Hindawi
Journal of Computer Networks and Communications
Volume 2020, Article ID 9024726, 16 pages
https://doi.org/10.1155/2020/9024726

Research Article

Hybrid Botnet Detection Based on Host and Network Analysis

Suzan Almutairi,[1] **Saoucene Mahfoudh**,[2] **Sultan Almutairi**,[3] **and Jalal S. Alowibdi**[4]

[1]*Technical and Vocational Corporation, Riyadh, Saudi Arabia*
[2]*Engineering, Computing and Informatics, Dar Al-Hekma University, Jeddah, Saudi Arabia*
[3]*Technology Control Company, Riyadh, Saudi Arabia*
[4]*Faculty of Computing and Information Technology, University of Jeddah, Jeddah, Saudi Arabia*

Correspondence should be addressed to Suzan Almutairi; salmutery@tvtc.gov.sa

Received 9 August 2019; Revised 25 October 2019; Accepted 29 November 2019; Published 22 January 2020

Academic Editor: Cong Pu

Botnet is one of the most dangerous cyber-security issues. The botnet infects unprotected machines and keeps track of the communication with the command and control server to send and receive malicious commands. The attacker uses botnet to initiate dangerous attacks such as DDoS, fishing, data stealing, and spamming. The size of the botnet is usually very large, and millions of infected hosts may belong to it. In this paper, we addressed the problem of botnet detection based on network's flows records and activities in the host. Thus, we propose a general technique capable of detecting new botnets in early phase. Our technique is implemented in both sides: host side and network side. The botnet communication traffic we are interested in includes HTTP, P2P, IRC, and DNS using IP fluxing. HANABot algorithm is proposed to preprocess and extract features to distinguish the botnet behavior from the legitimate behavior. We evaluate our solution using a collection of real datasets (malicious and legitimate). Our experiment shows a high level of accuracy and a low false positive rate. Furthermore, a comparison between some existing approaches was given, focusing on specific features and performance. The proposed technique outperforms some of the presented approaches in terms of accurately detecting botnet flow records within Netflow traces.

1. Introduction

Most of the people depend on the Internet for their day-to-day tasks such as business, education, and entertainment [1]. Botnet is one of the most popular security threats [2]. The term "botnet" is formed from two words: robot and network. Robot means an automatic program that is run and performs the intended tasks without user interaction. The bot can be good or bad. The bot in the botnet is a bad (malicious) software that is run on the victim's machine without his knowledge. The attacker who owns the bot takes control of the machine and formed, with other bots, a network of infected machines. The attacker is known as botnet master. The botnet master uses special mechanism to communicate with these bots and exchange commands through the botnet. This mechanism called Command and Control server (C&C).

Bots act in a coordinated manner and follow the botnet masters instructions, given the power of botnets to execute various malicious activities and massive attacks against e-commerce website, governments' websites, etc. [3].

The bot has the capabilities to lunch dangerous attacks such as Distributed Denial of Service (DDoS), fishing, data stealing, click fraud, and spamming. A survey from Kaspersky shows that DDoS botnet attacked 79 countries online resources in the first quarter of 2018 [4]. This number of attacks is increasing each year. The goal of botnet activities is typically to gain a financial benefit for different purposes. One of the botnet activities is to steal sensitive information from the compromised machines and send this information back to the C&C server. Then, this sensitive information is sold by the botnet master. An example of sensitive information is personal bank details or any other information

that an unauthorized person should not have the right to access to it.

Organizations firewalls are designed to allow legitimate traffic such as DNS, HTTP, and P2P. The botnet master takes this as an advantage to pass the organization firewall and download the bot into the users' computers. The huge amount of packets and data exchanged in the network prevent network administrator to detect this intrusion since he cannot monitor all these information. There are many types of botnets that are identified by the previous researchers [5–7]. The first developed botnets are Internet Relay Chat (IRC) botnets, followed by Hypertext Transfer Protocol (HTTP) botnets, and finally Peer to Peer (P2P) botnets. IRC bot is the easiest type of botnets to detect because it uses centralized architecture which means all bots are supervised from a center point. The existence of a central point makes it visible and therefore easy to be detected and blocked [8]. However, P2P bot is the most difficult one to detect because it uses distributed architecture which means the botnet master transfers command to an infected bot peer who transfers it to other peers. In this case, a single failure in distributed botnet does not create significant disruption [7].

The first action that must be taken to secure a network is to detect machines infected by bots. However, the botnet is employing and developing hiding techniques along with the development of the technology. These hiding techniques can be done by various ways such as trying to imitate the legitimate network traffic and using process injection. Fortunately, botnet follows a unique communication pattern that is different from nonmalicious (legitimate) traffic in different criteria such as packet size, flow size, flow and duration time.

Botnet detection may be implemented at the network level or at the host level. Botnet detection at the network level plays a critical role in security by monitoring the network traffic and providing warning to the network administrator when any unusual event is detected. On the other hand, the detection at the host level plays a crucial role in the detection of malware infection by monitoring files system modification, registry modification, and network traffic on the host.

Given the discussion above, our research question is as follows: Can botnets be detected effectively by combining both network traffic flow data analyzer and host processes data analyzer?

In this research, we propose a general hybrid technique capable of detecting botnets in early phase. Early phase means that we try to detect malicious behavior when the bot tries to propagate the bot malicious code to infect other machines or from the first packets exchanged between the bot and the C&C server. Our technique is deployed at the host level and the network level. The botnet communication traffic that we are interested in includes HTTP, P2P, IRC, and DNS using IP fluxing. Our technique consists of three components: network analyzer, host analyzer, and detection report:

(i) The host analyzer observes the process operation in file system and registry.

(ii) The network analyzer observes the network traffic activity of the host process. These activities include botnet propagation operation and botnet communication with command and control server.

(iii) The detection report component generates a report with infected machines' IP addresses.

To achieve our goal, we develop an algorithm called HANABot (Host And Network Analyzer for Botnet detection) to preprocess the data to detect bots based on set of features. These features have been combined to generate a classification technique capable of differentiating botnet and legitimate flow records with a high degree of accuracy. Although some of these features were previously studied, they are reused and ordered in a different way to improve the accuracy in differentiating legitimate traffic from botnet traffic. In fact, these features are fundamental to achieve high botnet detection accuracy. Moreover, new features are proposed to enhance the botnet detection especially in early stage.

