applied sciences

MDPI

Article

An Adaptive Multi-Layer Botnet Detection Technique Using Machine Learning Classifiers

Riaz Ullah Khan [1,*], Xiaosong Zhang [1], Rajesh Kumar [1], Abubakar Sharif [1], Noorbakhsh Amiri Golilarz [1] and Mamoun Alazab [2]

1 Center of Cyber Security, School of Computer Science & Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; jhonsonzxs@uestc.edu.cn (X.Z.); rajakumarlohano@gmail.com (R.K.); engr.abubakarsharif@gmail.com (A.S.); noorbakhsh.amiri90@gmail.com (N.A.G.)

2 College of Engineering, IT and Environment, Charles Darwin University, Casuarina 0810, Australia; mamoun.alazab@cdu.edu.au

* Correspondence: rerukhan@gmail.com; Tel.: +86-155-2076-3595

Received: 19 March 2019; Accepted: 24 April 2019; Published: 11 June 2019

Abstract: In recent years, the botnets have been the most common threats to network security since it exploits multiple malicious codes like a worm, Trojans, Rootkit, etc. The botnets have been used to carry phishing links, to perform attacks and provide malicious services on the internet. It is challenging to identify Peer-to-peer (P2P) botnets as compared to Internet Relay Chat (IRC), Hypertext Transfer Protocol (HTTP) and other types of botnets because P2P traffic has typical features of the centralization and distribution. To resolve the issues of P2P botnet identification, we propose an effective multi-layer traffic classification method by applying machine learning classifiers on features of network traffic. Our work presents a framework based on decision trees which effectively detects P2P botnets. A decision tree algorithm is applied for feature selection to extract the most relevant features and ignore the irrelevant features. At the first layer, we filter non-P2P packets to reduce the amount of network traffic through well-known ports, Domain Name System (DNS). query, and flow counting. The second layer further characterized the captured network traffic into non-P2P and P2P. At the third layer of our model, we reduced the features which may marginally affect the classification. At the final layer, we successfully detected P2P botnets using decision tree Classifier by extracting network communication features. Furthermore, our experimental evaluations show the significance of the proposed method in P2P botnets detection and demonstrate an average accuracy of 98.7%.

Keywords: botnet detection; anomaly detection; network traffic identification; machine learning

1. Introduction

Nowadays, the network environments are becoming more and more complex and so the security problems. A Botnet is a system of customized bots (computers) controlled remotely by a botmaster. A botnet can perform different noxious exercises, for example, sending spam messages, phishing, click misrepresentation, Distributed Denial of Service DDoS and spreading malicious programming. To viably oversee a botnet, the botmaster develops a framework of a correspondence channel to send directions to the Bots and to get results from them [1]. The fundamental contrast between a botnet and other malicious code is the structure utilized in the command and control (C & C) [2]. Compared to other malware programs which are being used to perform malicious conduct exclusively, a botnet functions as a gathering of contaminated hosts dependent on the C & C correspondence channel. A botnets system can be ordered into two principal classes dependent on the C & C foundation: brought together and decentralized C & C [3]. In incorporated botnets, the botmaster typically utilizes the C

& C server to send a direction to the bots. Figure 1 gives a brief overview of the centralized Internet Relay Chat IRC/HTTP traffic and decentralized Peer to Peer P2P botnet traffic. As the higher degree of concealment of botnet command and control server (C & C server), the attacker uses unknown programs to create intrusion in a large-scale network. Moreover, the botnets initiate almost all of the DDoS attacks and 80% to 90% of the spam attacks [4]. Consequently, the botnet has become a significant threat to network security. Early botnets were easily detected due to commonly used IRC and HTTP as a communication protocol, with a single failure point. Recently, most of the botnets use P2P technology to create C & C mechanisms to enhance network traffic concealment. As compared to botnets with IRC and HTTP protocols, the P2P botnets without central nodes have greater threat and concealment. Therefore, the P2P botnets are increasingly favored by attackers. The P2P botnet detection has also become a hot research area in the field of cyber security. At present, the P2P applications have caused the explosive growth of Internet traffic, which is a massive challenge in terms of data storage and real-time analysis. Thus, the network of non-P2P traffic filtering is particularly important.

Recent research works on botnets among our surveyed literature focuses mainly on designing systems to detect command and control (C & C) botnets, where many bot-infected machines are controlled and coordinated by few entities to carry out malicious activities [5]. Those systems need to learn decision boundaries between human and bot activities. Therefore, Machine Learning-based classifiers are at the core of those systems and are often trained by labeled data in supervised learning environments. The most popular classifier is support vector machines (SVMs) with different kernels, while spatial-temporal time series analysis and probabilistic inferences are also notable techniques employed in ML-based classifiers. Clustering is mostly used in natural language processing (NLP), to build a large-scale system to identify bot queries [6]. In botnet detection literature, the core assumption widely shared i.e., Botnet behaviors are different and distinguishable from a legitimate human user, e.g., human behaviors are more complex [7].

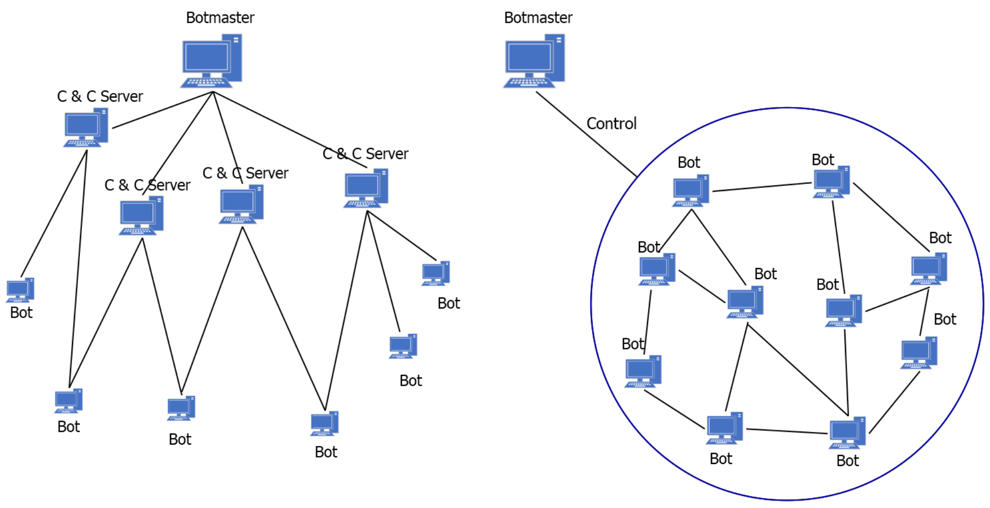
Research has been done on the state-of-the-art machine learning models of the network to simulate real-time network traffic and create honeypots. The ground reality is often heuristic or a combination, labeled by human experts, for example, the game masters' visual inspections serve to detect bots in online games [8]. In retrospect, the evolution of botnet detection is clear from earlier and more straightforward uses of classification techniques such as clustering and Naive Bayes, the research focus has been expanded from the last step of classification to the important preceding step of constructing suitable metrics, which measures and distinguishes bot-based and human-based activities [6,7].

This paper aims to examine the features and strategies to detect the botnets. The main contributions of this paper are as follows:

- This article presents a multi-layer approach to classify network traffic (P2P botnet traffic and non-P2P traffic) and identify botnets by applying machine learning classifier on network features such as port filtering, DNS query, and flow counting.
- Our work presents a P2P botnet detection framework based on a decision tree algorithm for feature selection to extract the most relevant features and ignore the irrelevant features.
- At the first layer, we filter non-P2P packets to reduce the amount of network traffic by applying port filtering using well-known ports, DNS query, and flow counting. The second layer further classified the captured network traffic into two classes such as non-P2P and P2P. At the third layer of our model, we reduced the features which may marginally affect the classification. At the final layer, we successfully detected P2P botnets using decision tree Classifier by extracting network communication features.
- The proposed technique of this study covers the limitations of single stage botnet detection, e.g., class imbalance.
- The accuracy of our model is 98.7% and the threshold of false alarm rate was reduced to 3. Furthermore, our experiments also demonstrate that the accuracy of proposed framework was improved up to 99% however at the expense of false reporting of benign files as botnets as well as false reporting of botnet as benign. We also consider the factor if benign files also send out search

requests consistently so benign files may be reported as botnets. Additionally, it was observed the accuracy might be improved by increasing the epochs of deep learning algorithms at the expense of more execution cost.

- To validate the performance of our proposed technique, we performed the experiments on diverse datasets and the results are compared with other machine learning algorithms implemented for botnet detection.



(a) Centralized IRC/HTTP Botnet (b) decentralized P2P Botnet

Figure 1. Difference between centralized and decentralized P2P botnets.

Structure of Paper

Section 1 of this paper discusses the background, problem description, and approach used in this study. Section 2 gives a brief overview of the literature. Section 3 gives a brief overview of the single stage along with its limitations and the motivation of choosing a multi-layer scheme. Moreover, the mechanism of the multi-layer scheme was also discussed. Section 4 discusses the results of the experiments. Finally, Section 5 concludes the paper.

2. Related Work

Machine Learning algorithms have been widely used to classify internet traffic. Irrespective of the class imbalance problem, Machine Learning classifiers such as Decision Trees and Neural Networks may produce a high accuracy for overall traffic flow but low accuracy for bytes. Accuracy for bytes means the number of bytes, carried by the packets of accurately classified traffic flow. Zhang et al. [9], proposed two algorithms based on feature selection and extended *wsu_auc* selection to apply the best features practically. They achieved more than 94% accuracy with an average byte accuracy of over 80%. Chen et al. [10] has proposed a high-speed network detection method for botnets. In this PF RING (to optimize spark streaming), the problem was solved at a high packet drop rate and for extracting required fields from traffic information. The authors used the random forest technique on the CTU dataset. They have acquired high precision, but the unconvincing aspect of this publication is that they used only offline data, and no other data was collected online. Azab et al. [11] proposed machine learning classifier (C45) and Correlation-based Feature Selection (CFS) for *Zeus V1.x*, *Zeus V2.x* and *benign HTTP* traffic detection in networks. Azab et al. demonstrated 99.3% accuracy with 55.6% recall results.

Wang et al. [12] proposed a solution to detect malicious Android apps. The authors utilized mobile traffic in which every HTTP traffic was treated as a document and this document was further used for word segmentation based on N-gram generation to generate candidate features which effectively characterized the particular flow of HTTP. The paper used the SVM machine learning algorithm for validation and demonstrated a high accuracy of 99.15% on static data while its detection rate was 54.81% on real environment testing. Albanese et al. [13] proposed a graph-based botnet detection technique called MTD (Moving Target Defense). The paper addressed the limitations of the static detection technique as MTD periodically altered the placement of detectors, but the authors did not mention the traffic speed which was affected by the periodical alteration in detective algorithms. Haddadi and Nur [14] have investigated five different botnet detection techniques. Two of them were rule-based systems and the other three were machine learning based techniques. The authors analysed their proposed approaches with different scenarios on twenty four publicly available datasets.

