Hindawi
Security and Communication Networks
Volume 2020, Article ID 9302318, 9 pages
https://doi.org/10.1155/2020/9302318

Research Article

Botnet Forensic Analysis Using Machine Learning

Anchit Bijalwan

Faculty of Electrical and Computer Engineering, Arba Minch University, Arba Minch, Ethiopia

Correspondence should be addressed to Anchit Bijalwan; anchit.bijalwan@amu.edu.et

Received 1 August 2019; Revised 20 December 2019; Accepted 21 January 2020; Published 20 February 2020

Academic Editor: Huaizhi Li

Botnet forensic analysis helps in understanding the nature of attacks and the modus operandi used by the attackers. Botnet attacks are difficult to trace because of their rapid pace, epidemic nature, and smaller size. Machine learning works as a panacea for botnet attack related issues. It not only facilitates detection but also helps in prevention from bot attack. The proposed inquisition model endeavors improved quality of results by comprehensive botnet detection and forensic analysis. This scenario has been applied in eight different combinations of ensemble classifier technique to detect botnet evidence. The study is also compared to the ensemble-based classifiers with the single classifier using different parameters. The results exhibit that the proposed model can improve accuracy over a single classifier.

1. Introduction

The intelligent learning system can read the user's actions and behavior in the cyber world. It can easily detect the behavioral nature and aspect of every activity on social media. However, the black hat community works only in the self-interest and focuses on propagating malicious activities. The botnet is one of the most emerging threats for the digital society.

A botnet is a collection of zombie networks whose tendency is to propagate bot continuously. A bot is a malicious program that acts upon botherder's command. Botherder executes this bot illegally further for the self-interest, which is called bot attack [1]. Bot attack is difficult to handle as botnet rapidly germinates in order to get off the detection process. Due to this dynamic behavior, the value of botnet information degrades quickly. In order to detect and analyze botnet attack, the dataset has been taken, which is completely implemented in a physical testbed environment. It uses real devices for generating the real traffic. This dataset contains both training set for real traces and testing set for normal and botnet traffic.

Botnet analysis is utilized for detecting the nature and kind of attack. This can be executed by disparate machine learning algorithms. These machine learning models may give different results but the model with comparatively better result can be taken as the best-fitted model.

Botnet detection can be improved by the SVM machine learning classification technique and packet histogram vector [2]. The textual spam e-mail classification can be analyzed using KNN model [3]; the author used summarization technique for knowledge extraction. P2P botnet traffic can be classified by differentiating the features using the machine learning algorithm [4]. The authors extracted 17 features first and then removed five features from them because of the nominal values. Subsequently, they bifurcated in the host and flow based feature. A framework can be also built with the help of Hive and Mahout Model to detect peer to peer botnet attacks using machine learning approach [5]. Bot activity [6] is detected by both command and control and attack phase using traffic behavior analysis and by applying machine learning classification. The author detected the bot activity with the help of decision tree classifier as a machine learning framework. After converting time domain network communication to frequency domain network, Narang et al. [7] proposed the work for detecting P2P botnet traffic. They used a machine learning approach by applying signal processing for making each pair of nodes. Barthkur et al. [8] exhibited the difference between flow feature P2P and P2P traffic for binary classification. They

combined both P2P and web-based traffic and finally classified the P2P data by applying optimum SVM model. On the other hand, conceptual DDoS detection and mitigation model designed by the ensemble classifier [9] shows that the classifier can be built through multiple data chunks. A classification model is also built in a real streaming environment in lieu of manual labeling of data [10]. The authors have taken unlabeled and some amount of labeled data from the trained set with KNN. The study results showed that it is also possible to improve results by merging more than one algorithm [11]. Masud et al. [12] advocate that the previous work was based on the technique for building one classifier per chunk. The author further concluded that the multiple chunks with the multiple ensembles can improve the classification technique. Similarly, Liu et al. [13] used binary classification problem with the help of an ensemble of a classifier. They have done this experiment through multisample train set and compared the performance of Bagging, Adaboost, Asyboost, Random forest etc. Galar et al. [14] implemented framework for imbalanced dataset using ensemble technique. The authors proposed taxonomy of class imbalance categorized by inner ensemble methodology. Mckay et al. [15] have shown their work through random forest, KNN, and J48 machine learning algorithm. Nazemi Gelian et al. [16] proposed a self-learning botnet detection system through which ensemble classifier enhanced its generalization capability.

Most of the researchers have utilized a single combination based ensemble of classifier, whereas, here, eight different combinations of an ensemble-based classifier have been applied. The contribution of this paper can be understood by reading the following:

(1) The proposed botnet inquisition model aims at improving the quality of results by considering every aspect of detection, analyzation, and forensics of botnet.

(2) The improved probability of detecting the accuracy values of attack intentions.

(3) The enhanced efficiency of an ensemble of ensemble classifier to detect the botnet is more than a single classifier.

(4) Comparative chart for accuracy, precision, recall, and F1 score using eight different combinations of an ensemble-based classifier.

(5) Comparative analysis of ensemble classifier proposed by authors.

