int. Conf. Innov. Inf. Technol. IIT 2020*, No. February, pp. 203-208, 2020, doi: 10.1109/IIT50501.2020.9299061.

[26] R. F. M. Dollah, M. A. Faizal, F. Arif, M. Z. Mas'ud, and L. K. Xin, "Machine learning for HTTP botnet detection using classifier algorithms", *J. Telecommun. Electron. Comput. Eng.*, Vol. 10, No. 1-7, pp. 27-30, 2018.

[27] R. Younisse, M. Alkasassbeh, M. Almseidin, and H. Abdi, "an Early Detection Model for Kerberoasting Attacks and Dataset Labeling", *Jordanian J. Comput. Inf. Technol.*, Vol. 9, No. 1, pp. 1-10, 2023, doi: 10.5455/jjcit.71-1661423262.

[28] S. Niveditha, R. R. Prianka, K. Sathya, S. Shreyanth, N. Subramani, B. Deivasigamani, and S. Karthikeyan, "Predicting Malware Classification and Family using Machine Learning: A Cuckoo Environment Approach with Automated Feature Selection", *Procedia Comput. Sci.*, Vol. 235, No. 2023, pp. 2434-2451, 2024, doi: 10.1016/j.procs.2024.04.230.

[29] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods", *Comput. Secur.*, Vol. 45, pp. 100-123, 2014, doi: 10.1016/j.cose.2014.05.011.

[30] D. P. Hostiadi and T. Ahmad, "Dataset for Botnet Group Activity with Adaptive Generator", *Data Br.*, Vol. 38, p. 107334, 2021, doi: 10.1016/j.dib.2021.107334.

[31] M. A. R. Putra, D. P. Hostiadi, and T. Ahmad, "Botnet dataset with simultaneous attack activity", *Data Br.*, Vol. 45, 2022, doi: 10.1016/j.dib.2022.108628.

[32] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems", In: *Proc. of Big Data Technologies and Applications*, pp. 117-135, 2021.

[33] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)", In: *Proc. of 2015 Military Communications and Information Systems Conference (MilCIS)*, pp. 1-6, 2015, doi: 10.1109/MilCIS.2015.7348942.

[34] N. Moustafa, G. Creech, and J. Slay, "Big Data Analytics for Intrusion Detection System: Statistical Decision-Making Using Finite Dirichlet Mixture Models", *Data Analytics and Decision Support for Cybersecurity: Trends, Methodologies and Applications*, I. Palomares Carrascosa, H. K. Kalutarage, and Y. Huang, Eds. Cham: Springer International Publishing, pp. 127-156, 2017, doi: 10.1007/978-3-319-59439-2_5.

[35] V. Quezada, F. Astudillo-Salinas, L. Tello-Oquendo, and P. Bernal, "Real-time bot infection detection system using DNS fingerprinting and machine-learning", *Comput. Networks*, Vol. 228, No. October 2022, p. 109725, 2023, doi: 10.1016/j.comnet.2023.109725.

[36] F. Ullah, S. Ullah, G. Srivastava, and J. C.-W. Lin, "IDS-INT: Intrusion detection system using transformer-based transfer learning for imbalanced network traffic", *Digit. Commun. Networks*, 2023, doi: 10.1016/j.dcan.2023.03.008.

[37] M. A. R. Putra, T. Ahmad, D. P. Hostiadi, R. M. Ijtihadie, and P. Maniriho, "Botnet Attack Analysis through Graph Visualization", *Int. J. Intell. Eng. Syst.*, Vol. 17, No. 1, 2024, doi: 10.22266/ijies2024.0229.75.

[38] S. Hosseini, A. E. Nezhad, and H. Seilani, "Botnet detection using negative selection algorithm, convolution neural network and classification methods", *Evol. Syst.*, Vol. 13, No. 1, pp. 101-115, 2022, doi: 10.1007/s12530-020-09362-1.

[39] M. Aamir and S. M. Ali Zaidi, "Clustering based semi-supervised machine learning for DDoS attack classification", *J. King Saud Univ. - Comput. Inf. Sci.*, Vol. 33, No. 4, pp. 436-446, 2021, doi: 10.1016/j.jksuci.2019.02.003.