Alazab [15] examined the evolution of malware by the nature of its activity and variants. The paper investigated malware implication on the computer industry and provided a framework using feature extraction from malicious code which reflected the behaviour of the code. In other paper, Alazab [16] proposed a four-step methodology to develop a fully automated system for suspicious behavior detection. The four-step method classified the behavior of Application Program Interface API function calls based on the malicious intent hidden with the packed program. Zeng and Shen [6] proposed a two-step distributed approach for storm botnet detection which included a set of heuristics and first-step port numbers and an SVM classifier. The accuracy of their method was more than 95% with 8–12% of false positive rate. This scheme worked well with 0% false positive rate and 8% false negative rate to detect storm botnets. According to Zhang et al. [17], the P2P client was first identified by extracting the statistical pattern of P2P communication. The legal or legitimate P2P network and P2P botnet were further distinguished. Abdullah [18] and Yin [19] detected botnets based on host behavior. They detected zombie host and/or programs by monitoring changes in the host behavior, file, network connection, and registry contents. The method of detecting botnets based on host behavior cannot detect new and variant botnets. For example, an attacker could use new detection and hiding techniques such as rootkits and so on.

The detection based on traffic behavior has been mainly used in command and control phase [20], by using difference between C & C flow and the normal traffic in the flow characteristics such as communication rules, average packet size, periodic connections, and so on [21]. Therefore, Buczak and Guven [22] suggested to combine machine learning neural network for botnet real-time monitoring. The botnet detection method based on traffic behavior mainly analyzed two characteristics such as “Connection failure rate”, and, “Flow characteristics”. According to Zhang [17], the communication behavior between zombie hosts that joined the same botnet was similar. Therefore, P2P botnet traffic identification could adopt the scheme that firstly, analysed the traffic, and features were extracted. Secondly, the clustering algorithm was proposed to cluster the extracted data. The scheme was used to set the cluster threshold to improve the detection accuracy, without the use of existing botnet data stream for training. However, if there was only one zombie host in the current network, or if no traffic from different zombie hosts was found in the captured packets, this method was demonstrated inefficient by Zhang [17]. The three DT algorithms REPTree, Carriage, and C45 were analyzed in the study [23,24], where C45 with the lowest performance because its algorithm was easily under pruning, and the overfitting algorithm was more severe than REPTree. Encrypted P2P or unknown traffic could be classified by the statistics-based method, but it is not highly accurate, and could not identify untrained P2P traffic correctly [23].

Dhainotti et al. [25] introduced a technique for characterization of a large scale surreptitious scan for entire IPv4 space (a “/0” scan) conducted by the Sality botnet. In addition, the authors presented an analysis of botnet scanning behaviour along with general correlation and visualizations method across global internet. Guofei. et al. [26] presented a stochastic mechanism using an empirical test to isolate C & C botnet conversations from human to human conversations. They implemented a prototype system called BotProbe with desired accuracy. According to Guofei et al., about 53 % of the botnet command detected amongst thousands of real-world botnets (IRC-based), and about 14.4% were related to binary downloaded. In other study, Guofei et al. [27] characterized a botnet as an orchestrated class of malicious instances operated through the C & C communication channels. A botnet's essential properties were that the bots communicate with some Command & Control servers/peers, executing malevolent operations, and doing so in a correlated manner. Guofei et al., implemented a clustering paradigm for identical network traffic and identical malicious traffic; and further conducted cross-clustering correlations to categorize hosts sharing similar patterns of communication and similar patterns of malicious activity. The evaluations of Guofei et al. showed that their framework can effectively detect botnets with a very low false positive rate (0.003). Zhang et al. [28] developed a framework to effectively and efficiently detect suspicious hosts probably to be bots. They transferred network traffic to botnet detection systems for fine-grained check and efficient botnet detection based on DPI (Deep Packet Inspection). The paper proposed to reduce network traffic to improve the scalability of existing botnet detection systems. Guofei et al. [29] proposed a detection approach that identified Command & Control servers and the bots infected hosts in the networks. Their technique was predicated on the notion that bots inside the similar botnets were probably indicate spatial-temporal relationship and resemblance due to the pre-programmed events associated with C & C. The stream feature also included the number of upstream and downstream packets, the size of the uplink and downlink transmission bytes, the average length of the uplink and downlink data packets, the maximum length, the average variance, the duration of the data stream, and the packets that have been loaded in one stream. The stream feature selection method had a high detection rate because it did not depend on the botnet class to extract the common feature vector of the stream. Therefore, the detection strategy was widely concerned by experts and scholars in the field of network traffic analysis [15]. In high-speed, complex, and changing network environments, the main factors that determine the efficiency and accuracy of detection are the characteristics of the extraction and the classification strategy used.

3. Proposed Scheme

Three unique qualities of communication, i.e., P2P botnet, non-P2P and, Normal P2P activity, are demonstrated in Figure 2. These three characteristics overlap each other. If the three types of activities are grouped into a single stage mechanism, a large part of the P2P botnet traffic with normal qualities for non-P2P activity could be misclassified and few substantial P2P streams could be misclassified as countless streams due to the class imbalance problem. In this way, a single phase mechanism doesn't have the capability to order the traffic of three different qualities at the same time.

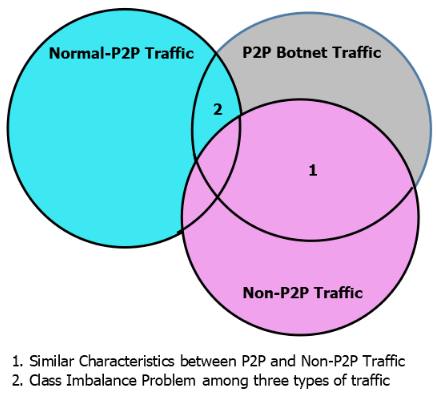

Normal-P2P Traffic
P2P Botnet Traffic
2
1
Non-P2P Traffic
1. Similar Characteristics between P2P and Non-P2P Traffic
2. Class Imbalance Problem among three types of traffic

Figure 2. Overlapping qualities among three types of traffic.

Therefore, a multi-layer method with a specific aim of settling the limitations of the single stage mechanism is very important. While there are many conceivable techniques exist, our research proposes a multi-layer mechanism, shown in Figure 3.

3.1. Multi-Layer Detection Method

This section describes the method proposed in this paper to detect bot-bling traffic in phases. The focus of this method is on non-P2P traffic filtering and the extraction of the characteristics of the session. The architecture of the model is shown in Figure 3. The first stage of the model will start from the traffic reduction followed by the three aspects of packet filtering rules, session characteristics, and classification algorithm. The second layer further classified the captured network traffic into two classes such as non-P2P and P2P. At the third layer of the model, we reduced the features which may marginally affect the classification. The final layer of the model classifies the traffic as either the traffic is normal P2P or botnet traffic. The detailed discussion on the entire mechanism of the proposed scheme is given in the rest of this section.

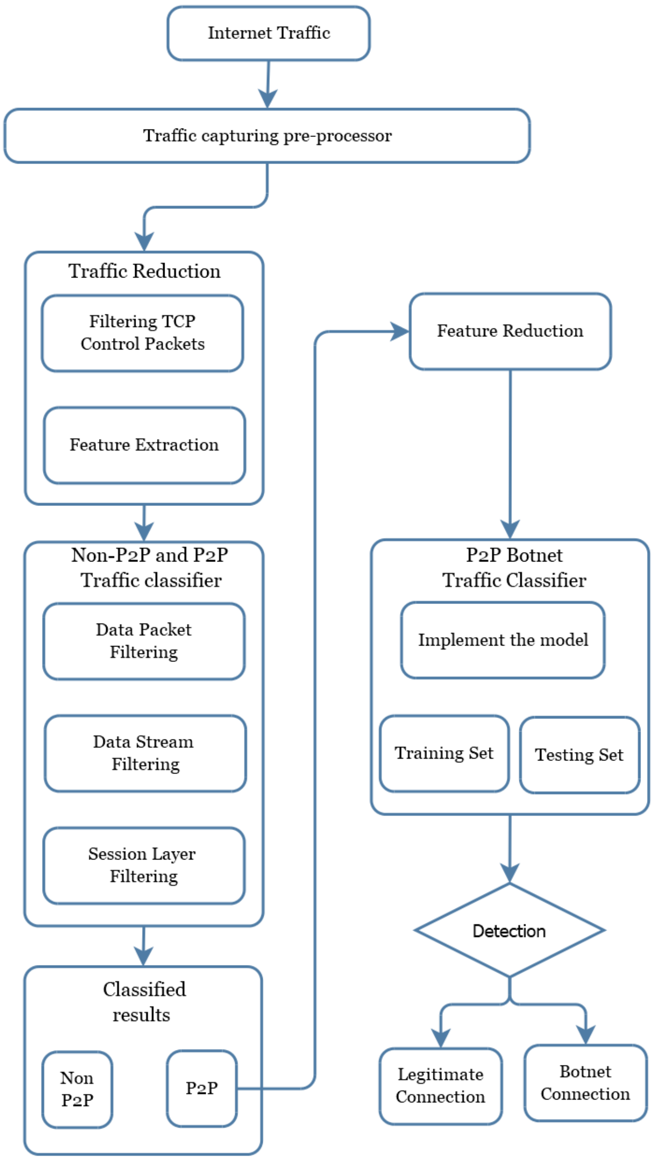


Figure 3. Architecture of the proposed method.

3.2. First Layer: Traffic Reduction

Reducing network traffic to detect malevolent behaviors is extremely important to manage a large amount of network traffic with limited resources, e.g., hard drive and memory. The hardest step in the process is to determine the network activity behavior by only scanning several packets for each flow. Our research therefore brings a new model to reduce congestion so that botnet detection systems can be deployed more easily on congested networks. In currently available identification methods for botnets, most of them use deep packet inspection (DPI) to evaluate packets content that is complicated and expensive and ineffective to simulate unidentified payload signatures [30]. It is assumed that the payload of each packet in DPI can be accessed by the system. This strategy can be especially accurate when the payload is in decrypted form. Most of the new malicious code generators, however, use the concealment strategies, such as encapsulation of protocols, obstruction and encrypting the payloads [31].

In addition, it is a costly task to evaluate all the packets on the congested networks because the amount of the packets transmitted through the networks are increasing day by day. Consequently, the DPI detection system can suffer from efficiency limited to the processing of the traffic from high-speed networks [30]. This study aims to improve the efficiency by reducing the amount of packets without hampering the precision rate. To accomplish an ultimate objective of this study, the selection of only Transmission Control Protocol TCP packets proposes a new reduction in traffic for a Botnet identification paradigm. This detection technique is used to reduce the volume of the network traffic and will increase the efficiency of the Botnet detection model.

In this study, TCP traffic control packets are filtered to reduce network traffic volumes and improve the efficiency of the envisaged strategy. Filtering involves mainly two steps: filtering the network flow associated with the TCP protocol and obtaining the control packets SYN, ACK, FIN and RST TCP. Figure 4 displays the network traffic reduction process (.PCAP files). An array of TCP Control Packets lists is initialized in the flowchart. New packets (TCP Control Packets List) are added to the array by tuples through the packets until the last packet in the file is reached. After the first loop i.e., for loop, the next loop examines the TCP packet header for loop, and the third loop selects packets without payload data. The flowchart's final loop gets the packet header.