This paper shows the inquisition model of botnet forensics in Section 2 and the machine learning model in Section 3. The improvement in the accuracy for identifying and detecting the botnet is shown in Section 4 and final conclusion in Section 5.

2. Botnet Inquisition Model

There are two ways of evaluating the network security aspects, that is, prevention and detection. The prevention mechanism is being done by firewall and Intrusion Prevention System (IPS), and the detection can be done by Intrusion Detection System (IDS). Botnet forensics uses postmortem techniques to collect, identify, detect, examine, analyze, and postmortem document for bot shreds of evidence from digital sources. It uses network security tools to uncover facts related to the cybercrimes specifically on the botnet. The major challenge of the botnet forensics is to analyze digital evidence of cybercrime. The term Botnet forensics was first coined by Anchit Bijalwan in 2013 [17].

Generally, botnet forensics analysis faces many challenges. It requires an efficient repository that can be obtained through the passive deployment of vulnerable systems to be compromised. Attackers can use encrypted malware traffic by modifying web traffic for the detection and analysis aberration, reconstruction, attack behavior, and so forth. Normally, it has to reconnoiter full traces of malicious behavior in order to get through the nature of the attack. The classification and clustering process can be applied when there is a protocol's complexity. Furthermore, the reconstruction method is used to understand the purpose of attack and to resolve the convoluted shreds of evidence.

This proposed inquisition model is able to refurbish the quality of results of the malicious evidence analysis specifically for a botnet. It incorporates all the information at different levels of the model by tracing and detecting the anomaly and by applying the forensics. These results curtail the time duration of the made decision in the botnet investigation phase. In general, most of the frameworks hinge upon distance, feature, or probabilistic measurements. However, this inquisition model is often used in the alert correlation techniques, which depends on the attack attributes.

The model primarily focuses on investigating the various kinds of botnet attacks. It helps in identification, detection, and classification of botnet and analyzes the attack intentions. It further visualizes and generates the report so that such bot attacks can be prevented in the future. The entire process requires deep investigation and analysis of various factors; therefore the term "inquisition" is used in the title of the model.

The model analyzes various attacks including cybercrime on the networks. It computes the probability of detecting the accuracy values of the attack intentions and performs calculation with the help of various algorithms like attack intention analysis (AIA). It also gives a list of probability attack intentions depending on the relevant evidences. The Dempster–Shafer (D-S) evidence theory with causal networks can also be used to get a better estimation of the attack purpose. This evidence theory is used to compute the probability of attack intentions as it provides better values and better accuracy. Figure 1 represents the proposed model for detecting the network.

2.1. Data Sources. This is the first phase of the botnet forensics model, which is utilized for collecting all the data traffic and packets from the network or the system. It is responsible for collecting all the ingress and the egress packets from the network. Further it captured and