Our proposed approach has the following three contributions:

(1) A general botnet detection technique capable of detecting the three types of botnet (IRC, HTTP, and P2P). For the network traffic analysis, our solution monitors two activities: botnet propagation techniques and network data flow. Botnet propagation detector aims to detect newly infected hosts before this bot starts the communication with the C&C server.

(2) Our technique, for the host process analysis, monitors file system creation and registry modification.

(3) An effective algorithm, HANABot for Host and Network Analyzer for Botnet detection, is developed.

This paper is organized as follows. Section 2 gives an overview of some research works proposed for botnet detection. Section 3 describes our methodology to reach our objectives and the laboratory experimental setting. Section 4 introduces the proposed technique, HANABot, and its components. Section 5 presents HANABot performance and a comparison with some previous solutions. Finally, we conclude our paper in Section 6 and recommend some future works.

2. Literature Review

One of the hot research topics is the botnet detection. Many researches in the literature focus on botnet detection techniques. Many of these techniques are focused only on specific botnet type. Botnet behavior-based detection approaches can be classified into three classes: host-based detection, network-based detection, and hybrid detection. Here, we briefly describe some of the previous researches that propose different solutions to detect botnet.

2.1. Botnet Detection at the Host Side. Using antivirus software and a firewall to protect the host are not enough to prevent its infection by botnet malware. Moreover, even if we can stop the C&C server, the infected host (with bot) can

be launched again into future attack. In this case, we need a host-based detection to eliminate the bot program from the host. There are various researches in host-based detection techniques [9–11]. Huang [10] proposes an effective bot host detection solution based on network failure tracking in a host during short period. The solution consists of two phases: (1) training phase and (2) detection phase. The first phase is used to extract features from failure flows. The second phase analyses the data based on knowledge obtained during training phase using C4.5 algorithm. In [9], Etemad and Vahdani deal with the detection of C&C centralized botnet by using host analysis. Their solution is based on incoming and outgoing host real traffic analyses. It consists of two basic components:

(i) Protocol classifier: the host outgoing traffic and incoming traffic are redirected to this component. First, this protocol classifier separates IRC traffic and HTTP protocol traffic from the rest of host traffic. Then, it forwards the separated traffic to the communication pattern interpreter component.

(ii) Communication traffic pattern interpreter consists of two modules: IRC module and HTTP module. This component distinguishes malicious traffic from legitimate traffic. Then, the host firewall can filter this malicious packet out. In IRC module, it detects the IRC malicious bot based on its communication traffic with the botnet C&C server. In the HTTP module, it recognizes HTTP-based botnet C&C server communication traffic based on periodic pattern of HTTP messages.

This approach is based on the traffic analyzer in the host and not on process detection. It does not process encrypted packets (command) from the botnet master to bots. Moreover, it is limited to centralized botnet and does not deal with P2P botnet.

In [12], Zeng et al. proposed a per-process level containment technique for each host in the monitored network. The per-process containment consists of two components: behavior analysis component and containment model. The behavior analysis component consists of various monitor and suspicion-process level generators. The process is monitored in the runtime behavior at the operating system level in registry, file system, and network stack. Then based on the process activities analysis, they apply the SVM algorithm to each running process to assign a suspicious level for that process. In the containment, there is a mapping function optimizer to transfer the suspicious score to a threshold for each process.

Their approach has many advantages:

(1) They propose a technique that integrated both behavior analysis and containment for early and an automatic defense against malicious network worms.

(2) They generate a suspicious level for each process using a machine learning classification technique and develop an algorithm to map each suspicious level score to a threshold for rate-limiting procedure.

(3) They perform in-depth analysis and experiment using traces of real world worm binaries and normal programs.

2.2. Network-Based Botnet Detection. Earlier, botnet detection techniques are based on payload inspection analysis techniques which check the TCP and UDP packets contents for malware signature. Payload analysis techniques are resource consuming that require processing large amount of packet data and it is a slow process. Moreover, new botnet generation employs encryption algorithms and other methods to hide their communication traffic and to crush packet payload inspection analysis techniques.

Flow-based attributes extracted from the network traces were similar to NetFlow features such as bytes-per-packet, bytes-per-flow, and bytes-per-second. In the last few years, botnet detection techniques using flow analysis has emerged. There are many network traffic flow detection techniques that have been proposed in the last years. In this section, the discussion of some of those techniques and their limitations will be stated. Anomaly detection can be through mining-based detection techniques which are used to extract unexpected network traffic patterns. Hence, it can detect abnormal traffic even if the packets are encrypted. Many techniques used flow analysis [5, 7, 13–15].

In [5], general botnet detection that is able to detect different types of botnet is proposed. This approach analyzes the network traffic flow during constant time intervals. Then, statistical correlation in the network traffic flow is performed for building effective classification system. This approach can detect botnet regardless of the topology or the protocol used. Moreover, it is capable of detecting unknown botnet.

In [7], Liao and Chang propose a P2P botnet detection approach based on analyzing and monitoring network traffic using data mining scheme. They evaluate their solution using three data mining popular algorithms: J48, naïve Bayes, and Bayesian networks. The accuracy rate of these algorithms is 98%, 89%, and 87%, respectively.

Zhao et al. in [13] propose a new solution for *P2P* to detect botnets by analyzing network traffic. Their idea consists in selecting 12 features from the network flow to be analyzed and hence extract flow behavior for predefined time window. They use machine learning algorithm to isolate the botnet traffic from the legitimate traffic. They select decision tree using the reduced error pruning algorithm. Then, they use correlation attributes evaluator to choose the key discriminating attribute for botnet detection. Their approach can detect single bot activity offline or in live traffic. Moreover, it is able to detect unknown bots and detect bots in early stage through its activity in C&C phase.

Hung and Sun in [14] propose a botnet detection system based on machine learning. The solution is based on network traffic. First, a set of flow-based features is selected and extracted from the network traffic. Then, to make sure the model performs well, some noise is added to payload, interarrival time, and features. This solution achieves high accuracy that reaches 99.7% for some botnet types.

Compared with an existing botnet detection system, the proposed solution shows that it can resist more noise.

Alauthaman et al. [15], propose a P2P Botnet detection scheme based on decision tree and adaptive multilayer neural networks. This framework passively monitors the network traffic to detect the botnet communication with the C&C server and between bots. First, network traffic reduction is done to manage the enormous amount of network traffic. Then, 29 features are proposed for botnet detection. Features with small influence on classification model are eliminated by using the feature reduction approach: the classification and regression tree (CART). The target of this feature reduction is to keep only worthy features to obtain better rates of neural network learning and classification accuracy. The evaluation of this approach shows that it gives very good accuracy with 99.2% and outperforms some existing solutions.