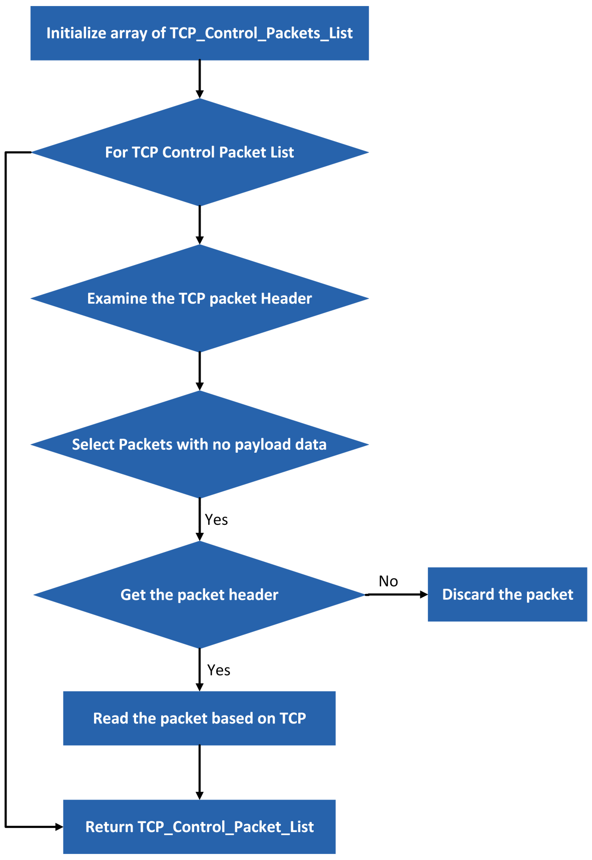

Initialize array of TCP_Control_Packets_List
For TCP Control Packet List
Examine the TCP packet Header
Select Packets with no payload data
Yes
Get the packet header
No
Discard the packet
Yes
Read the packet based on TCP
Return TCP_Control_Packet_List

Figure 4. Flow chart for network traffic reduction.

3.3. Second Layer: P2P and Non-P2P Traffic Classification

At present, port identification, signature recognition, and DNS query identification are commonly used methods for P2P traffic identification. These methods are based on stream feature [21]. However, the port identification method cannot recognize P2P applications with random ports or custom ports. DPI (Deep Packet Inspection Technology) does not recognize encrypted P2P traffic [32]. Stream-based identification methods can only determine P2P applications of the partial flow and has a high false alarm rate. Therefore, we use non-P2P well-known port filtering mechanism, DNS query, flow counting rules to filter non-P2P traffic, combined with fast heuristic P2P traffic identification method, as shown in Figure 5.

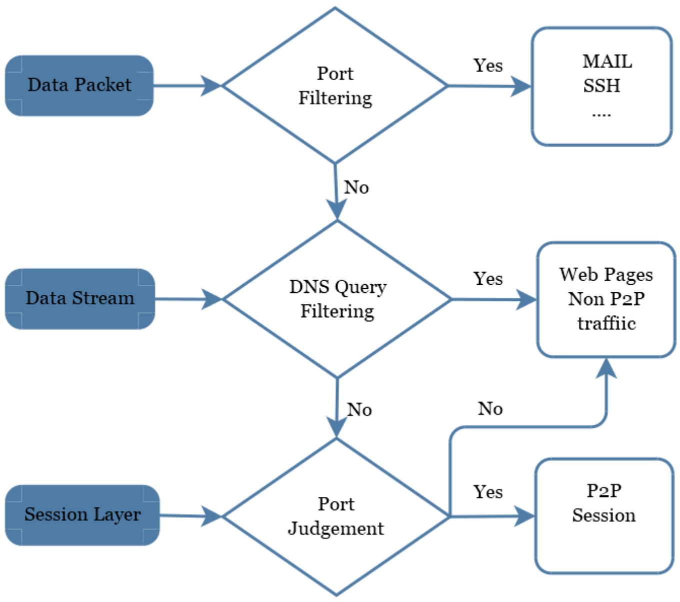


Figure 5. Second layer of traffic classifier.

The vulnerability of attack on internet application is usually initiated using open ports [33]. We choose port and time interval as the unit of analysis. Since some ports are more often used than other ports, therefore, it is important to identify the frequency of attack on those specific ports. Limitations of Port scan attack identification techniques are associated with the extent which they can quantify their complexity, accuracy, performance, or timely delivery. The previously studied techniques for detecting port scan attacks work either at the packet level or at the flow level or both [34]. In our proposed detection approach, we utilized packet level information because of its significance in determining the connection information along with capability of analysing payload of a packet. Another limitation of port scan detection is the challenge of labelling the large amount of network traffic data. This challenge is magnified further, if the security models produce profiles such as network traffic labelling for normal as well as malicious traffic, so updating the database of signatures (labels) dynamically will be a challenging task. However, more advanced approaches are still needed that can analyse both header and payload information to detect both labelled and unlabelled attacks.

To address the limitations of port scan detection methods, our framework is based on a multi-layer approach to detect multiple types of attacks such as IRC, SPAM, Click Fraud, DDoS, FastFlux, Port Scan, Compiled and Controlled record by CTU, HTTP, Waledac, Storm and Zeus botnets. More precisely, instead of limitations of port scan detection technique, our framework used multi-layers to effectively detect botnets. Port filtering is one layer used in our proposed framework and if we ignore port filtering or remove this layer from our framework so our system can be easily attacked by port scan attacks. Secondly, our experiments demonstrate in Table 3 that our framework captured port scan attacks 100%, which means that not a single packet of port scan attack was missed by our port filtering layer. Therefore, the error associated with port filtering method is very small due to a multi-layer approach. Algorithm 1 demonstrates the mechanism of second layer in our proposed model. We choose port and time interval as the unit of analysis. Since some ports are used more often than other ports; therefore, it is important to identify the frequency of attack on these specific ports. For instance, when we see something going through on port 25, we expect it to be a mail server, and likewise if we find something going through on port 80, we simply find that it is a webserver. The Internet Assigned Numbers Authority (IANA) [35] officially assign port numbers for specific uses and applications.

Algorithm 1 Second layer classification

```
INPUT: Data Packet/ Data Stream/ Session Layer
OUTPUT: MAIL/ SSH/ Non-P2P Traffic/ P2P Session
Step1: Classify Data Packets by $PortFiltering$:
           if data packet is true then go to MAIL/SSH
           else go to step 2.
Step2: Classify Data Stream by $DNSQueryFiltering$:
           if true then go to Non-P2P Traffic
           else go to step 3.
Step3: Classify Session Layer$PortJudgement$:
           if true then go to P2P Session else go to Non-P2P traffic Traffic
Step4: Return
```

Port filtering is a packet-level filtering method, mainly filtering the commonly used non-P2P application traffic. DNS query is a stream-level filtering method, flow counting, and port judgment is a session-level filtering method, the two rules are mainly filtered web pages and other non-P2P traffic. Among them, the port-based filtering method can identify some common non-P2P application traffic, such as Secure Shell SSH Defined generally using port 22 and Telnet (remote login) using port 23. Some commonly used applications and their corresponding port numbers are shown in Table 1. TCP means Transmission Control Protocol, UDP means User Datagram Protocol.

Table 1. Common applications and their corresponding ports.

Application	Port Number	Transport Protocol	Description
SSH	22	TCP, UDP, SCTP	Secure Shell Protocol
TelNet	23	TCP, UDP	TelNet
MAIL	25, 110, 143, 220, 465, 993, 995	SMTP, POP3, IMAP4, IMAP, SMTP over TLS, IMAP4 over SSL, POP3 over SSL	Simple Mail Transfer, Post Office Protocol—Version 3, Internet Message Access Protocol, Interactive Mail Access Protocol v3, Message Submission over TLS protocol, IMAP4/POP3 over TLS/SSL (993, 995)
NetBios	125, 137, 139, 445	TCP, UDP	Locus PC-Interface Net Map Server, NETBIOS Name Service, Microsoft-DS
Remote	3389	TCP, UDP	MS WBT Server
FTP-Data	20, 21	TCP, UDP, SCTP	File Transfer
NTP	123	TCP, UDP	Network Time Protocol

In general, P2P node communication does not require domain name resolution but directly read the IP list stored in the local configuration file to obtain IP [36]. However, for non-P2P applications, DNS domain name resolution must be used to obtain IP. Therefore, one of the criteria for determining non-P2P network data flows such as Web and Mail, etc., is resolved by the domain name and may be the destination IP address in the network flow.

When a user sends a Web application service request normally, the Web application uses a multi-port, parallel-requested connection to an IP address on a page. As a result, multiple data streams appear in the same session. The P2P network node communicates each time using a pair of random source and destination ports. Therefore, we can use flow counting and port determination to filter non-P2P traffic. If a session is using the TCP protocol and 808,080 or 443 port, and the number of sessions in the flow exceeds the threshold, then the session can be considered a web page traffic session. Where the number of valid streams in a session is represented, the threshold is selected based on the number of streams that appear in the normal page access session. Using the capture tool to collect simple and relatively complex web page requests, the analysis results show that the simple web page is generally three to four connection requests, and the complexity of the page connection request is five to eight. Therefore, the threshold is set to three in this article.

3.4. Third Layer: Feature Extraction and Feature Reduction

Reducing the number of characteristics from the features list is the method to remove those functionalities which may marginally effect the classification [37]. Feature reduction helps in reducing the overfitting and data set imbalance problems. The reliability of reducing the features is therefore one of the most significant aspects affecting the precision rate of the detection model.

In the phase of feature extraction, the features are synthesized which are valuable in locating the malevolent behavior of the Bot and these characteristics are obtained in 29 iterator qualities based on 60-s connections. The interpretation of a link is obtained as the subgroup of packets received between two different networks (source port, source IP address, destination port, and destination IP address). All features are obtained explicitly from the control packet header via a thorough check of the payload material of the packets instead of previous approaches. The efficiency is therefore improved and utilization of the computational resources is reduced e.g., memory allocation and processing. The feature list was collected from the 60-s connection and consists of 60-s connection function vector.

The purpose of this study is to select an appropriate subset of features that will enhance the efficiency of the machine learning classifiers and reduce model complexity without drastically reducing the precision. We used a classification technique for feature reduction to eliminate less important features in order to reduce the amount of data, in order to achieve better levels of accuracy. The decision tree consists of two node types: two children's internal nodes and children's leaf nodes. Any internal node has a decision mechanism to demonstrate the next node to visit. Training samples containing a number of features starts the construction of the tree. During construction of the tree, the training set is repetitively divided into small subsets. A predicted class is assigned to each resulting node based on the decision matrix of the class distribution in the training set. The internal node test is determined on the basis of an impurity measurement to pick the selected function and threshold values. The most common impurity metric is entropy impurity [38], which we used for feature reduction, is shown in Figure 6.

3.5. Fourth Layer: P2P Traffic Classification

According to data flow characteristics analysis of P2P botnet, the traffic characteristics of benign and botnets are similar. Therefore, this paper uses the session-based strategy for feature extraction. Thus, improving the detection efficiency by reducing the number of stream features and the number of data for same destination address in same session. P2P zombie host and other suspicious host communication processes automatically initiate a connection using a malicious program or code. The flow of the duration is generally short and very fixed. Therefore, we can extract the session features such as the average, maximum, minimum, and standard deviation of the duration of the session, and the average interval of the upstream (downstream) stream packets in the session. Figure 7 demonstrates the flow diagram and Algorithm 2 describes the mechanism of P2P traffic classification which is fourth layer in our proposed model.