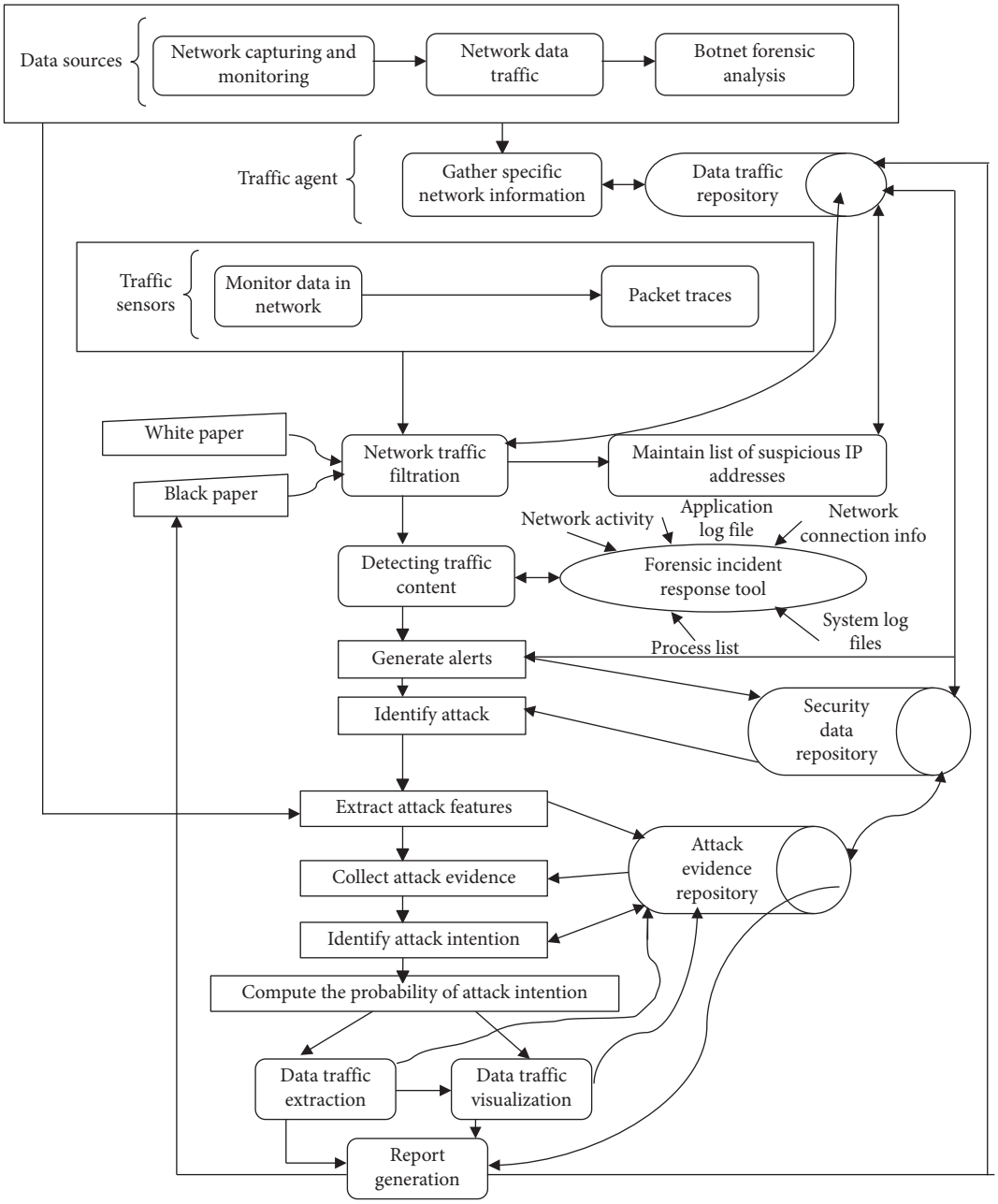

Figure 1: Botnet inquisition model.

monitored all the data traffic and the packets from the network and then analyzed the entire network traffic. This was done by using different botnet forensic analysis and monitoring tools such as Wireshark, Tcpdump, and Silent Runner.

2.2. Traffic Agents. The specific information collected in the previous phase is gathered. The information that is useful in detecting the attacks and collecting all the packet traces to identify the attack intentions is gathered. All the information and packet traces are collected in the data traffic repository so that no crucial information can be lost. The data traffic repository can be utilized for future use and can retrieve the data or the packets as and when required.

2.3. Traffic Sensors. All those network packet traces or the required information that is gathered in the previous phase using different network monitoring tools like packet capturing, fingerprinting, IDS, and Pattern Matching and statistics such as Ngrep, Bro, Snort, Argus, and Wireshark is monitored. These packets are analyzed to collect the traces of the attack and to identify the intentions of the attacker.

2.4. Network Traffic Filtration. In this phase, all the packets that have been captured in the earlier phase are filtered. It gives full concentration on unwanted packets, thus resulting in the reduced workload. There are two ways of doing it, that is, the whitelist and the blacklist filtering.

2.4.1. Whitelist. The whitelist is the set of packets that are not infectious in nature. For instance, windows update, antivirus update, and a list of known sites are examples of the whitelist.

2.4.2. Blacklist. In this section of network traffic filtration, blacklist filtering weeds out the infectious packets. These details can be utilized for filtering out the malware and detecting whether they are bots or not in the next phase. After filtering out those packets, a list of all the suspicious IP addresses that make the work easier is also maintained. This helps in finding out all the malicious activities done on the network or the system. Further, it can also easily identify the attack intentions. A list from the data traffic repository can be maintained and can be saved. In the future, the same facilitates an easy identification of the malicious activities on the network.

2.5. Detecting Malicious Traffic Content. This phase mainly aims at identification and detection of unknown packets or the infected data traffics that go through the blacklist filtration. The forensic incident response tool is utilized for infected packets detection. The forensic incident response tool facilitates network connection info, application log, system log, process list, and many other functions. It is followed by detection and identification of the organization's policy, legal issues, and business constraints. The role of the incident response tool is vital in deciding whether to carry on the investigation and collect more traces or to abort the process.

Further, alert generator generates the alerts in order to enable the network forensic investigation. A copy of the captured data is analyzed to identify the attack alerts. Alerts are generated on the basis of matching the pattern of the known or unknown packets that are collected in the previous phase. After generating the alerts, all the malicious packets or unknown data that could be malicious in nature are saved to the security data repository so that these packets could not be lost as they are very crucial for attack intentions in botnet forensic inquisition. This information can be used from the security data to identify whether it is an attack or not. This identification is done on the generated alerts. This can also use the data saved in the data traffic repository to generate the alerts and to identify an attack. These repositories are linked together for finding an attack and saved the data traffic securely.

Feature extraction is used after identifying the attack alert. The attack evidence can be collected and saved in the attack evidence repository for the future use. After collecting the evidence, all the attack features like how the attack has occurred, who was involved in that attack, duration of the exploit, and the methodology used in the attack can be extracted. Each and every possible feature of the attack is extracted so that the attack intentions can be identified, which is the main purpose in this proposed framework for botnet forensics inquisition. Attack evidence and security data repository are linked together to collect the evidences and to save them securely for the future. Various machine learning algorithms can be applied for detecting malignant and benign data. For this work, ensemble-based classification techniques have been applied for detecting malicious and benign data.