DNS detection techniques are based on specific DNS features analysis produced by a botnet. However, these DNS techniques are not useful to differentiate C&C server traffic and new botnet types. Some solutions focus on DNS traffic to detect botnet which relays on DNS to discover the C&C server.

Zhao et al. [16] propose an IDns system to detect malicious domain names in the C&C server for APT (advanced persistent threat) attack. The proposed system is deployed at the network side which can reduce the network traffic volume that has to be registered then analyzed. Their approach consists of four components:

(i) Data collector: to store the inbound and outbound network traffic.

(ii) Malicious DNS detector: to analyze the DNS record that is stored by data collector. Then, it detects the suspicious APT C&C domain and supplies the suspicious server IP address corresponding to the domain to the next components.

(iii) Network traffic analyzer component: it consists of two units which are signature detector and anomaly detector.

(iv) Reputation engine: it calculates a reputation result for each IP address from the previous components.

The 14 features that are studied for malicious DNS detector in this approach are divided into 5 categories: domain name features, DNS answer features, time value features, TTL value-based features, and active probing features. This botnet detection can detect only malwares that depend on DNS. Hence, P2P botnet cannot be detected.

2.3. Hybrid-Based Detection. Hybrid-based detection is a correlation between network traffic analysis and host traffic analysis.

Zeng et al. [17] propose a botnet detection technique that incorporates both host level detection and network level detection. Their solution is the first that combines the host and network level detections and correlating the alerts which could increase the detection accuracy. They propose to use 9 features for host analysis: 6 features for file and registry operations and 3 features for network traffic in the host. For network analysis, 17 features are extracted from netflow data. They evaluate the combined host and network detection technique and show that their solution is effective against IRC, P2P, and HTTP botnets.

In [18], a botnet detection technique that incorporates both host level detection and network level detection is developed. The proposed solution, called EFFORT, is implemented as a multimodule approach to correlate information from different host and network level aspects to achieve an efficient and effective detection. The solution is implemented and evaluated in real-word machines. Results show that EFFORT effectively detects until 15 real-word bots with low false positive rate.

In [19], Abdullah et al. deal with the P2P botnet architecture. Their solution depends on the combination between the host- and network-based analyses. In the host level, the analysis will be on the file system: registry and log file. This analysis looks for abnormal behavior and characteristics in every single activity that happened in the host. In the network level, the analysis is on the full packet payload. This analysis can differentiate both the C&C servers and the infected hosts with bots in the network.

This solution can be also considered as prevention mechanism. In fact, it is effective in detecting bot in early stage from the host analysis. However, it takes time to complete the analysis due to the network violations in P2P bots. For accurate detection rate, it combines the host analysis result with network analysis result.

2.4. Discussion. We have presented a brief state of the art related to botnet detection, significantly contributing to localize and detect bots in the network.

We distinguished between (1) botnet detection at the host side, (2) botnet detection at the network side, and (3) hybrid botnet detection. It is clear that the highest benefits will be obtained by a solution that

(i) Combines the host and network analysis which means hybrid botnet detection: to be efficient, some solutions integrate more than one strategy. Using hybrid botnet detection, many parameters are combined and correlated to localize and detect abnormal behavior. In this case, if the bot traffic is not detected by the network analyzer, the host analyzer can detect it using other rules and taking into account other features and the reverse is true (if the bot behavior is not detected by the host analyzer, the network analyzer can detect it using other rules and taking into account other features).

(ii) Minimizes the complexity: the target of a botnet detector is to detect botnet traffic as early as possible while reducing the complexity and the processing time of the analyzer. For example, the network analyzer must analyze and classify all the ingoing and outgoing traffic. Consequently, an efficient

botnet detector should filter the traffic before starting the analysis.

(iii) Easily adapted to detect new botnets and scalable: a botnet detector may have a good detection rate for known botnets. However, the solution should also detect unknown botnet with a very high accuracy.

In all cases, the target of all these solutions is to detect as soon as possible the botnet and stop it. The ideal solution is to obtain 100% of detection rate with 0% of false positive rate. To be adopted, such solutions should be effective with low complexity and capable to detect new or unknown botnets.

3. Methodology

In this section, we present the research methodology to be able to perform an investigation as well as analyzing the bot in the infected host.

3.1. Laboratory Environment. The laboratory environment in this research consists of physical machines and virtual machines (VMs). One of the physical machines is used to collect malwares, and the other physical machines are used to analyze malwares. Using the VM software, in this research, affords an efficient and safe environment to analyze the botnets and makes a flexible way to deploy a laboratory to analyze botnets.

This laboratory consists of several computers. If this laboratory depends only on physical computers to analysis botnets, the cost of the research will be very high. However, the advantages of using the VM are first to reduce the cost and second to be able to restore the computer to its original state if the VM is infected. This will save the time in repeating the experiment multiple times which help us to provide results with high accuracy and in safe way.

On the other hand, this research is dealing with a malicious activity that can threat the security of the researcher's network. Therefore, we need first to consider the security of the researcher's network. The solution is to use a demilitarized zone (DMZ) on the physical machine that collects malwares. This DMZ provides a honeypot in an isolated environment to capture all the malicious traffic in experimental machine.

3.2. Laboratory Components. This subsection presents the laboratory components used for the experiment of this research. These components are shown in Figure 1.

It consists of a physical computer (the host) with Linux as an operating system. On this host, we install a VM with Windows 7 as an operating system. The VM will be the target of the attack in this research, and the Linux operating system is the controller on which honeypot is installed. For a dynamic analysis, different tools will be used to carry out the experiment:

(i) Honeypot: this tool is used to collect malware samples from the Internet. In fact, the collection of the malwares using this tool is the safest way. The physical computer is connected to the Internet directly to expose the computer vulnerabilities and hence attracts attackers. The honeypot chosen for this research is the low-interaction honeypot Dionaea [20].

(ii) Controller: the controller is the Linux-based operating system. It will be used to monitor the network activity and to control the windows virtual machine activities. It will construct a database to collect malware signatures.

(iii) Virtual target: it is a windows 7 VM. The malware analysis tools will be installed on it. This will allow us to examine the bot behavior.

(iv) Dynamic analysis: this is a VM with Windows 7 as an operating system; we will have an ability to perform a dynamic analysis. A dynamic analysis will be performed in this research separately.