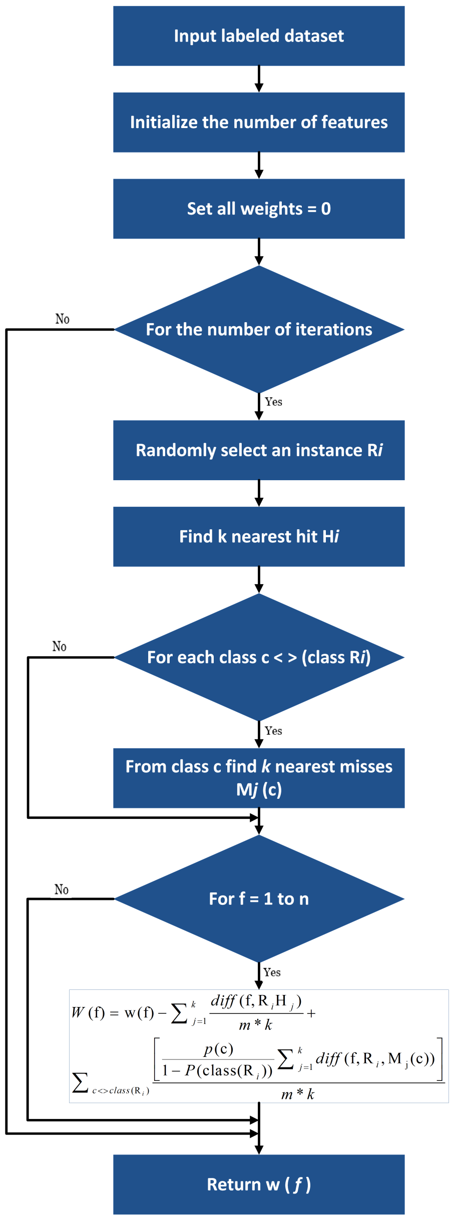

Input labeled dataset
Initialize the number of features
Set all weights = 0
For the number of iterations
No
Yes
Randomly select an instance R*i*
Find k nearest hit H*i*
For each class c < > (class R*i*)
No
Yes
From class c find *k* nearest misses M*j* (c)
For f = 1 to n
No
Yes
$W(f) = w(f) - \sum_{j=1}^{k} \frac{diff(f, R_i H_j)}{m * k} + \sum_{c <> class(R_i)} \frac{\left[\frac{p(c)}{1 - P(class(R_i))} \sum_{j=1}^{k} diff(f, R_i, M_j(c)) \right]}{m * k}$
Return w (*f*)

Figure 6. Flow chart of Feature Reduction Layer.

Algorithm 2 Fourth Layer: P2P Traffic Classification

```
INPUT:    Pre-labled Dataset from the second layer (Non-P2P/P2P Session)
OUTPUT:   Legitimate Connection/Botnet Connection
-         Traning Phase:
Step1:    Inset Pre-labled Dataset i.e., Non-P2P Traffic/P2P Traffic
Step2:    Extract Flows with fine-grained features
Step3:    Generate Classifier Model
          Go to Step 6.
-         Classifying Phase:
Step4:    Classified P2P Traffic
Step5:    Get flows with fine-grained features gain the features of the p2p traffic.
Step6:    P2P Traffic Classifier classify the P2P traffic in two classes as follows:
          Legitimate connection, Botnet Connection
Step7:    Return
```

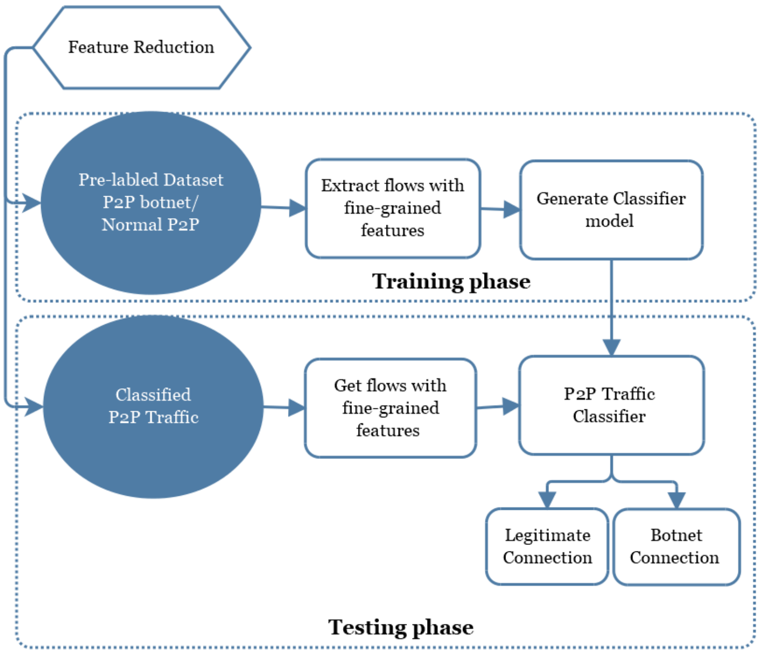

Feature Reduction
Pre-labled Dataset P2P botnet/ Normal P2P
Extract flows with fine-grained features
Generate Classifier model
Training phase
Classified P2P Traffic
Get flows with fine-grained features
P2P Traffic Classifier
Legitimate Connection
Botnet Connection
Testing phase

Figure 7. Fourth layer traffic classifier.

In the process of communication between two nodes in the P2P botnet, the size and transmission quantity of the transmitted packets are relatively small, and the C & C communication flow generated by the zombie host in the same network has great similarity. Therefore, we can distinguish between normal P2P network traffic and P2P botnet traffic by distributing the traffic into small sessions.In summary, all the features which were extracted from the session are shown in Table 2.

Table 2. Session characteristics.

Eigenvalues	**Description**
avg_duration	The mean of the total duration of the different network flows in the same session.
std_duration	The standard deviation of the total duration of the different network flows in the same session.
min_duration	The minimum total duration of the different network flows in the same session.
max_duration	The maximum total duration of the different network flows in the same session.
avg_f(b)int	Average interval of uplink (downstream) packet transmission for different network flows in the same session.
max_f(b)pl	Average value of the maximum value of the uplink transmission packet length for different network flows in the same session.
min_f(b)pl	Average value of the minimum value of the uplink transmission packet length for different network flows in the same session.
std_avg_f(b)pl	The standard deviation of the average of the length of the packet transmitted in the uplink (downstream) of different network flows in the same session.
avg_f(b)pen	Average number of valid packets transmitted upstream (downstream) of different network flows in the same session.
std_avg_f(b)pen	The standard deviation of the number of valid packets transmitted upstream (downstream) of different network flows in the same session.
avg_f(b)pb	The average of the total number of bytes transmitted upstream (downstream) of different network flows in the same session.
std_f(b)pb	The standard deviation of the total number of bytes transmitted upstream (downstream) of different network flows in the same session.

4. Results and Analysis

4.1. Evaluating Metrics

We assessed the execution of our methodologies utilizing 10-fold cross-validation. The methodology of k-fold cross-validation. The first sample was arbitrarily apportioned into ten equivalent measured sub-tests. Nine sub-tests were utilized for training the model and the remaining one sub-test was held for testing the model. The procedure was rehashed ten times, utilizing an alternate sub-sample for testing, every time. The outcomes then found the average values for the single and final results. Furthermore, we used the wrapper method for feature selection. The mechanism of the wrapper technique is shown in Figure 8.

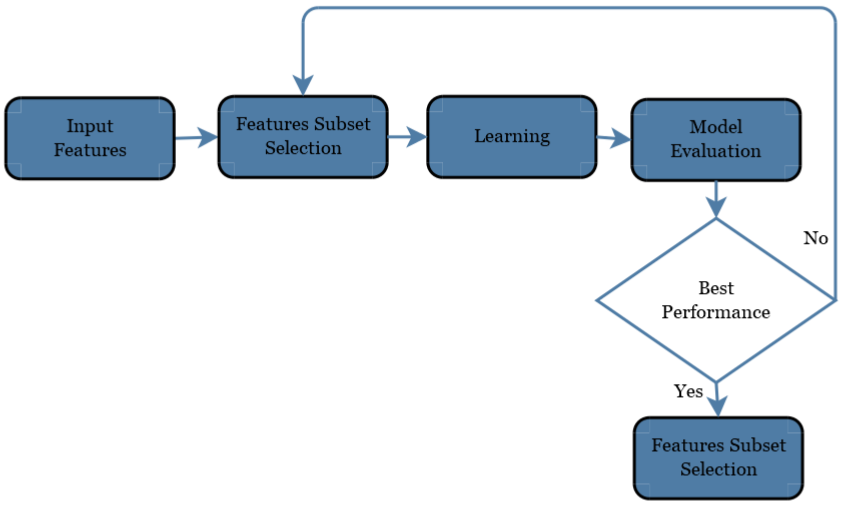


Figure 8. Wraper method for features selection.

4.1.1. Accuracy of Detection Model

The percentage of correctly classified instances among the total number of instances:

$$ACC = \frac{TN + TP}{TN + FP + FN + FP}.$$

4.1.2. False Alarm Rate (FAR)

The FAR is referred to sensitivity or the False Positive Rate (FPR) [39]. We have used the following formula for False Alarm Rate:

$$FAR = \frac{FP}{TN + FP}.$$

4.2. Dataset and Experimental Setup

We have used two diverse datasets i.e., CTU and ISOT. These two datasets described in the following discussions and are substantially different for our cross-dataset learning mechanism i.e., directory change is the most pertinently in the cross-domain, but the same behaviour dataset scenarios contained network traffic of similar kinds of malicious attacks. In our experiments, we demonstrated that these supervised domain adaptation methods worked efficiently on our cross-dataset transfer learning task because the target dataset contained similar behaviour. Diverse datasets with similar behaviour can be used for cross-dataset-validation [40]. We set a *VMWare* virtual environment in Windows 10 and Linux operating systems for experiments. *Nfdump* was set up on the system to collect network data. *Nfdump* captured network flow and stored into *nfcapd* files.

One instance of the *nfcapd* file is associated with a flow data record over time. Description of the datasets used in our experiments are as given below.

CTU-13 Dataset [41]: The dataset from CTU-13 project contains thirteen various captured scenarios of different botnet samples. Table 3 describes different types of attacks included in CTU-13 dataset, such as IRC (to block normal conversation and increase server load), Spam traffic (emails containing malicious links), Click Fraud for example (pay per click), Port Scan (an application designed to probe a server or host to exploit vulnerabilities), DDoS (Distributed Denial of Service attacks), Fast Flux (fake domain name system that host malicious contents), included in CTU dataset [42]. The captured files were stored in the *pcap* form. The dataset of CTU-13 project is a labelled dataset with background botnet traffic.

ISOT [43]: The ISOT dataset includes both malicious (Waledac, Storm and Zeus botnets) and non-malicious traffic (HTTP traffic, gaming packets and P2P applications e.g., BitTorrent). Table 4 describes different types of attacks quantitatively and qualitatively included in ISOT dataset, for example, storm network traffic is a worm traffic, Waledac is fake SMTP and UDP traffic, Zues is Trojan horse traffic [44].

Figures 9 and 10 demonstrate the working environment of *Wireshark* setup to capture the network traffic to obtain the CSV files for further analysis and experimentation. For non-malicious traffic (benign traffic), while running the *Wireshark* on real-time network traffic, we used to access the well-known and the most trusted network traffic for example UESTC intranet, thus the data packets were captured and stored. Figure 11 shows the simulation of non-malicious traffic along with date, time, duration and packets per second. *Wireshark* has the capability to decrypt Secure Sockets Layer (SSL)/ Transport Layer Security (TLS) network traffic for packet analysis [45]. The mechanism of decrypting [46] presented an analysis on how to decrypt of SSL/TLS network traffic. Different decryption procedures and techniques have been used by leading companies for encrypted malicious activities detection. In addition, the SSL/TLS-transmitted traffic interception and decryption approaches were established and validated by Martin et al. [46]. The proposed strategy was utilized for distant connection eavesdropping, enabling transmitted data to be decrypted in a near real-time mode. There have been two fundamental inspection mechanisms for SSL/TLS traffic, depending on what kind of pertinent information and certificates are obtainable and how the devices are installed on the network [47,48].