2.6. Attack Intention. The attack intention probability is computed with either Dempster–Shafer's evidence theory or AIA algorithm. All the values are associated with a relevant attack to generate the value of the attack intention. These are the main aim and specialty of the framework, which differentiate it from the other existing frameworks. It employs probability values to approximate the attack intentions to determine the similarity of the new attacks with the other predefined intentions.

There are already a defined set of values which contains all the previous attacks and another set of values which contains the attack intentions for all the predefined attacks. Using any given algorithms, estimating the similarity of values between the new attack intentions and the others is necessary. It identifies the attacks that contain one or more attack intentions and computes the sum of all the probability values of the attack intentions that are relevant. Different techniques differentiate the stage of the attack and determine the target. The stage of attack can be bifurcated on the grounds of increased access based, disclosure of information based, and denial of service based. Further, it can be observed through targets such as a file, computer, or network and analyzed through intruder skills, capability, and tools. It determines the threat estimation, intention list, and attack probability.

2.7. Data Traffic Extraction/Visualization. The circumstances of attack and motive can be explained and proven by extracting the relevant information from the collected values. Data visualization helps in presenting the situation. Complete information about that attack is maintained in a log that validates those packets or collected information. It takes all the required information from the attack evidence repository and maintains an attack log by taking those values. Further, the attack log and evidence are used to identify the attack intention for botnet forensics inquisition. Visualization can be done by separating the normal traffic and botnet traffic. Further, botnet traffic is used for analysis through the ensemble classifier algorithm and tools such as NetMate and Orange.

Report generation is the final phase of the framework in which observations are presented in an understandable format, providing an explanation of the various procedures to arrive at the conclusion of detection of attacks and identify an intention of the attack. The required information has been taken from the attack evidence repository and the attack log maintained in it and generated a document or a report based on those shreds of evidence. It also updates the data traffic repository for finding out the new malicious packets as well as for updating the list of suspicious IP addresses. A detailed review of an entire case documentation is done for future examination, detection, and identification of the attack intentions.

The inquisition model is compared with the previous similar models such as DFRDS, Reith et al.'s, Prosise et al.'s, Seamus et al.'s, Beebe et al.'s, Ren et al.'s, Pilli et al.'s, Thapaliyal et al.'s, and and Bijalwan et al. All the previous similar models have not defined identity attack intention, probability of attack intention, and traffic extraction/visualization. That makes this model more refined than others. This model exhibits computing probability of attack intention using Dempster–Shaffer's theory or artificial immune algorithm (AIA). The result will find out new vulnerabilities that help improve the decision-making process. This proposed model provides better results and accuracy for the detection and identification of the attack intention. The dependencies of packet attribute from various tools and reconnaissance of attributes from different host validate an attack.

2.8. Botnet Analysis Using Ensemble of Classifier. Botnet inquisition framework provides the details of botnet detection and its analysis through step by step process. It is obvious to analyze the botnet when botnet identification process gets completed. There is a different way to get the features extracted and to analyze them. This machine learning model refers to the classification technique to analyze the data and has taken the ensemble of classifier; the machine learning algorithm is used to improve the accuracy in detecting the botnet. The work particularly deals with the specific kind of botnet dataset which infiltrates the network from inside denial of service (DoS), distributed denial of service (DDoS), and brute force data. Collected botnet traffic is the ingestion of SSH, HTTP, and SMTP traffic that refers to the user's behavior. It is further classified and characterized through set of attributes which distinguishes the malicious traffic from normal traffic. This dataset is next filtered into normal traffic and botnet traffic and botnet traffic sample is selected for further analysis. This process has extracted 42 attributes, provided labels to every instance, and bifurcated them into training and testing datasets.

Maximum attributes are extracted from TCP/UDP headers directly such as source IP and destination IP. These 42 extracted attributes are srcip (source IP address), srcport (source port no.), dstip (destination IP address), dstport (destination port no.), proto (protocol), total_fpackets (total packets in forward direction), total_fvolume (total bytes in forward direction), total_bpackets (total packets in backward direction), total_bvolume (total bytes in backward direction), min_fpktl (minimum packet size in forward direction), mean_fpktl (mean packet size in the forward direction), max_fpktl (maximum packet size in the forward direction), std_fpktl (standard deviation of packet length in forward direction), min_bpktl (minimum packet size in backward direction), mean_bpktl (mean packet size in the backward direction), max_bpktl (maximum packet size in the backward direction), std_bpktl (standard deviation of packet length in backward direction), min_fiat (minimum time between two packets in forward direction), mean_fiat (mean time between two packets in forward direction), max_fiat (maximum time between two packets in forward direction), numroot (number of root accesses), rootshell (if rootshell is generated), numcompromised (number of compromised conditions), suattempted (attempted su root command), hot (number of hot indicators), num_le_creation (operation on number of file creations), aglaud (average payload packet length), numaccess_les (number of operations on access control file), count (in last two seconds, number of connections), duration (number of seconds of the connection), std_fiat (standard deviation time between two packets in forward direction), min_biat (minimum time between two packets in backward direction), mean_biat (mean time between two packets in backward direction), max_biat (maximum time between two packets in backward direction), std_biat (standard deviation time between two packets in backward direction), sflow_fbytes (subflow of forward direction in average number of bytes), sflow_bpacket (subflow of backward direction in average number of packets), sflow_bbytes (subflow of backward direction in average number of bytes), sflow_fpackets (subflow of forward direction in average number of packets), total_fhlen (total size of forward packet), total_bhlen (total size of backward packet), and mean active (mean time of active flow before idle state).