3.3. Dataset Collection. We have run the honeypot for one month and collected six types of malware binaries. Unfortunately, only two of the collected binaries belong to botnet. Hence, we download the rest of wanted malware binaries from VirusTotal. Then, we do the experiment to build our dataset by capturing the activities of normal and malicious processes. The monitored applications are eMule, Bittorrent, web services, Rbot, Zues, Neris, and Storm. The legitimate application traces were collected from clean (malware-free) VM PCs in regular use. The dataset collected is presented in Table 1.

To construct our proposed technique training dataset, we have applied the publicly available botnet dataset from multiple sources. We collect three different botnet datasets and one benign dataset from Dalhousie University [21], two botnet datasets and one normal dataset from CTU University [22], one botnet dataset from the Sourcefire Vulnerability Research Team (VRT) [23], and one dataset from ISCX which is a mixture of several accessible publicly available botnet and normal datasets [24].

Table 2 shows a detailed description of the collected datasets.

To evaluate our botnet detection technique, we have used the collected dataset from the experiment.

3.4. Research Design. The research design is divided in two phases. The first phase consists of the collection of malware signatures by using honeypot. These malware binaries will be used to infect the VMware host in order to monitor their activities.

The target of monitoring the infected host, using sniffer tool (wireshark) and malware analysis tools (process monitor and process explorer), is to be able to extract the communication pattern characteristics between the C&C server and this host. After that, we use these tools to analyze the collected malware behavior and gather behavior information artifacts that we need.

The second phase of this research consists of the development of an algorithm (using python) to extract needed

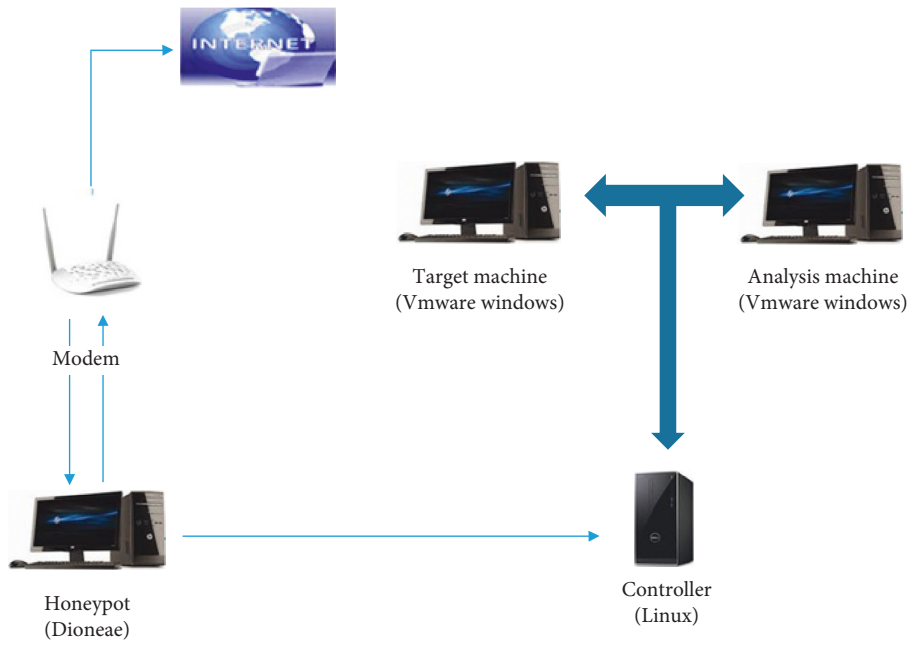

Figure 1: Laboratory components.

Table 1: Description of experiment dataset.

Dataset type	Size
Legitimate	15.4 GB
Malicious	11 GB

Table 2: Description of public datasets.

Dataset	Dataset type	Size
ISCX	Mixture	5 GB
Citadel	P2P botnet	8.39 MB
Zues	P2P botnet	6.98 MB
Rbot	IRC botnet	27 MB
Neris	IRC botnet	1.04 GB
NormalCapture	Normal	2.44 GB

artifacts. Then, we perform preprocessing steps to collect features which conduct to the final decision. This final decision will be produced using machine learning algorithm using Knime.

3.5. Extracted Proprieties for Features Building. The bots need the Internet to perform various actions such as communication with C&C server and the propagation of the botnet by infecting other computers. As a result, monitoring the network traffic and processes activities in hosts are an essential aspect to detect the botnet. The artifacts needed to our features for host processes analysis are explained in Section 4.2.

In the following, we review the features and attributes that are used for classifying P2P, IRC, HTTP, and IP fluxing activities from bots activities in network monitoring. In the traffic classification field, finding a combination of features and attributes that distinguishes each category with a high accuracy and successfully determines the label of each unknown item to a right class is the challenge.

The characteristics needed for our features vector analysis can be extracted directly from the network flow record. For each flow record, we first determine the traffic type then compute the features needed based on the traffic type.

Table 3 shows the artifacts needed to build up features for machine learning tools.

The four first artifacts are used to determine packets belonging to the same flow. The artifact numbers five and six are used to compute the failed connection ratio feature presented in Section 4.1.1. The last five artifacts are used as follows.

3.5.1. Packet Size. P2P application and Internet services that need a large amount of bandwidth will typically have large packet sizes. This is because these packets are linked with a large payloads content and contain a lot of data and information. These large packet sizes are generated by network bandwidth consuming services such as uploading or downloading files, video streaming, and P2P file sharing. The (TCP or UDP) packets, Maximum Transmission Unit (MTU), size running on the Ethernet is approximately 1400 bytes and for P2P application is between 1000 and 1400 as shown in Figure 2.

However, the botnet uses small packet size to minimize their observable impact on the network traffic and does not consume network bandwidth. There are two reasons for that: first, to maintain the connection undetectable and hidden in the network; second, to maximize the reliability of the connection between the bots and command and control server. The packet size used in the botnet communication is

Table 3: List of selected artifacts for network monitor.