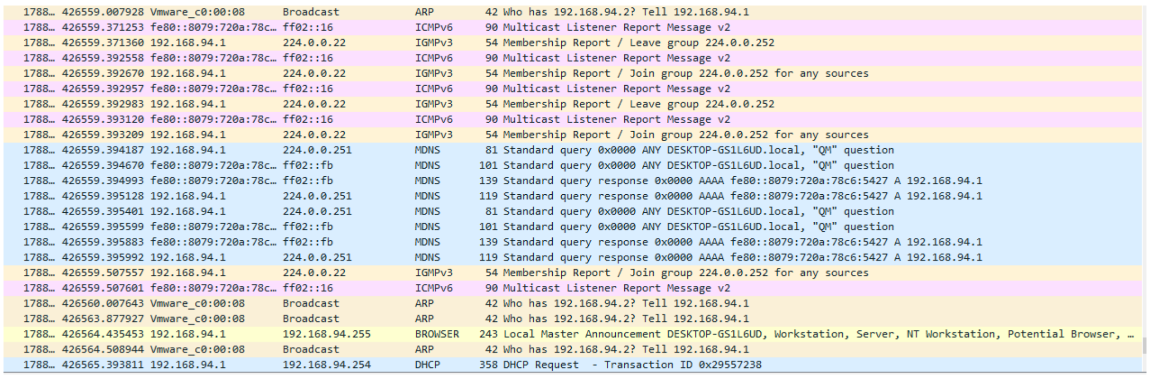
Figure 11 is the sample simulation of benign traffic while Figures 12–15 are the sample simulations of *Neris* botnets, *Rbot* botnets *Donbot* botnets and *Sogou* botnets, respectively. The *X-axis* of the graphs show the time intervals while the *Y-axis* shows the packet lengths per second. Blue lining in the graphs show the capture of all benign traffic while the black color shows the “Bad TCP capture” or error filter catch. “Bad TCP” capture indicates that there is something wrong with the TCP communication in the capture file so it was observed in each and every graph that there were particular botnets in the traffic while Figure 11 indicates benign traffic in the network.

Table 3. Quantitative description of CTU-13 dataset.

Scenario No.	Scenario Name	Type of Attack	Packets Captured	Size (GB)	Duration (h)	Number of Infected Nodes	Displayed
1	Neris	IRC, SPAM, CF	323,154	52	6.15	1	100%
2	Neris	IRC, SPAM, CF	176,064	60	4.21	1	100%
3	Rbot	IRC, PS, US	495,056	121	66.85	1	100%
4	Rbot	IRC, DDoS, US	256,712	53	4.21	1	100%
5	Virut	SPAM, PS, HTTP	45,853	37.6	11.63	1	100%
6	Menti	PS	24,764	30	2.18	1	100%
7	Sogou	HTTP	20,663	5.8	0.38	1	100%
8	Murlo	PS	85,735	123	19.5	1	100%
9	Neris	IRC, SPAM, CF, PS	2,129,949	94	5.18	10	100%
10	Rbot	IRC, DDoS, US	66,340,518	73	4.75	10	100%
11	Rbot	IRC, DDoS, US	3,941,769	5.2	0.26	3	100%
12	NSIS.ay	P2P botnet	352,266	8.3	1.21	3	100%
13	Virut	SPAM, PS, HTTP	440,625	34	16.36	1	100%

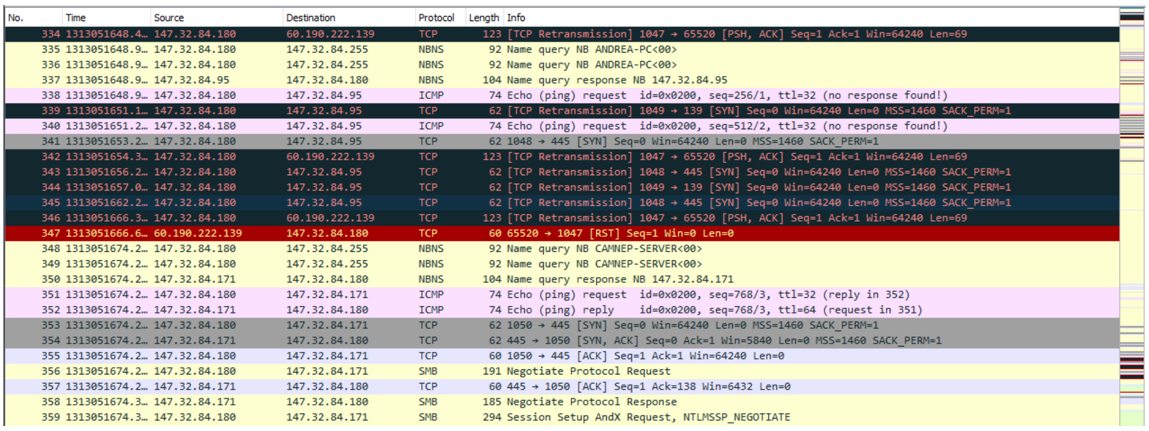
Table 4. Quantitative description of ISOT dataset.

Dataset Name	Type of Attack	Description	Packets Captured	Size (GB)	Duration (h)	Displayed
ISOT [43]	Storm Waledac Malicious SMTP and Malicious UDP	Worm Traffic Fake SMTP and UDP (SPAM) Trojan Horse	371,899	10.6	4.1	100%
Benign Traffic	TCP, UDP	Clean Traffic	148,222	3.5	31.2	100%



```
1788… 426559.007928 Vmware_c0:00:08       Broadcast        ARP       42 Who has 192.168.94.2? Tell 192.168.94.1
1788… 426559.371253 fe80::8079:720a:78c… ff02::16         ICMPv6    90 Multicast Listener Report Message v2
1788… 426559.371360 192.168.94.1          224.0.0.22       IGMPv3    54 Membership Report / Leave group 224.0.0.252
1788… 426559.392558 fe80::8079:720a:78c… ff02::16         ICMPv6    90 Multicast Listener Report Message v2
1788… 426559.392670 192.168.94.1          224.0.0.22       IGMPv3    54 Membership Report / Join group 224.0.0.252 for any sources
1788… 426559.392957 fe80::8079:720a:78c… ff02::16         ICMPv6    90 Multicast Listener Report Message v2
1788… 426559.392983 192.168.94.1          224.0.0.22       IGMPv3    54 Membership Report / Leave group 224.0.0.252
1788… 426559.393120 fe80::8079:720a:78c… ff02::16         ICMPv6    90 Multicast Listener Report Message v2
1788… 426559.393209 192.168.94.1          224.0.0.22       IGMPv3    54 Membership Report / Join group 224.0.0.252 for any sources
1788… 426559.394187 192.168.94.1          224.0.0.251      MDNS      81 Standard query 0x0000 ANY DESKTOP-GS1L6UD.local, "QM" question
1788… 426559.394670 fe80::8079:720a:78c… ff02::fb         MDNS     101 Standard query 0x0000 ANY DESKTOP-GS1L6UD.local, "QM" question
1788… 426559.394993 fe80::8079:720a:78c… ff02::fb         MDNS     139 Standard query response 0x0000 AAAA fe80::8079:720a:78c6:5427 A 192.168.94.1
1788… 426559.395128 192.168.94.1          224.0.0.251      MDNS     119 Standard query response 0x0000 AAAA fe80::8079:720a:78c6:5427 A 192.168.94.1
1788… 426559.395401 192.168.94.1          224.0.0.251      MDNS      81 Standard query 0x0000 ANY DESKTOP-GS1L6UD.local, "QM" question
1788… 426559.395599 fe80::8079:720a:78c… ff02::fb         MDNS     101 Standard query 0x0000 ANY DESKTOP-GS1L6UD.local, "QM" question
1788… 426559.395883 fe80::8079:720a:78c… ff02::fb         MDNS     139 Standard query response 0x0000 AAAA fe80::8079:720a:78c6:5427 A 192.168.94.1
1788… 426559.395992 192.168.94.1          224.0.0.251      MDNS     119 Standard query response 0x0000 AAAA fe80::8079:720a:78c6:5427 A 192.168.94.1
1788… 426559.507557 192.168.94.1          224.0.0.22       IGMPv3    54 Membership Report / Join group 224.0.0.252 for any sources
1788… 426559.507601 fe80::8079:720a:78c… ff02::16         ICMPv6    90 Multicast Listener Report Message v2
1788… 426560.007643 Vmware_c0:00:08       Broadcast        ARP       42 Who has 192.168.94.2? Tell 192.168.94.1
1788… 426563.877927 Vmware_c0:00:08       Broadcast        ARP       42 Who has 192.168.94.2? Tell 192.168.94.1
1788… 426564.435453 192.168.94.1          192.168.94.255   BROWSER  243 Local Master Announcement DESKTOP-GS1L6UD, Workstation, Server, NT Workstation, Potential Browser, …
1788… 426564.508944 Vmware_c0:00:08       Broadcast        ARP       42 Who has 192.168.94.2? Tell 192.168.94.1
1788… 426565.393811 192.168.94.1          192.168.94.254   DHCP     358 DHCP Request  - Transaction ID 0x29557238
```

Figure 9. Network traffic captured with Wireshark.



```
No.   Time             Source           Destination      Protocol Length Info
  334 1313051648.4… 147.32.84.180    60.190.222.139   TCP      123 [TCP Retransmission] 1047 → 65520 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=69
  335 1313051648.9… 147.32.84.180    147.32.84.255    NBNS      92 Name query NB ANDREA-PC<00>
  336 1313051648.9… 147.32.84.180    147.32.84.255    NBNS      92 Name query NB ANDREA-PC<00>
  337 1313051648.9… 147.32.84.95     147.32.84.180    NBNS     104 Name query response NB 147.32.84.95
  338 1313051648.9… 147.32.84.180    147.32.84.95     ICMP      74 Echo (ping) request  id=0x0200, seq=256/1, ttl=32 (no response found!)
  339 1313051651.1… 147.32.84.180    147.32.84.95     TCP       62 [TCP Retransmission] 1049 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
  340 1313051651.2… 147.32.84.180    147.32.84.95     ICMP      74 Echo (ping) request  id=0x0200, seq=512/2, ttl=32 (no response found!)
  341 1313051653.2… 147.32.84.180    147.32.84.95     TCP       62 1048 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
  342 1313051654.3… 147.32.84.180    60.190.222.139   TCP      123 [TCP Retransmission] 1047 → 65520 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=69
  343 1313051656.2… 147.32.84.180    147.32.84.95     TCP       62 [TCP Retransmission] 1048 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
  344 1313051657.0… 147.32.84.180    147.32.84.95     TCP       62 [TCP Retransmission] 1049 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
  345 1313051662.2… 147.32.84.180    147.32.84.95     TCP       62 [TCP Retransmission] 1048 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
  346 1313051666.3… 147.32.84.180    60.190.222.139   TCP      123 [TCP Retransmission] 1047 → 65520 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=69
  347 1313051666.6… 60.190.222.139   147.32.84.180    TCP       60 65520 → 1047 [RST] Seq=1 Win=0 Len=0
  348 1313051674.2… 147.32.84.180    147.32.84.255    NBNS      92 Name query NB CAMNEP-SERVER<00>
  349 1313051674.2… 147.32.84.180    147.32.84.255    NBNS      92 Name query NB CAMNEP-SERVER<00>
  350 1313051674.2… 147.32.84.171    147.32.84.180    NBNS     104 Name query response NB 147.32.84.171
  351 1313051674.2… 147.32.84.180    147.32.84.171    ICMP      74 Echo (ping) request  id=0x0200, seq=768/3, ttl=32 (reply in 352)
  352 1313051674.2… 147.32.84.171    147.32.84.180    ICMP      74 Echo (ping) reply    id=0x0200, seq=768/3, ttl=64 (request in 351)
  353 1313051674.2… 147.32.84.180    147.32.84.171    TCP       62 1050 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
  354 1313051674.2… 147.32.84.171    147.32.84.180    TCP       62 445 → 1050 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1
  355 1313051674.2… 147.32.84.180    147.32.84.171    TCP       60 1050 → 445 [ACK] Seq=1 Ack=1 Win=64240 Len=0
  356 1313051674.2… 147.32.84.180    147.32.84.171    SMB      191 Negotiate Protocol Request
  357 1313051674.2… 147.32.84.171    147.32.84.180    TCP       60 445 → 1050 [ACK] Seq=1 Ack=138 Win=6432 Len=0
  358 1313051674.3… 147.32.84.171    147.32.84.180    SMB      185 Negotiate Protocol Response
  359 1313051674.3… 147.32.84.180    147.32.84.171    SMB      294 Session Setup AndX Request, NTLMSSP_NEGOTIATE
```

Figure 10. Network traffic captured with Wireshark.