Attributes are extracted by two types of segregation, that is, host based and flow based. Network flows refer to the set of attributes' extraction. P2P traffic and non-P2P traffic are obtained from these attributes by link flows. Flow vectors are utilized and inserted into NetMate and Orange tool for extracting 42 different attributes. Further, it is labeled into normal traffic and P2P botnet traffic. Normal traffic is legitimate traffic.

(Figure 2) shows how the botnet analysis model works. The dataset was extracted into normal and malicious traffic. Here malicious traffic was taken for the further machine learning analysis. For this purpose, the training and testing set used for extraction and all inputs were given to the model. The classification model was applied here for further analysis. On the other hand, quality metric that refers to error differences were also set with the machine algorithm and applied to the classification model intended for the final output. Normal traffic is legitimate traffic so the process has not paid heed on it, especially for normal P2P traffic. P2P botnet traffic is basically fraught with different bot traces. Therefore, this process has not used both collected traffics.

Machine learning ensemble of classifier algorithm refers to the multiple combinations of the single classifier so that the power of detecting botnet clues can be increased. This model is a combination of bagging, AdaBoost, and soft-voting method of ensemble-based classifier. It also compared the performance of each classifier based on its accuracy to predict classes of unknown instances as mentioned in Algorithm 1.

Assume an example E of N classifier, that is, $\{E_1, E_2, E_3 \ldots, E_N\}$.

Ensemble E is actually having two-level ensemble itself so each classifier E_x in the ensemble E is actually a collection of ensembles of N classifier.

Each classifier E_i is at the middle level. Lowest level contains the actual classifier (Algorithm 1).

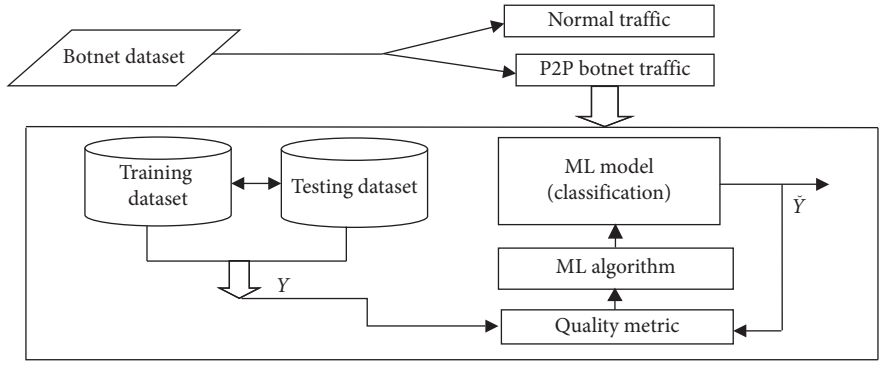

Figure 2: Botnet analysis model.

Input:
D: dataset; tnd: training data; tsd: testing data point; cl: class label; *f*: feature; *M*: model
Output:
(1) Obtaining tnd ($tnd_1 + tnd_2 + \ldots + tnd_n$) and tsd ($tsd_1 + tsd_2 + \ldots + tsd_n$) from D
(2) Extracting *f* from tnd and tsd of D
(3) Segregation on Normal and Botnet traffic
(4) If no cl on botnet traffic then
(5) Providing cl elseif
(6) Goto next step
(7) Frame *M*, test each *M* on cl data on tnd and tsd and obtain its accuracy
(8) Test M1 from knn, DT and svm
(9) Test ensemble M_2 from multiple combinations
(10) Compare step 8 and step 9
(11) $M \longleftarrow$ best from *M*1 & *M*2 models based on accuracy
(12) Predict the cl

Algorithm 1

Suppose that the middle-level ensemble E_i is trained with r followed wedges. As soon as new chunk appeared, it is necessary to train next middle-level ensemble till E_N.

Let data wedge $W = \{W_x, W_{x-1}, \ldots, W_{x-r+1}\}$ where W is randomly divided into n equal ports, that is, $\{W, W_1, W_2 \ldots, W_n\}$, where all ports will be having the same number of positive as well as negative examples.

Next build E_X with n classifier = $\{E_{X(1)}, E_{X(2)}, \ldots, E_{x(n)}\}$, where each classifier $E_{X(j)}$ is trained with the dataset and computed the expected error. Error of ensemble E_x is expected by testing each classifier $E_{X(j)}$ on W_j and averaging its error. Finally, the upper level ensemble E is updated by replacing middle level.

All the classifiers trained on instance sample were taken with replacement from the training set. Some instances have been represented many times. Figure 3 describes the flow diagram of an ensemble classifier.