Number	Artifact
1	Port source and destination
2	IP source and destination
3	Protocol (UDP or TCP)
4	HTTP method (POST or GET)
5	Total number of connections
6	Number of failed connections
7	First packet length
8	Packet size
9	Total number of packets
10	Number of input small packets
11	Number of output small packets

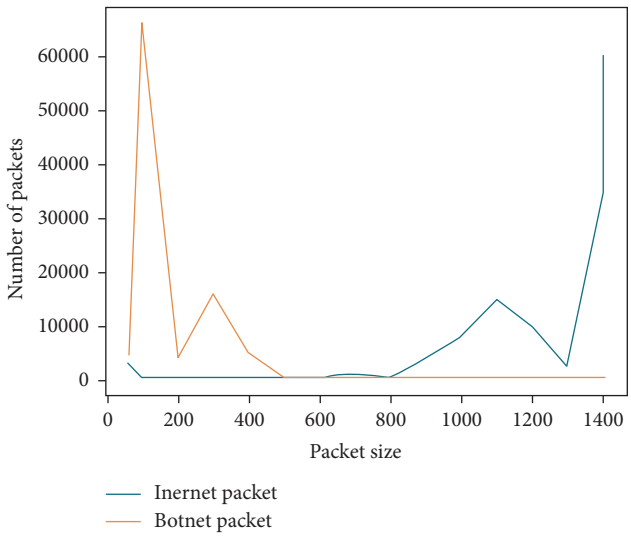

Figure 2: Packet distribution in general Internet.

less than 400 bytes as shown in Figure 2. We consider the packet size as small packet size if its size is less than 400 bytes as a measurement for botnet detection.

For HTTP traffic, 3 ranges can be identified for the request and response size which are 0–500, 501–1000, and 1001–1500 bytes. In HTTP traffic, we distinguish two cases:

(i) The abnormal behavior in which HTTP request packet size is in range1 (0–500) and the response packet size is in the same range (range1).

(ii) The normal behavior in which HTTP request packet size is in range1 (0–500) and the response packet size is in the different range (range2 or range3).

Example of HTTP request and response ranges is presented in Table 4.

3.5.2. Equal small Packet size. As mentioned in subsection 3.5.1, bots use small packet size. Furthermore, during the life-cycle of the botnet, the communication activities between the C&C server and bots follow constant pattern (since they are hard coded in the bot program). In fact, the packets exchanged between a C&C server and a bot have the same content. This means the same packet size which is

Table 4: Example of HTTP request and response ranges sizes.

Classification	RequestPacketSize	ResponsePacketSize
Abnormal	<=500 bytes	<=500 bytes
Normal	<=500 bytes	>500 bytes

generally small as explained above. Thus, the bot follows uniformity in their communication traffic behavior such as packet size, whereas legitimate users use distinct packet sizes which are generally large packet sizes. The packet size and the number of equal small packet size are used to make decision on the traffic if it is normal or abnormal behavior. Therefore, for each flow the first incoming packet size (IPS) if it is small packet or the first outgoing packet size (OPS) if it is small packet is taken as reference. These values are used at the end of the flow to conclude. Then,

(1) The number of equal incoming small packet (EIS) is computed. To achieve that, the number of all small incoming packet that their size is equal to IPS is counted. Then, the total number of packets in the flow called total incoming packet (TIP) is counted. At the end of the flow, the following equation is computed:

$$\mathrm{EIS} = \sigma\mathrm{TIP} - \mathrm{NIP}, \tag{1}$$

where σ is a positive parameter used since all packets are not equal to the first packet (IPS) in botnet flow. From our experiment, we notice that there are some distinct values. Therefore, we choose $\sigma = 0.75$.

(2) The same step is used in the case of outgoing traffic. The number of equal outgoing small packet (EOS) is computed. Hence, the number of small outgoing packets (NOPs) where the size of this packet is equal to OPS is counted. The total outgoing packets is also counted. Then, the following equation is computed:

$$\mathrm{EOS} = \sigma\mathrm{TOP} - \mathrm{NOP}, \tag{2}$$

where $\sigma = 0.75$.

4. HANABot Technique Description and Analysis

Our solution, HANABot (Host And Network Analysis for Botnet detection), works at the network level and host level. In the network level, it can monitor the traffic of all devices connected to this network and analyze it to observe specific connection patterns. In the host level, our solution can monitor the host processes operations and network traffic to detect the bot process. The monitored host process operations involve registry and file system. Then, specific features are varied to decide if a flow including a host's IP address or a process in a host is suspected of bot activity or not. Figure 3 shows our technique architecture in which we distinguish three components: network analyzer component, host analyzer component, and detection report.

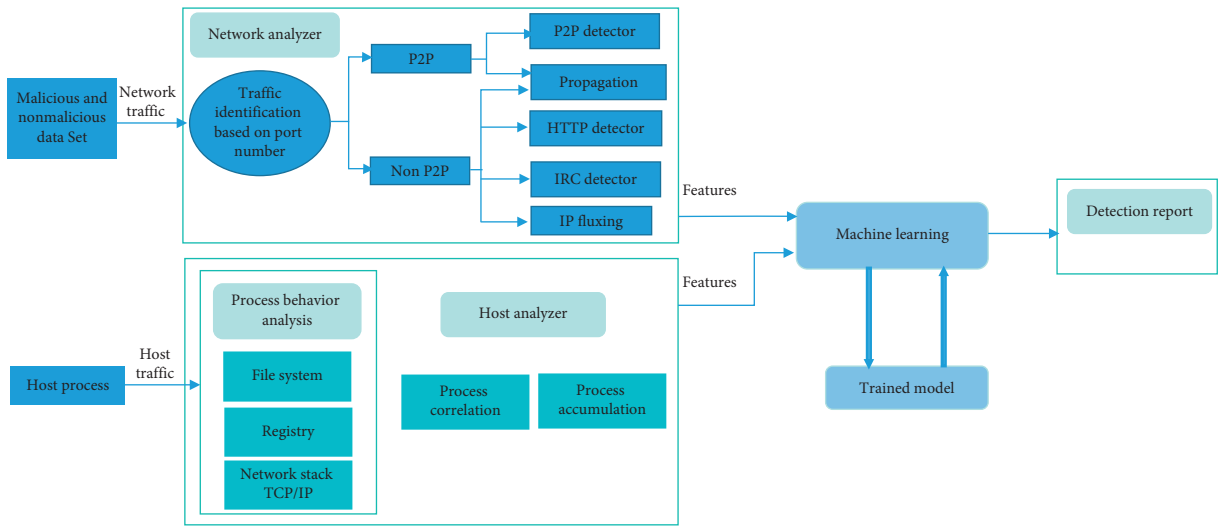

Figure 3: HANABot architecture.

In this research, we try to detect the botnet as efficiently as possible. In Section 4.1, we detailed each component of our architecture.