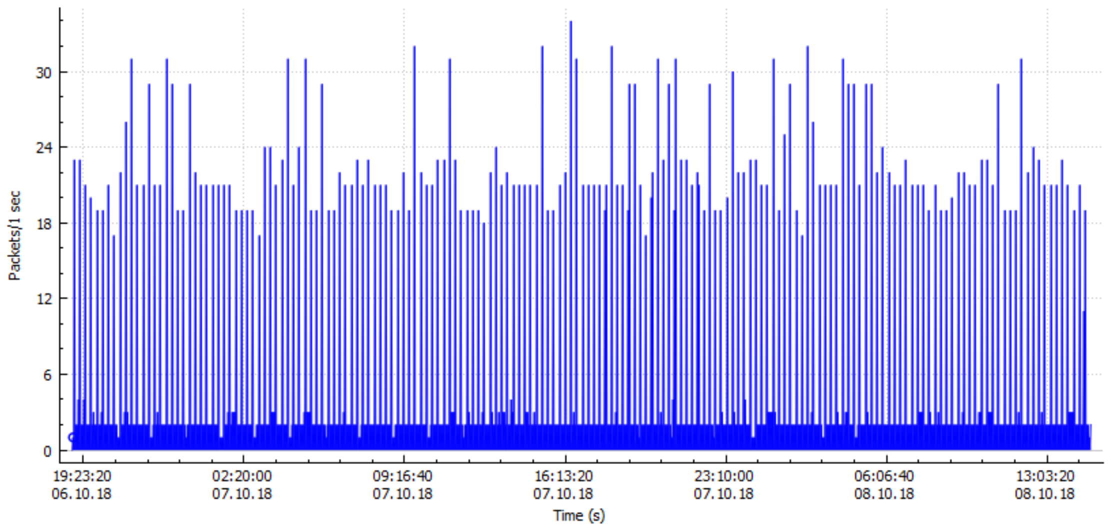


Figure 11. Benign network traffic captured.

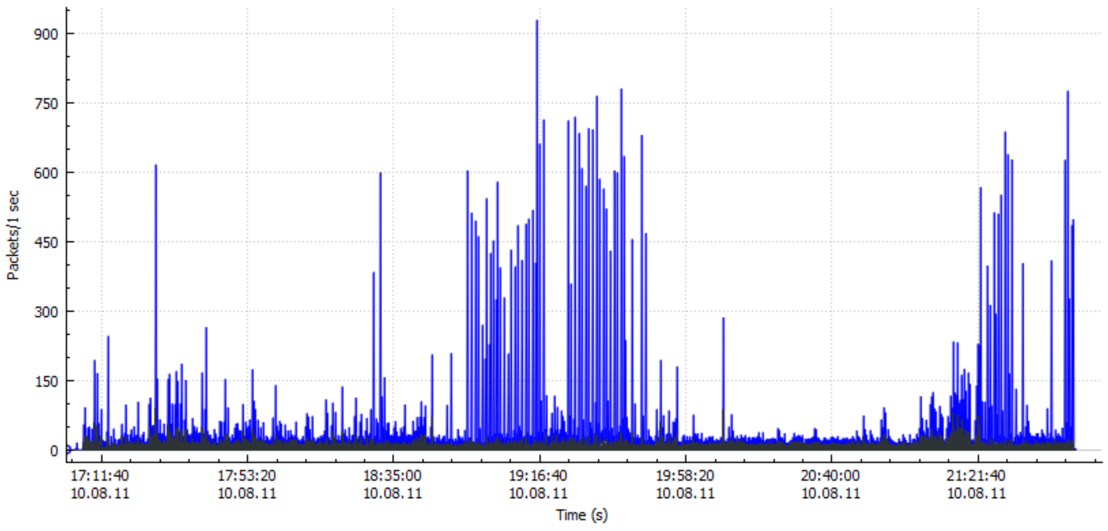


Figure 12. Neris Botnet captured.

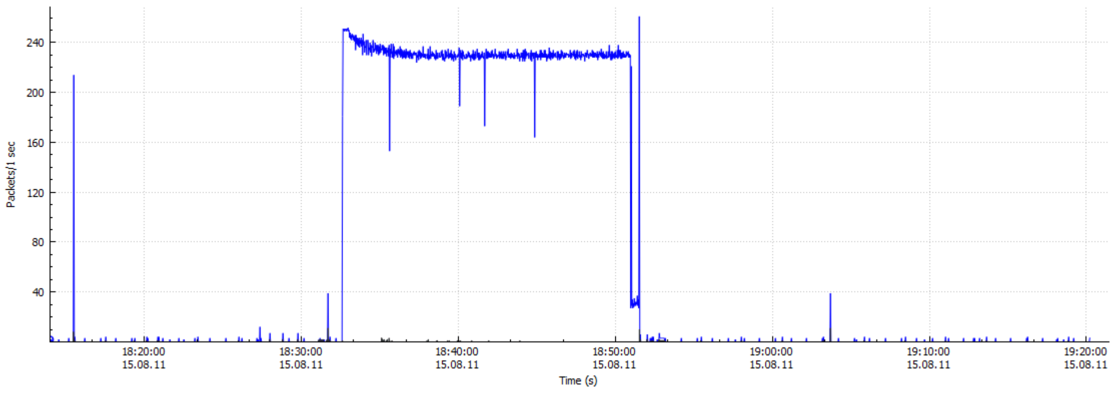


Figure 13. RBot Botnet captured.

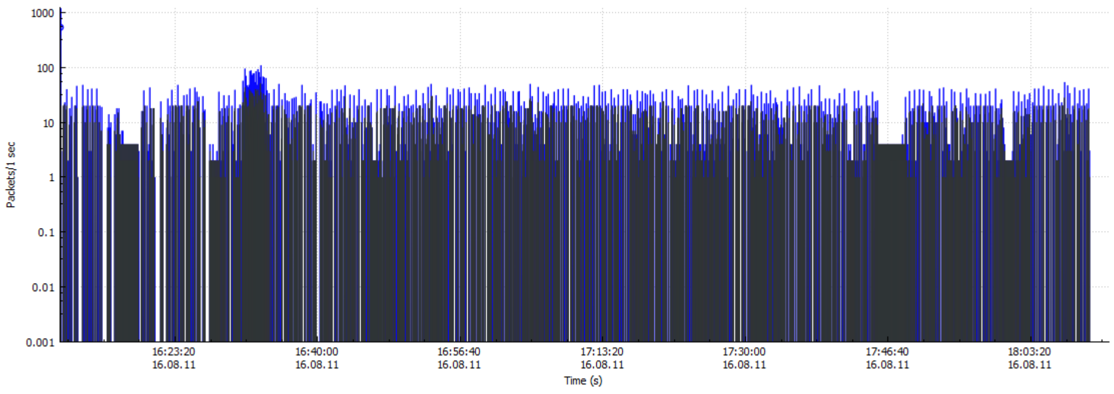


Figure 14. Donbot Botnet captured.

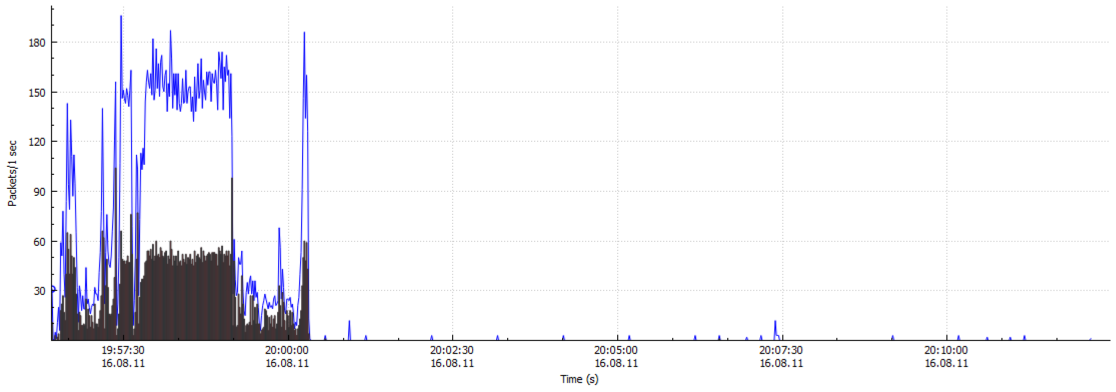


Figure 15. Sogou Botnet captured.

4.3. Detection Results and Discussion

This paper presents a multi-layer technique for classification (P2P botnet traffic and non-P2P traffic) and identification of botnets by applying the machine learning classifier. The network traffic is filtered out by exploiting network features such as port filtering, DNS query, and flow counting. Furthermore, the 10-fold cross-validation mechanism was deployed to compare and analyze the performance of the filter-less session and the filtered session feature after classification. A threshold value was used to analyze and compare the results. In fact, the threshold is a minimum or maximum value (developed for a characteristic, feature or variable) that serves as a benchmark for comparison or instruction and which may require a complete system review. The recognition rate of internet traffic is related to the set of the thresholds, flow counting, and port identification. The detection rates of the botnet are 99%, 98.9%, 98.7% and so on. The false alarm rate of normal traffic is 9%, 7.6%, 3% for thresholds 1, 2 and 3, respectively. The false alarm rate of normal traffic for thresholds 4, 5, 6, 7 and 8 is 1%. In this experiment, we use different thresholds (1, 2, 3, 4, 5, 6, 7, 8) to identify internet traffic. The results are shown in Figure 16, i.e., when the threshold reaches 3, the detection rate drops rapidly. With the increase of the threshold value more than 3, and the false alarm rate does not change. However, when the threshold is too small which is 1 in our case, so the false alarm rate will increase, and the detection rate does not change much. Therefore, according to the test results, we set the threshold 3 in the entire P2P traffic detection process. The results show that the classification of a well-known port, DNS, and flow counting filtering is not only better than P2P without filtering, but also the false alarm rate of normal traffic is further reduced. Furthermore, our experiments also demonstrate that the accuracy of the proposed framework was improved up to 99%, however, at the expense of false reporting of benign files as botnets as well as false reporting of botnet as benign. Consider the factor of whether benign files also send out search requests consistently so benign files may be reported as botnets. Additionally, it was observed that the accuracy might be improved by increasing the epochs of deep learning algorithms at the expense of more execution costs.