A confusion matrix is an important tool for analyzing how well the classifier can recognize tuples of different classes. True positive (TP) and true negative (TN) are exhibited when the classifier is accepting right things. On the other hand, the false positive (FP) and false negative (FN) are exhibited when the classifier is accepting the wrong things. Table 1 presents confusion matrix shown with totals for positive and negative tuples. It shows the parameter taken for the evaluation.

3. Results and Discussions

3.1. Single Classifier. The botnet is a large network of compromised computers, which is instructed by botherder. The reactive approach refers to the evidence that should be preserved in one place for postmortem of bot attacks. This evidence is further applied for the analysis of botnet traffic and to retrieve the relevant information from it. For this purpose, machine learning model 1 has been taken, which reveals the analysis using a single classifier.

Table 2 presents the details of a single classifier. In this table, the decision tree algorithm shows 93.7% accuracy, 92.09% precision, 93.48% recall, and 94.76% F1 score. In the case of KNN algorithm, 94.65% accuracy, 95.0% precision, 93.48% recall, and 94.76% F1 score are observed. Subsequently SVM shows 75.99% accuracy, 81.07% precision, 76.05% recall, and 66.78% F1 score.

Figure 4 shows the comparison chart in single classifier. Red column exhibits the decision tree, blue column exhibits the KNN, and, subsequently, green column shows SVM.

3.2. Ensemble of Classifier. AdaBoost decision tree also increases accuracy from 93.7% to 98.36%, improving learning process of a decision tree, and highest accuracy is achieved by using soft-voting rule because it merges the powers of two

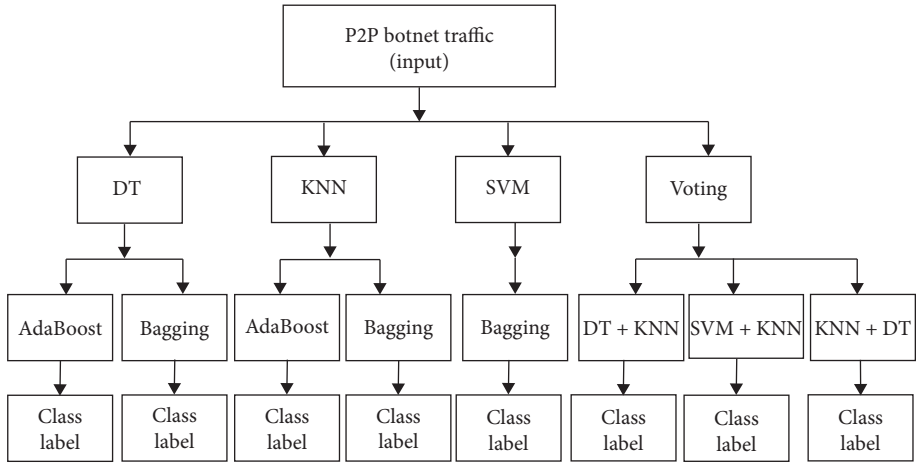

Figure 3: Ensemble of classifier method.

Table 1: Confusion matrix.

		Predicted class		
		Yes	No	Total
Actual class	Yes	TP	FN	P
	No	FP	TN	N
	Total	P	N	$P+N$

Table 2: Single classifier.

Classifiers	Decision tree	KNN	SVM
Accuracy	93.7	94.65	75.99
Precision	92.09	95.0	81.07
Recall	93.48	95.0	76.05
F1 score	94.76	95.0	66.78

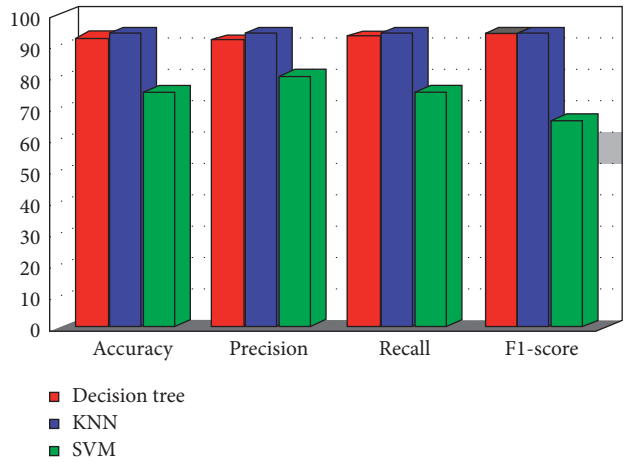

Figure 4: Comparison in single classifier.

algorithms and gives more weight to the decision of better performing algorithm. The output of a single classifier does not give perfect bot findings. The performance of bot evidence using the ensemble of a classifier is better than the single classifier.

Table 3 shows the comparison chart of the different ensemble of classifiers. As observed in the table performance of bagging-KNN, that is, 94.77%, which is better than KNN, that is, 94.65%, in Table 2, an ensemble of classifier reduces the variance in input data and avoids overfitting. It demonstrates that ensemble classifier is better than single classifier and gives highest accuracy, that is, 98.36% for AdaBoost-DT, 94.65% for AdaBoost-KNN, 95.30% for Bagging-DT, 94.77% Bagging-KNN, 75.99% for Bagging-SVM, 95.47% for Voting-KNN + DT, 85.06% for Voting-DT + SVM, and 94.65% for Voting-SVM + KNN. The AdaBoost with SVM decreases the performance because SVM is a strong learner, while AdaBoost is used mainly to improve weak learners. Secondly, AdaBoost provides sampling to train the instance according to the complexity of classification; that is, more weight is given to the instances that are hard to classify.