4.1. Network Analyzer. The network traffic of the botnet is classified into two main categories:

(i) Propagation and infection of other computers in the network either through scanning for other vulnerabilities in the network or through social engineering. Scanning techniques may be a port scanning or failed connection. Social engineering techniques may be an e-mail spam.

(ii) Botnet master communicates with the botnet members through the C&C server in the same botnet with a specific communication pattern. The communication traffic of HTTP, IRC, IP fluxing, and P2P is discussed.

As a starting step for the classification technique, we need to distinguish between P2P activities and nonP2P activities. The packets are parsed from the Wireshark packet analyzer (PCAP file) to our HANABot algorithm. The process for distinguishing the network traffic is based on the source port number, destination port number, and the protocol used. In non-P2P traffic, we identify the following network traffic:

(i) HTTP using TCP protocol and port number 80

(ii) IRC using TCP protocol and port number 194

(iii) IP fluxing by DNS using UDP protocol and port number 53

If the port source number and port destination number is bigger than 1024, then the traffic is considered P2P traffic.

In this stage, we filter out all irrelevant network traffic flows. Notice that this filtering step is just an optional procedure and not critical to our network analyzer. The purpose of this step is to reduce the computational cost by reducing the total number of flows. Then, the network traffic is split into different categories (HTTP traffic, IRC traffic, etc). In each category, all packets that belong to the same flows are extracted. Next, our HANABot algorithm extracts all the artifacts presented in Table 3 from these flows. After that, the HANABot algorithm preprocesses these extracted artifacts to obtain the relevant features of each flow. The preprocessing is applied on the network traffic to collect valuable information that based on the decision is made at the end. In the following, we present HANABot preprocessing steps.

Figure 4 shows the network analyzer components. In the following, we present each component in more details.

4.1.1. Botnet Propagation Detector. This component is common for P2P and non-P2P botnet. In fact, in all case, the botnet uses propagation techniques to spread its malicious code and infect new machines. The bots will be looking for known vulnerabilities in new victim machine as part of the propagation process. Bots look for a new target randomly or a specific machine and scan it. The propagation techniques that we interested in are as follows:

(i) In traditional non-P2P traffic, such as HTTP, DNS, and SMTP, the server works well and generally all connections are successful. Therefore, the failed connection is low in client/server networks. However, in P2P networks, such as BitTorrent, the failed connection is high. The reason for that is that a peer maintains the connection with other peers and tries to connect to the peers that have been disconnected in P2P networks. However, IRC botnet and HTTP botnet have high failed connection ratio since the bot is trying to find the C&C server IP address. Regarding P2P botnets, it has low failed connection ratio because the bot communicate with known peers via hard coded IP addresses.

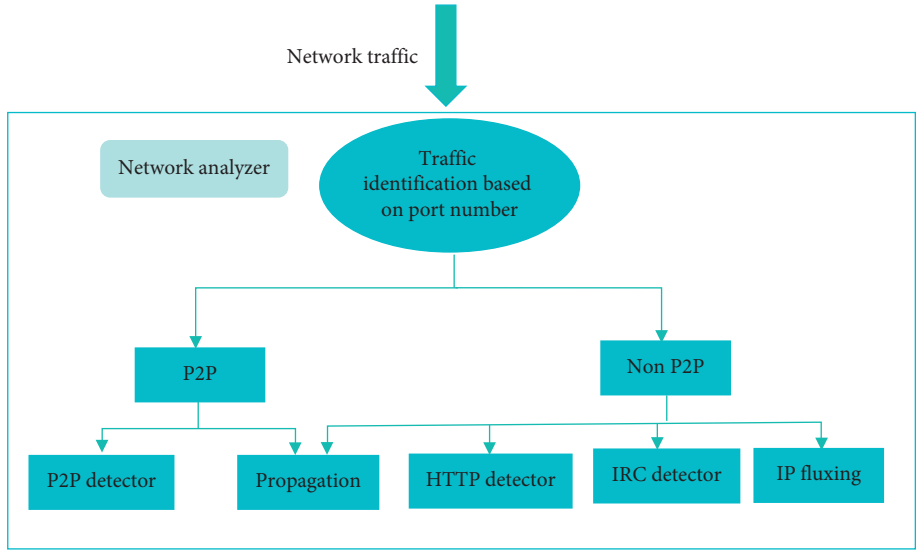

Figure 4: Network analyzer components.

The failed connection types are listed in Table 5. We have a failed connection if the outgoing connection is not successful.

The failed connection ratio is computed using the following equation:

$$\text{FCR} = \frac{\text{FCN}}{\text{ACN}}, \tag{3}$$

where FCR means failed connection ratio, FCN means failed connections number, and ACN means all connections number.

(ii) Port scanning can be detected by monitoring the failed connection from the same source IP to different hosts in the network. If the packet is non-P2P packet (if the source port or the destination port is <1024), many failed connections to different hosts in sliding time window is considered as abnormal behavior.

Based on Table 5, we keep track of the number of failed connections issued from the same sender IP address. Every time a failed connection is tracked, a variable flag *is_failed* is set to 1. Then after 2 seconds, we sum up the *is_failed* value for each sender in the monitored network.

(iii) E-mail spam is a method for sending unwanted e-mail messages frequently with commercial or marketing content in large quantities to a random set of recipients. Delivering e-mail spam procedure is same as delivering legitimate e-mail, utilizing Simple Mail Transfer Protocol (SMTP) on port 25 [25].

The botnet master uses botnet for diffusing spam e-mails at a large scale. The senders of e-mails may belong to different botnet networks and also they may serve different botnet masters for different purposes and this spam emails can have different contents.

We keep track of the number of SMTP packets issued from the sender IP address. Every time a SMTP packet is tracked, a variable flag *is_SMTP* is set to 1. Then after 120 seconds passes, we sum up the *is_SMTP* value for each sender in the monitored network.

Table 5: Monitored network failure types.

Protocol	Description of the failure	
	Packet sent	Packet received
TCP	TCP SYN	TCP reset
	TCP SYN	ICMP unreachable
	TCP SYN	No packet received for 120 seconds
UDP	UDP	ICMP unreachable
	UDP	No packet received for 120 seconds
DNS	DNS query	A DNS server error code to the queried domain

Features for botnet propagation detection:

(1) Failed connection ratio (FCR)

(2) Number of failed connection from same IP address during an interval of time (2 seconds in our case)

(3) Number of emails (port 25) from same IP address during an interval of time (120 seconds in our case)

4.1.2. P2P Communication Traffic Detector. In [26], we present our work to detect P2P botnet. Here, we mention briefly the presented approach. If both the source port number and the destination port number of the packet is greater than 1024, then the packet is considered as P2P packet. In P2P application, each node can act as server to download data or as client to upload data. In order to

communicate with each other, each peer should have a peer list. It can get the peer list from distributed hash table (DHT) using the UDP transmission protocol or from the application server using the TCP transmission protocol. After obtaining the peer list, the peer, first, tries to establish the communication with other peers. This is the first stage in a P2P application session. The second stage consists of data exchange stage.