In general, the classification accuracy of the Decision Tree algorithm is positively correlated with the number of classification trees and the classification tree depth. However, the detection rate of the Decision Tree algorithm is negatively correlated with the number of classification trees. In order to determine the model used to identify P2P botnet traffic in a high-speed network environment, this paper makes a comparative analysis of the influence of a different number of the classification tree and depth of the classification tree on the basis of the detection rate.

Based on the session characteristics, this paper uses a Decision Tree classification algorithm to classify Normal P2P and botnet traffic. The depth of the Decision Tree algorithm was set to 8, and the classification tree was set to 100, as shown in Figure 16. The Decision Tree detection rate is 98.7% for the threshold 3 of the false positive rate, while other machine learning P2P botnet traffic detection results are shown in Table 5 and Figure 17.

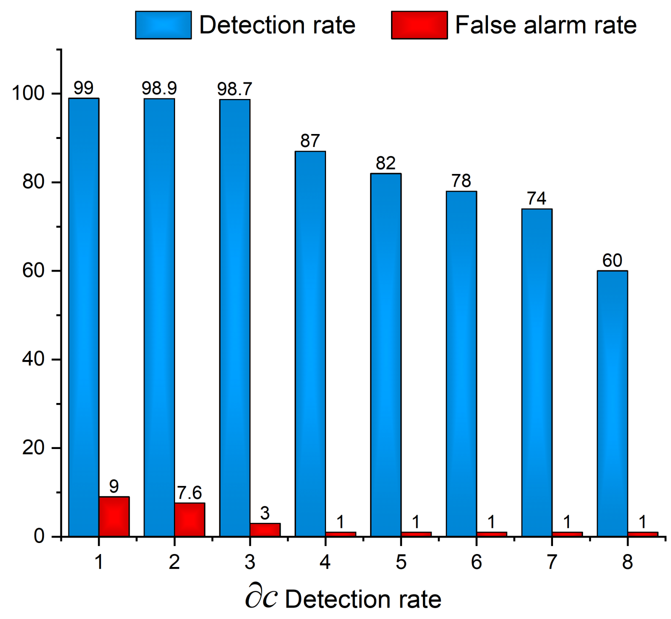


Figure 16. Network traffic identification under different thresholds during the training process.

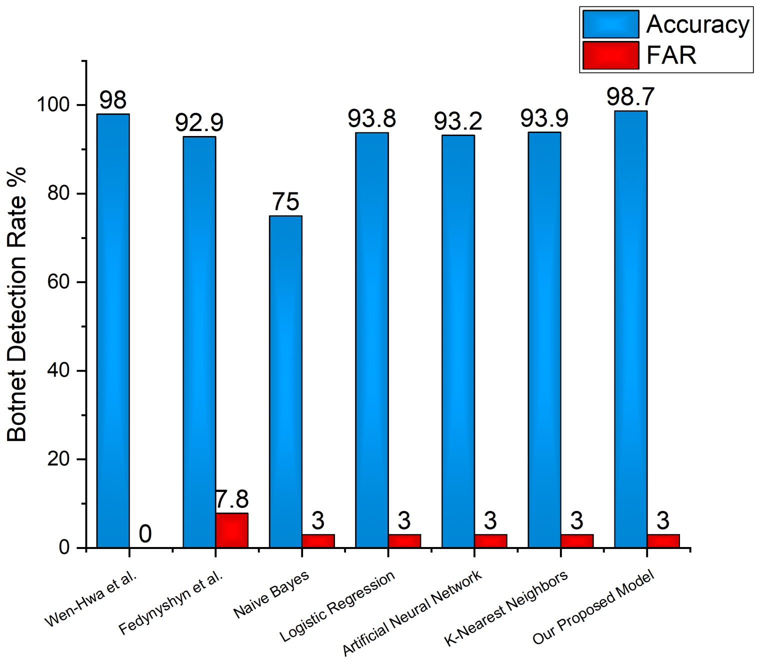


Figure 17. Comparison with other machine learning classifiers on P2P botnet detection.

4.4. Comparison with Other Machine Learning Classifiers

So far, many supervised machine learning algorithms are used to classify the data. Khan et al. [49,50] analysed ResNet and GoogleNet models for malware detection using image processing technique. Kumar et al. [51,52] used the Convolutional Neural Network CNN model for malicious code detection based on pattern recognition and permission-induced risk for Android IoT Devices, respectively. Comparison of the different approaches for botnet identification is a difficult task because different evaluations and experiments use different botnet samples and data sets. We have compared our proposed model with other detection models based on precision and false positive rates. We have compared five machine learning models and two other models previously published. In this study, Table 5 demonstrates the comparison of our results with other machine learning classifiers and published work on the basis of the network flow analysis.

Table 5. Comparison with other published work and other machine learning classifiers.

Scenario #	Approaches	False Alarm Rate	Accuracy %
1	Wen-Hwa et al. [53]	0	98%
2	Fedynyshyn et al. [54]	7.8	92.9%
3	Naive Bayes classification algorithm	3	75%
4	Logistic Regression algorithm	3	93.8%
5	Artificial Neural Network (ANN)	3	93.2%
6	K-Nearest Neighbor (KNN)	3	93.9%
7	Our Proposed Model	3	98.7%

In this study, we proposed a general methodology for malicious network traffic analysis, discussed the involved issues, and presented a software platform that can be used for this kind of study. Our model overpasses the previously reported models [55] in terms of vulnerability against different attacks. For example, the model reported in [55] was only designed to handle worm attacks; however, our proposed model is able to provide security against multiple types of botnet attacks that spread over the Internet such as Internet Relay Chat (IRC), SPAM, Click Fraud, Port Scan, DDoS, Fast-Flux, Port Scan, Compiled and Controlled record by CTU, HTTP, Waledac, Storm and Zeus botnets. Tables 3 and 4 describe and quantify different types of attacks in detail. For example, our proposed method can effectively detect worm traffic such as Storm (worm) traffic, Waledac (fake SMTP and UDP) traffic, Zues (Trojan horse) traffic, included in ISOT dataset, and IRC (to block normal conversation and increase server load), Spam traffic (emails containing malicious links), Click Fraud for example (pay per click), Port Scan (an application designed to probe a server or host to exploit vulnerabilities), DDoS (Distributed Denial of Service attacks), and Fast Flux (fake domain name system that host malicious contents), included in the CTU dataset. Our study uses the Decision Tree classification algorithm to achieve high botnet detection accuracy for high-speed network environment since the principle of each classification tree is recursively from top to bottom, and its training set is obtained by returning the original training dataset. In order to minimize the occurrence of the fitting phenomenon, the Decision Tree uses the Bagging random sampling method to construct the classification tree.

Using the Naive Bayes classification algorithm, Logistic Regression, Artificial Neural Network, and K-Nearest Neighbor, the detection rate was 75.5% and 93.8%, 93.2% and 93.9%, respectively. However, the results of Decision Tree algorithm was significantly higher as 98.7%. Therefore, the Decision Tree algorithm for various types of P2P botnet traffic detection is more accurate than the other four classification algorithms. Moreover, the proposed method was also compared with previously published work as Wen-Hwa et al. [53] achieved 98% with a 0 false alarm rate, while they did not quantify the behaviour of the captured packets. However, with such a low false rate, the normal traffic flow will be very challenging. Another solution was proposed by Fedynyshyn et al. [54], where they achieved 92.9% with a relatively higher false alarm rate of 7.8% and they proposed a method to detect only IRC and C & C traffic.

5. Conclusions

This study demonstrates some efficient results by analysing the network traffic about the most relevant attacks: IRC, SPAM, Click Fraud, DDoS, FastFlux, Port Scan, Compiled and Controlled record by CTU, HTTP, Waledac, Storm and Zeus botnets. Our results prove the significance of the proposed framework as compared to the results obtained from different classifiers. It was observed that the Decision Tree algorithm had a high accuracy to detect P2P botnet traffic. A multi-layer hybrid technique for P2P botnet detection is proposed on the basis of session features. First, non-P2P traffic was filtered by packet, stream and session levels, respectively. Next, P2P botnet classifiers were used to classify the Normal P2P communication and P2P botnet on the basis of session features. Finally, this study combines the advantages of two different detection strategies i.e., Detection Methods Based on traffic behavior and Detection Method Based on Flow Similarity. The validity of the proposed method was verified by using aforementioned datasets. The network environment was set up in the lab of cyber security at UESTC. The experimental results show the significance of the multi-layer technique in the detection of P2P botnet traffic effectively, using the decision tree classifier. We evaluated the model by comparing the five different classifiers and two other previously published models and noted that our proposed model has a higher accuracy than others.

Author Contributions: Formal analysis, R.K.; Methodology, R.K.; Project administration, X.Z.; Validation, R.U.K.; Writing—original draft, R.K.;Writing—review and editing, A.S. Conceptualization, R.U.K. and R.K.; Methodology, R.U.K.; Software, R.U.K.; Validation, R.K, M.A, A.S and N.A.G; Formal Analysis, R.U.K., M.A, A.S. and N.A.G; Investigation, R.U.K, X.Z. and R.K; Resources, M.A and X.Z.; Data Curation, R.U.K, R.K, and N.A.G; Writing—Original Draft Preparation, R.U.K; Writing—Review and Editing, M.A, X.Z., A.S., N.A.G and R.K; Visualization, R.U.K, R.K.; Supervision, X.Z. and M.A; Project Administration, X.Z.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grant 61572115, and in part by the Sichuan Department of Science and Technology under Grant 2018GZ0218.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ma, X.; Zhang, J.; Tao, J.; Li, J.; Tian, J.; Guan, X. DNSRadar: Outsourcing malicious domain detection based on distributed cache-footprints. *IEEE Trans. Inf. Forensics Secur.* **2014**, *9*, 1906–1921. [CrossRef]
2. Zhao, S.; Lee, P.P.; Lui, J.; Guan, X.; Ma, X.; Tao, J. Cloud-based push-styled mobile botnets: A case study of exploiting the cloud to device messaging service. In Proceedings of the 28th Annual Computer Security Applications Conference, Orlando, FL, USA, 3–7 December 2012; pp. 119–128.
3. Ma, X.; Guan, X.; Tao, J.; Zheng, Q.; Guo, Y.; Liu, L.; Zhao, S. A novel IRC botnet detection method based on packet size sequence. In Proceedings of the 2010 IEEE International Conference on Communications (ICC), Cape Town, South Africa, 23–27 May 2010; pp. 1–5.
4. Alazab, M.; Broadhurst, R. Spam and criminal activity. In *Trends and Issues in Crime and Criminal Justice*; Australian Institute of Criminology: Canberra, Australia, 2016.
5. Arora, A.; Yadav, S.K.; Sharma, K. Denial-of-Service (DoS) Attack and Botnet: Network Analysis, Research Tactics, and Mitigation. In *Handbook of Research on Network Forensics and Analysis Techniques*; IGI Global: Hershey, PA, USA, 2018; pp. 117–141.
6. Zhang, J.; Xie, Y.; Yu, F.; Soukal, D.; Lee, W. Intention and Origination: An Inside Look at Large-Scale Bot Queries. In Proceedings of the NDSS Symposium 2013, San Diego, CA, USA, 24–27 February 2013.
7. Roto, V.; Oulasvirta, A.; Haikarainen, T.; Kuorelahti, J.; Lehmuskallio, H.; Nyyssönen, T. You Are a Game Bot!: Uncovering Game Bots in MMORPGs via Self-similarity in the Wild. In Proceedings of the NDSS 2016, San Diego, CA, USA, 21–24 February 2016; pp. 1–19. [CrossRef]
8. Krueger, T.; Gascon, H.; Krämer, N.; Rieck, K. Learning stateful models for network honeypots. In Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence (AISec '12), Raleigh, NC, USA, 19 October 2012; p. 37. [CrossRef]
9. Zhang, H.; Lu, G.; Qassrawi, M.T.; Zhang, Y.; Yu, X. Feature selection for optimizing traffic classification. *Comput. Commun.* **2012**, *35*, 1457–1471. [CrossRef]
10. Chen, C.M.; Lai, G.H.; Young, P.Y. Defense Joint Attacks Based on Stochastic Discrete Sequence Anomaly Detection. In Proceedings of the 2016 11th Asia Joint Conference on Information Security (AsiaJCIS), Fukuoka, Japan, 4–5 August 2016; pp. 74–79. [CrossRef]
11. Azab, A.; Alazab, M.; Aiash, M. Machine Learning Based Botnet Identification Traffic. In Proceedings of the 2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, 23–26 August 2016; pp. 1788–1794. [CrossRef]
12. Wang, S.; Yan, Q.; Chen, Z.; Yang, B.; Zhao, C.; Conti, M. Detecting android malware leveraging text semantics of network flows. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 1096–1109. [CrossRef]
13. Albanese, M.; Jajodia, S.; Venkatesan, S. Defending from stealthy botnets using moving target defenses. *IEEE Secur. Priv.* **2018**, *16*, 92–97. [CrossRef]
14. Haddadi, F.; Zincir-Heywood, A.N. Botnet behaviour analysis: How would a data analytics-based system with minimum a priori information perform? *Int. J. Netw. Manag.* **2017**, *27*, e1977. [CrossRef]