Figure 5 shows the comparison of the ensemble of classifier where white bar chart refers to the combining power of AdaBoost and decision tree, red bar shows the AdaBoost with KNN, green bar shows Bagging with DT, grey bar shows Bagging with KNN, blue bar shows Bagging with SVM, and yellow bar shows voting and KNN with DT.

4. Discussion

The results show that ensemble-based classifier provides better results because it is made up by combining multiple algorithms for botnet analysis. Observation showed decision trees are very flexible, easy to understand, and easy to debug. Simple decision trees tend to overfit the training data more so that other techniques generally have to do tree pruning and tune the pruning procedures. KNN keeps all the training data. Through KNN, calculations comparatively become larger and complexity is higher when a dimension is very low. A KNN calculation becomes larger because it calculates similarity from its nearest neighbors and after sorting them it applies majority voting on top K neighbors to predict the class of data point. Therefore, complexity is directly proportional to the value of K.

Table 3: Comparison chart of ensemble of classifier.

Classifier	AdaBoost-DT	AdaBoost-KNN	Bagging-DT	Bagging-KNN	Bagging-SVM	Voting-KNN + DT	Voting-DT + SVM	Voting-SVM + KNN
Accuracy	98.36	94.65	95.30	94.77	75.99	95.47	85.06	94.65
Precision	98.85	95.0	95.25	94.89	81.07	96.0	87.0	95.0
Recall	98.23	95.0	95.48	95.0	76.05	95.78	85.0	95.0
F1 score	98.54	95.0	95.76	94.42	66.78	95.23	83.0	95.0

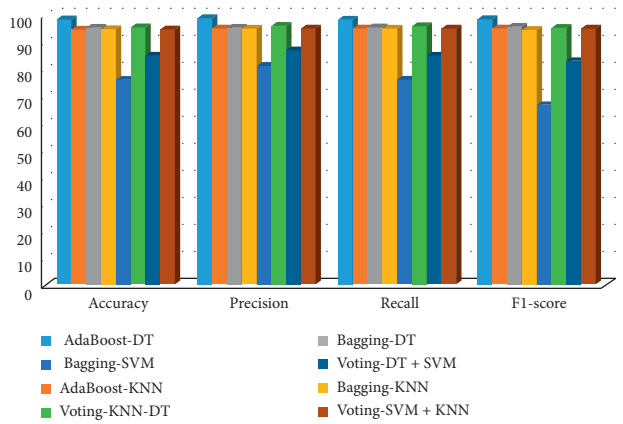

Figure 5: Ensemble comparison.

Table 4: Comparative chart among authors.

Ensemble of classifier	Li et al. [18]	Garg et al. [4]	Lin and Chen [19]	Ye et al. [20]	Bijalwan et al. [11]	Cadenas et al. [21]	Liu et al. [13]	Proposed work
AdaBoost-SVM	✓	X	X	X	X	X	X	X
Random forest	X	✓	✓	X	X	✓	✓	X
AdaBoost-DT	X	X	X	X	X	X	X	✓
AdaBoost-KNN	X	X	X	X	X	X	X	✓
AdaBoost	X	X	X	X	X	X	✓	X
Bagging-DT	X	X	X	X	✓	X	X	✓
Bagging-KNN	X	X	X	X	✓	X	X	✓
Bagging-SVM	X	X	X	✓	X	X	X	✓
Bagging	X	X	X	X	X	X	✓	X
Voting-KNN + DT	X	X	X	X	✓	X	X	✓
Voting-DT + SVM	X	X	X	X	X	X	X	✓
Voting-SVM + KNN	X	X	X	X	X	X	X	✓
Classifier	1	1	1	1	3	1	3	8

When all features give continuous real value, KNN provides the good result. When a number of features are very large as compared to the training samples, SVM cannot work efficiently. SVM should not be taken in the case of multiple classes. Here binary classifier can be taken and can use the voting method to classify any of the classes.

This model also compared the performances of all classifiers based on their accuracy, precision, recall, and F1 score to predict classes of unknown instances. The accuracy of the results shows that all the proportions of observed prediction are correctly taken, which is a sign of a good model. Here, results exhibit that ensemble of classifier model can detect botnet traffic more accurately than a single classification model. Precision refers to the proportion of all positive observations that are correct. F1 score refers to the harmonic mean (average) of both precision and recall. Table 4 [21] shows comparative analysis of other authors with proposed work.