(1) Communication establishment stage: the transport protocol UDP or TCP can be used to establish communication with existing peer list.

UDP communication establishment: the peer sends a small UDP packet to establish the P2P session and then waits for a packet from the receiver to start the transmission of the data. Figure 5 presents the UDP packet sizes of the most common P2P application [27].

According to Figure 5, we distinguish two cases:

(i) If the packet request size is between 36 bytes and 67 bytes, then it is considered as an abnormal behavior. Therefore, if the size of UDP packet used to initiate a P2P session is between 36 bytes and 67, it is a botnet communication packet. In fact, this case is when the bot knows who is the peer bot to communicate with via hard coded IP addresses. Moreover, the bots send many small packets to maintain connections.

(ii) If the size of UDP communication request packet is more than 120 bytes and the response packet size exceeds 400 bytes, then the traffic is considered as suspicious bot. In fact, if the peer needs to retrieve other peers IP addresses who has the wanted file using DHT, it should send a request UDP packet size as mentioned above and receives packet with a size less than 400 bytes as in case of BitTorrent. However, the response, in case of botnet, contains the malicious code which is generally larger than 400 bytes.

TCP communication establishment: a normal three handshake TCP communication establishment is performed. Therefore, the detection for TCP communication is performed in the data transmission stage.

(2) Data transmission stage: After the establishment of the connection, the data transmission starts by uploading or downloading data through the P2P network. During this transmission, peers use large packet sizes to reduce the overhead in the network. These packet sizes are generally higher than 1000 bytes. However, bots use small packet size to minimize the impact of its traffic on the network as explained above in this paper. Equations (1) and (2) computed in the preprocessing step are used to make a decision concerning the traffic type:

(i) Based on Equation (1), if NIP >= EIS, then the number of incoming equal small packet size is high (abnormal behavior).

(ii) Based on equation (2), if NOP >= EOS, then the number of outgoing equal small packet size is high (abnormal behavior).

An example of Equation (1) computation is presented in Table 6.

The flow is terminated in TCP connection by TCP finish flag or TCP reset flag. There are other cases where the flow can be terminated in a nonstandard way such as failure in the physical link. In this case and in UDP transmission protocol, we check the time of the last packet arrived if it exceeds 120 seconds (this time is the default value used by open source linux kernel's connection tracking module), we consider this flow as terminated.

Features for P2P communication traffic detection:

(i) Packet request size for UDP protocol

(ii) Packet communication request size for UDP protocol

(iii) Packet response size for UDP protocol

(iv) NIP, EIS, NOP and EOS for TCP protocol (for each flow)

4.1.3. Non-P2P Communication Traffic Detector. Non-P2P application is any application that runs on well-known port number (if the source port number or the destination port number is less than 1024). In non-P2P detector, we monitor HTTP, IRC, and IP fluxing.

(1) HTTP Detector. HTTP traffic uses the TCP transmission protocol. The standard HTTP traffic contains request and response messages.

(i) The request can be one of three types: GET, POST, and HEAD.

(ii) Each response has a status codes. This status code can be one of five classes. The interesting response status code for us is 200 from the second class (2xx) which means that the HTTP request successful. The actual response code status number will depend on the HTTP request method type used. In a HTTP GET request, the HTTP response status code 200 will contain an information entity corresponding to the requested web resource. In a HTTP POST request, the HTTP response status code 200 will contain an information entity containing the result of the activity.

We analyze three types of parameters which are the HTTP request size, HTTP response size, and payload packet size. In normal HTTP traffic, to download data, legitimate users will send small request packet size and receive large response packet size with large incoming packet size. To upload data, legitimate users will send large request packet size with large outgoing packet size and receive small response packet size.

However, C&C server communication with bots using HTTP as shown in Figure 6 can be categorized in three cases.

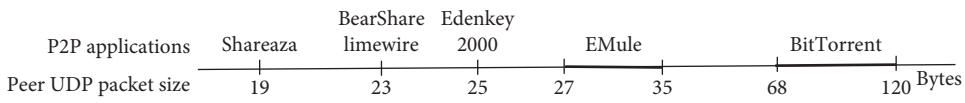

Figure 5: UDP communication packet in famous P2P application.

Table 6: Example of equal incoming small packet calculation.

TIP	NIP	EIS	Compare NIP with EIS
30	15	7.5	High
20	5	10	Low

(i) The C&C server sends to the bot instructions to do it or malicious code to download them. In this case, the incoming packets size will be small and the number of equal small incoming packets size should be large.

(ii) The bot sends the compromised host machine's information to the C&C server. In this case, the outgoing packets size will be small and the number of small outgoing packets that have the same size should be large.

(iii) The bot maintains the connection with C&C for updates. In this case, the incoming and the outgoing packet size will be small. Moreover, the request size and response size will be in same range (both are in small range).

The preprocessing of number of small packet is done by equations (1) and (2).

The flow is terminated by TCP finish flag or TCP reset flag. There are other cases where the flow can be terminated in a nonstandard way such as failure in the physical link. In this case, we consider this flow is terminated if the last packet arrived 120 seconds ago.

Features of the HTTP detector are as follows:

For each flow,

(i) HTTP request size (POST or GET messages)

(ii) HTTP response size (status code 200)

(iii) NIP, EIS, NOP, and EOS

(2) IP Fluxing. DNS protocol is used to resolve the server name to the IP address. When the host performs DNS query for a specific domain, it will retrieve DNS response records. The response record can be one of the different types such as A record for IPv4. The mapped IP address in the response record can contain one or many IP addresses if the domain name was located. This response record optionally has a time to live (TTL) period. This time period is predefined by the authority name server for the zone. Caching resolver performs lookup and stores the results for a TTL period of time. This period informs how long (in seconds) a server and application should cache this address. Most of the legitimate services use long caching times such as 86400 seconds or even more. This long caching relieves the loads on the authority servers and provides response faster [28–30].