15. Alazab, M. Profiling and classifying the behavior of malicious codes. *J. Syst. Softw.* **2015**, *100*, 91–102. [CrossRef]
16. Alazab, M.; Venkataraman, S.; Watters, P. Towards understanding malware behaviour by the extraction of API calls. In Proceedings of the 2010 Second Cybercrime and Trustworthy Computing Workshop, Ballarat, VIC, Australia, 19–20 July 2010; pp. 52–59.
17. Zhang, J.; Perdisci, R.; Lee, W.; Luo, X.; Sarfraz, U. Building a Scalable System for Stealthy P2P-Botnet Detection. *IEEE Trans. Inf. Forensics Secur.* **2014**, *9*, 27–38. [CrossRef]
18. Abdullah, R.S.; Abdollah, M.F.; Noh, Z.A.M.; Mas'ud, M.Z.; Sahib, S.; Yusof, R. Preliminary study of host and network-based analysis on P2P Botnet detection. In Proceedings of the 2013 International Conference on Technology, Informatics, Management, Engineering and Environment, Bandung, Indonesia, 23–26 June 2013; pp. 105–109. [CrossRef]
19. Yin, C. Towards accurate node-based detection of P2P botnets. *Sci. World J.* **2014**, *2014*, 425491. [CrossRef]
20. Botnet detection based on traffic behavior analysis and flow intervals. *Comput. Secur.* **2013**, *39*, 2–16. [CrossRef]
21. Sharifnya, R.; Abadi, M. DFBotKiller: Domain-flux botnet detection based on the history of group activities and failures in DNS traffic. *Digit. Investig.* **2015**, *12*, 15–26. [CrossRef]
22. Buczak, A.L.; Guven, E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1153–1176. [CrossRef]
23. Ye, W.; Cho, K. P2P and P2P botnet traffic classification in two stages. *Soft Comput.* **2017**, *21*, 1315–1326. [CrossRef]
24. Ye, W.; Cho, K. Hybrid P2P traffic classification with heuristic rules and machine learning. *Soft Comput.* **2014**, *18*, 1815–1827. [CrossRef]
25. Dainotti, A.; King, A.; Claffy, K.; Papale, F.; Pescapé, A. Analysis of a/0 stealth scan from a botnet. *IEEE/ACM Trans. Netw.* **2015**, *23*, 341–354. [CrossRef]
26. Guofei, G.; Yegneswaran, V.; Porras, P.; Stoll, J.; Lee, W. Active botnet probing to identify obscure command and control channels. In Proceedings of the IEEE Annual Computer Security Application Conference, Honolulu, HI, USA, 7–11 December 2009; pp. 241–253.
27. Guofei, G.; Perdisci, R.; Zhang, J.; Lee, W. BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In Proceedings of the 17th Conference on Security Symposium, San Jose, CA, USA, 28 July–1 August 2008; pp. 139–154.
28. Zhang, J.; Perdisci, R.; Lee, W.; Sarfraz, U.; Luo, X. Detecting stealthy P2P botnets using statistical traffic fingerprints. In Proceedings of the 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN), Hong Kong, China, 27–30 June 2011; pp. 121–132. [CrossRef]
29. Gu, G.; Zhang, J.; Lee, W. BotSniffer: Detecting botnet command and control channels in network traffic. In Proceedings of the 15th Annual Network and Distributed System Security Symposium, San Diego, CA, USA, 8–11 February 2008; p. 18.
30. Gupta, B.; Badve, O.P. Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a cloud computing environment. *Neural Comput. Appl.* **2017**, *28*, 3655–3682. [CrossRef]
31. Wang, P.; Wu, L.; Aslam, B.; Zou, C.C. Analysis of Peer-to-Peer botnet attacks and defenses. In *Propagation Phenomena in Real World Networks*; Springer: Cham, Switzerland, 2015; pp. 183–214.
32. Wang, C.; Zhou, X.; You, F.; Chen, H. Design of P2P Traffic Identification Based on DPI and DFI. In Proceedings of the 2009 International Symposium on Computer Network and Multimedia Technology, Wuhan China, 18–20 January 2009; pp. 1–4. [CrossRef]
33. Yue, W.T.; Wang, Q.H.; Hui, K.L. See No Evil, Hear No Evil? Dissecting the Impact of Online Hacker Forums. *MIS Q.* **2019**, *43*, 73–95. [CrossRef]
34. Bhuyan, M.H.; Bhattacharyya, D.K.; Kalita, J.K. Surveying port scans and their detection methodologies. *Comput. J.* **2011**, *54*, 1565–1581. [CrossRef]
35. IANA. Service Name and Transport Protocol Port Number Registry. Available online: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml (accessed on 20 April 2019).
36. Stanek, W. Name-Resolution Services. *Windows Server 2012 Pocket Consultant*; The Microsoft Press Store by Pearson: Redmond, WA, USA, 2012; Chapter Windows Se.

37. Papernot, N.; McDaniel, P.; Wu, X.; Jha, S.; Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 23–25 May 2016; pp. 582–597.
38. Kim, K. A hybrid classification algorithm by subspace partitioning through semi-supervised decision tree. *Pattern Recognit.* **2016**, *60*, 157–163. [CrossRef]
39. Davis, J.; Goadrich, M. The Relationship Between Precision-Recall and ROC Curves. In Proceedings of the 23rd International Conference on Machine Learnin, Pittsburgh, PA, USA, 25–29 June 2006; pp. 1–8.
40. Peng, P.; Xiang, T.; Wang, Y.; Pontil, M.; Gong, S.; Huang, T.; Tian, Y. Unsupervised cross-dataset transfer learning for person re-identification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 1306–1315.
41. Garcia, S.; Grill, M.; Stiborek, J.; Zunino, A. An empirical comparison of botnet detection methods. *Comput. Secur.* **2014**, *45*, 100–123. [CrossRef]
42. Chowdhury, S.; Khanzadeh, M.; Akula, R.; Zhang, F.; Zhang, S.; Medal, H.; Marufuzzaman, M.; Bian, L. Botnet detection using graph-based feature clustering. *J. Big Data* **2017**, *4*, 14. [CrossRef]
43. Saad, S.; Traore, I.; Ghorbani, A.; Sayed, B.; Zhao, D.; Lu, W.; Felix, J.; Hakimian, P. Detecting P2P botnets through network behavior analysis and machine learning. In Proceedings of the 2011 9th Annual International Conference on Privacy, Security and Trust (PST 2011), Montreal, QC, Canada, 19–21 July 2011; pp. 174–180. [CrossRef]
44. Sherif, S.; Traore, I.; Ghorbani, A.A.; Sayed, B.; Zhao, D.; Lu, W.; Felix, J.; Hakimian, P. ISOT Dataset Description. In Proceedings of the 9th Annual Conference on Privacy, Security and Trust (PST2011), Montreal, QC, Canada, 19–21 July 2011; pp. 19–21.
45. CiTRIX. How to Decrypt SSL and TLS Traffic Using Wireshark. Available online: https://support.citrix.com/article/CTX116557 (accessed on 20 April 2019).
46. Martin, H.; Milan, Č.; Tomáš, J.; Pavel, Č. HTTPS Traffic Analysis and Client Identification Using Passive SSL/TLS Fingerprinting. *EURASIP J. Inf. Secur.* **2016**, *1*, 1–22.
47. Radivilova, T.; Kirichenko, L.; Ageyev, D.; Tawalbeh, M.; Bulakh, V. Decrypting SSL/TLS traffic for hidden threats detection. In Proceedings of the 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT), Kiev, Ukraine, 12 July 2018; pp. 143–146. [CrossRef]
48. Meyer, C.; ePrint Archive, J.S.I.C.; 2013, U. Lessons Learned From Previous SSL/TLS Attacks—A Brief Chronology Of Attacks And Weaknesses. *IACR Cryptol. EPrint Arch.* **2013**, *49*, 1–14.
49. Khan, R.U.; Zhang, X.; Kumar, R. Analysis of ResNet and GoogleNet models for malware detection. *J. Comput. Virol. Hacking Tech.* **2018**, *15*, 29–37. [CrossRef]
50. Khan, R.U.; Zhang, X.; Kumar, R.; Aboagye, E.O. Evaluating the Performance of ResNet Model Based on Image Recognition. In Proceedings of the 2018 International Conference on Computing and Artificial Intelligence (ICCAI 2018), Chengdu, China, 12–14 March 2018; pp. 86–90. [CrossRef]
51. Kumar, R.; Zhang, X.; Khan, R.U.; Ahad, I.; Kumar, J. Malicious Code Detection Based on Image Processing Using Deep Learning. In Proceedings of the 2018 International Conference on Computing and Artificial Intelligence (ICCAI 2018), Chengdu, China, 12–14 March 2018; pp. 81–85. [CrossRef]
52. Kumar, R.; Zhang, X.; Khan, R.; Sharif, A.; Kumar, R.; Zhang, X.; Khan, R.U.; Sharif, A. Research on Data Mining of Permission-Induced Risk for Android IoT Devices. *Appl. Sci.* **2019**, *9*, 277. [CrossRef]
53. Liao, W.H.; Chang, C.C. Peer to peer botnet detection using data mining scheme. In Proceedings of the 2010 International Conference on Internet Technology and Applications, Wuhan, China, 20–22 August 2010; pp. 1–4.
54. Fedynyshyn, G.; Chuah, M.C.; Tan, G. Detection and classification of different botnet C&C channels. In *International Conference on Autonomic and Trusted Computing*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 228–242.
55. Dainotti, A.; Pescape, A.; Ventre, G. Worm Traffic Analysis and Characterization. In Proceedings of the 2007 IEEE International Conference on Communications, Glasgow, UK, 24–28 June 2007; pp. 1435–1442. [CrossRef]