[40] S. Chowdhury, M. Khanzadeh, R. Akula, F. Zhang, S. Zhang, H. Medal, M. Marufuzzaman, and L. Bian, "Botnet detection using graph-based feature clustering", *J. Big Data*, Vol. 4, No. 1, 2017, doi: 10.1186/s40537-017-0074-7.

[41] M. A. R. Putra, T. Ahmad, and D. P. Hostiadi, "Analysis of Botnet Attack Communication Pattern Behavior on Computer Networks", *Int. J. Intell. Eng. Syst.*, Vol. 15, No. 4, pp. 533-544, 2022, doi: 10.22266/ijies2022.0831.48.

[42] D. P. Hostiadi, T. Ahmad, and W. Wibisono, "A New Approach of Botnet Activity Detection Model based on Time Periodic Analysis", In:

Proc. of CENIM 2020 Comput. Eng. Network, Intell. Multimed. 2020, No. Cenim 2020, pp. 315-320, 2020, doi: 10.1109/CENIM51130.2020.9297846.

[43] D. A. Dung, "Slow HTTP DDoS High-Speed Detection Method for Slow HTTP DDoS Attacks", In: *Proc. of 2023 Computer Security Symposium (CSS) in Japanese*, 2F1-1, 2023.

[44] N. Gavric, "Introducing the Slowloris E-DoS Attack : a Threat Arising From Vulnerabilities in the FTP and SSH Protocols", Preprint, 2024, doi: 10.21203/rs.3.rs-4889002/v1.

[45] N. Jalal, M. A. Noor, A. A. Siddiqui, Z. Mubarak, and M. A. Yaqub, "Analysis of DoS Attack Using Machine Learning", *Journal of Computers and Intelligent Systems*, Vol. 2, No. 2, 2024.

[46] A. Alabdulatif, N. N. Thilakarathne, and M. Aashiq, "Machine Learning Enabled Novel Real-Time IoT Targeted DoS/DDoS Cyber Attack Detection System", *Comput. Mater. Contin.*, Vol. 80, No. 3, pp. 3655-3683, 2024, doi: 10.32604/cmc.2024.054610.

[47] C. Li, Y. Zhang, W. Wang, Z. Liao, and F. Feng, "Botnet Detection with Deep Neural Networks Using Feature Fusion", In: *Proc. of 2022 Int. Semin. Comput. Sci. Eng. Technol. SCSET 2022*, pp. 255-258, 2022, doi: 10.1109/SCSET55041.2022.00066.

[48] G. Wu, X. Wang, Q. Lu, and H. Zhang, "Bot-DM: A dual-modal botnet detection method based on the combination of implicit semantic expression and graphical expression", *Expert Syst. Appl.*, Vol. 248, No. January, p. 123384, 2024, doi: 10.1016/j.eswa.2024.123384.

[49] R. U. Khan, X. Zhang, R. Kumar, A. Sharif, N. A. Golilarz, and M. Alazab, "An adaptive multi-layer botnet detection technique using machine learning classifiers", *Appl. Sci.*, Vol. 9, No. 11, 2019, doi: 10.3390/app9112375.

[50] L. Mathur, M. Raheja, and P. Ahlawat, "Botnet Detection via mining of network traffic flow", *Procedia Comput. Sci.*, Vol. 132, pp. 1668-1677, 2018, doi: 10.1016/j.procs.2018.05.137.

[51] M. Singh, M. Singh, and S. Kaur, "Issues and challenges in DNS based botnet detection: A survey", *Comput. Secur.*, Vol. 86, pp. 28-52, 2019, doi: 10.1016/j.cose.2019.05.019.

[52] D. P. Hostiadi, T. Ahmad, and W. Wibisono, "A New Approach to Detecting Bot Attack Activity Scenario", In: *Proc. of the 12th International Conference on Soft Computing and Pattern Recognition (SoCPaR 2020)*, pp. 823-835, 2021.

[53] C. Joshi, R. K. Ranjan, and V. Bharti, "A Fuzzy Logic based feature engineering approach for Botnet detection using ANN", *J. King Saud Univ. - Comput. Inf. Sci.*, Vol. 34, No. 9, pp. 6872-6882, 2021, doi: 10.1016/j.jksuci.2021.06.018.