5. Conclusion and Future Work

Botnet forensics uses scientific techniques to collect, examine, analyze, and document digital bot shreds of evidence from digital sources and network security tools. It uncovers facts related to the cybercrimes specific to the botnet. Inquisition model shows the mechanism for applying forensics on botnet traffic after passing from various phases. The existing classifiers have been combined in an ensemble

model for detecting the botnet traffic. This ensemble of different classifiers performs better because it is made up by combining the powers of multiple algorithms. It segregated the features into classes, that is, on normal traffic and botnet traffic, and provided labeling. Thereafter, by using data mining tool, ensemble of classifier algorithm has been applied. This result shows that the ensemble model improved various parameters like accuracy, precision, recall, and F1 score in detecting the botnet traffic as compared to the previous single classification algorithm. However, the inquisition model can be implemented for botnet forensics in the future. Machine learning technique can be used in analyzing big data of botnet attacks with the combinations of ensemble classifier.

Data Availability

The source code of the author's framework along with the datasets and analysis during the current study is already publically available on GitHub (https://github.com/ISCX). NetMate software and Orange software have been used for the preprocessing purpose during the author's research experiment.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] A. Bijalwan, V. K. Solanki, and E. S. Pilli, "Botnet forensic: issues, challenges and good practices," *Network Protocols and Algorithms*, vol. 10, no. 2, 2018.

[2] S. Kondo and N. Sato, "Botnet traffic detection techniques by C & C session classification using SVM," in *Proceedings of the International Workshop on Security*, Vienna, Austria, September 2007.

[3] R. M. Alguliev, R. M. Aliguliyev, and S. A. Nazirova, "Classification of textual e-mail spam using data mining techniques," *Applied Computational Intelligence and Soft Computing*, vol. 2011, Article ID 416308, 8 pages, 2011.

[4] S. Garg, A. K. Singh, A. K. Sarje, and S. K. Peddoju, "Behaviour analysis of machine learning algorithms for detecting P2P botnets," in *Proceedings of the 15th international conference on Advanced computing technologies (ICACT)*, Rajampet, India, September 2013.

[5] K. Singh, S. C. Guntuku, A. Thakur, and C. Hota, "Big data analytics framework for peer-to-peer botnet detection using random forests," *Information Sciences*, vol. 278, pp. 488–497, 2014.

[6] D. Zhao, I. Traore, B. Sayed et al., "Botnet detection based on traffic behavior analysis and flow intervals," *Computers & Security*, vol. 39, pp. 2–16, 2013.

[7] P. Narang, V. Khurana, and C. Hota, "Machine-learning approaches for P2P botnet detection using signal-processing techniques," in *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, pp. 338–341, Mumbai, India, May 2014.

[8] P. Barthakur, M. Dahal, and M. K. Ghose, "A framework for P2P botnet detection using SVM," in *Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, p. 195_0, Sanya, China, October 2012.

[9] S. Bhatia, D. Schmidt, and G. Mohay, "Ensemble-based ddos detection and mitigation model," in *Proceedings of the Fifth International Conference on Security of Information Networks*, pp. 79–86, Jaipur, India, October 2012.

[10] M. M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham, "A practical labeled approach to classify evolving data streams: training with limited amount of data," in *Proceedings of the Eighth IEEE International Conference on Data Mining ICDM'08*, pp. 929–934, Pisa, Italy, December 2008.

[11] A. Bijalwan, N. Chand, E. S. Pilli, and C. Rama Krishna, "Botnet analysis using ensemble classifier," *Perspectives in Science*, vol. 8, pp. 502–504, 2016.

[12] M. M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham, "Mining concept-drifting data stream to detect peer to peer botnet traffic," Tech. Report UTDCS-05-08, University of Texas at Dallas, Richardson, Texas, 2008.

[13] X. Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 539–550, 2009.

[14] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 42, no. 4, pp. 463–484, 2012.

[15] R. McKay, B. Pendleton, J. Britt, and B. Nakhavanit, "Machine learning algorithms on botnet traffic: ensemble and simple algorithms," in *Proceedings of the 3rd International Conference on Compute and Data Analysis*, pp. 31–35, Kahului, HI, USA, 2019.

[16] M. Nazemi Gelian, H. Mashayekhi, and Y. Mashayekhi, "A self-learning stream classifier for flow-based botnet detection," *International Journal of Communication Systems*, vol. 32, pp. 1–15, 2019.

[17] A. Bijalwan, M. Thapaliyal, E. S. Pili, and R. C. Joshi, "Survey and research challenges of botnet forensics," *International Journal of Computer Applications*, vol. 75, no. 7, 2013.

[18] X. Li, L. Wang, and E. Sung, "AdaBoost with SVM-based component classifiers," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 5, pp. 785–795, 2008.

[19] W.-J. Lin and J. J. Chen, "Class-imbalanced classifiers for high-dimensional data," *Briefings in Bioinformatics*, vol. 14, no. 1, pp. 13–26, 2012.

[20] Y. Ye, L. Chen, D. Wang, T. Li, Q. Jiang, and M. Zhao, "SBMDS: an interpretable string based malware detection system using SVM ensemble with bagging," *Journal in Computer Virology*, vol. 5, no. 4, pp. 283–293, 2009.

[21] J. M. Cadenas, M. C. Garrido, R. Martinez, and P. P. Bonissone, "Extending information processing in a fuzzy random forest ensemble," *Soft Computing*, vol. 16, pp. 845–861, 2012.