The main challenge for the botnet master is to hold the C&C server undetected to longer time as many security administrators try to block down and track the IP addresses of these servers [30]. Moreover, to better manage large distributed infrastructure, botnet masters have implemented fast flux techniques using DNS traffic for four reasons:

(i) They intend to minimize cache time by setting a low TTL period for C&C domain names. This is because long caching delays the propagation of new IP of the C&C servers.

(ii) They obtain the flexibility to change the IP addresses of the malicious servers that they manage.

(iii) The C&C server becomes more difficult to identify and take down.

(iv) To hide their critical servers behind proxy services.

DNS, for load balancing, uses a round robin technique in two steps:

(i) The domain name is mapped to multiple IP addresses.

(ii) The DNS server rounds through these IP addresses and returns each time a different IP address mapping [31].

Whenever a computer is infected with a bot, it joins to the botnet as an asset to the collection [29]. Malicious domains IP addresses are gathered from bots that exist in different autonomous systems and countries. Botnet masters do not aim a specific country or a specific IP range [32].

We can detect IP fluxing in DNS by keeping track of two artifacts: IP address assigned to domain and domain cache TTL. Every time the host sends a query for specific domain, we keep track of the last IP assigned (if there), the new one and the domain cache TTL. In fact, if newest IP address is not in the same class subnet of the previous IP and cache TTL period is less than 1000, then this domain is suspicious. Indeed, normal domain assigns multiple IP addresses in the same class subnet and long domain cache TTL period.

Features for IP fluxing detector are as follows:

(i) IP address assigned to domain

(ii) Next IP address assigned to domain

(iii) TTL value

(3) IRC Detector. We follow the same procedure as discussed in P2P data transmission. We keep track of the number of small packet size and the ratio of the equal small packet size.

4.2. Host Analyzer. Table 7 shows our features for host process analysis.

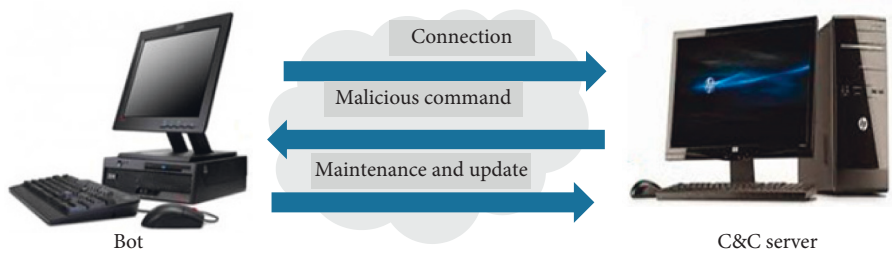

Figure 6: C&C server communication with bots.

Table 7: List of selected artifacts for host monitor.

Feature number	Behavior features
1	Creation of DLL or EXE in system directory
2	Creation of and set the value of AutoRun key in registry
3	Critical registry key modification
4	Active time of the bot process

Host analyzer, as shown in Figure 7, consists of three components: process behavior analysis, process correlation, and process behavior accumulation. In the host analyzer, we monitor three activities: registry keys, file system, and process active time, for a period of time. The monitoring time is randomly chosen from 10 min to 30 min. In Section 4.2.1, we discuss these components in more details.

4.2.1. Process Behavior Analysis. We develop an algorithm to monitor the registry keys, file system, and active process time. The first action to reduce the propagation speed of botnet is to identify processes that perform malicious activities in a host computer. The botnets share certain behavior patterns, such as creating autorun registry key and creation of EXE in system directory, that are different from normal application. However, a single activity, such as creation of an autokey in the registry or creation of EXE in system directory, alone could appear harmless, while the combination of multiple activities may expose a malicious intent.

In our analysis, we extract the common botnet behavior. We focus on software process run-time behavior vector including registry modification, file system creation, and network traffic. The host process network traffic analyzing and detection is performed and followed the procedure explained in Section 4. In this section, we explain the features that are used by our technique to analyze the monitored registry, file system, and active time of processes in the host.

(1) Registry. Bots, in the registry, deal with two keys entries operations:

(i) Bots will modify the value of critical registry key and the creation of autorun key. The means of critical registry key value is the value of any software meant to do process monitoring to detect any malicious action. An example of process monitoring software is Windows Task Manager, Sysinternals tools, and antivirus. The bot changes the value of the software registry to evade detection such as Taskmgr disable, overriding antivirus, windows firewall disable, etc. An example of value of the registry key that had been modified by a bot is HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System.

(ii) Bots will create an autorun key in the registry to automatically run themselves when the operating system starts up. The value of the autorun key will set to the name of the executable file of the bot. An example of bot creation of an autorun key is Maripose [33].

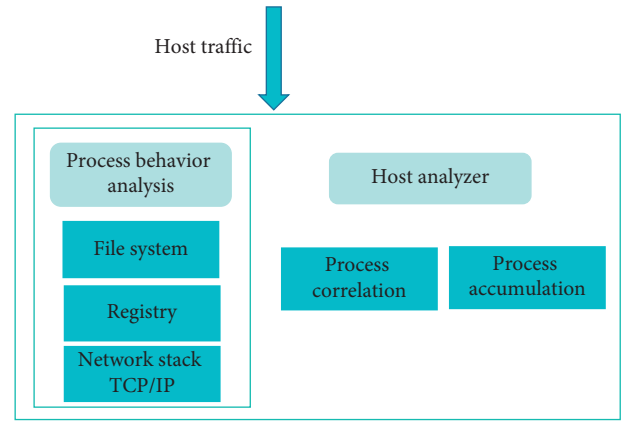

Figure 7: Host analyzer components.

We use the HANABot algorithm to periodically check the registry location mentioned above. Then, HANABot extracts the value of the registry key and finds the process ID who own this value.

(2) File System. When a victim computer is infected with a bot, a bot always downloads its malicious binary from the Internet to the file system directory. This is because it could be launched again when the operating system starts up.

Bayer et al. [34] indicate that malware files creation belongs to one of the two following categories:

(i) The first one contains executable files (EXE). Most of the bots choose system directory location such as C:\WINDOWS or one of its subfolders as a typical location to drop the binary. This is because normal users rarely check the system directory and the bot

binary is less noticeable among the system files which could be a thousands of files. New bots start to target user folder which is C:\USERS\username\-Documents and Settings.

(ii) The second one contains nonexecutable files which are dynamic link library (DLL), temporary data files, and batch scripts. Most of these files are placed either in the windows directory or in the user folder. Internet traffic creates significant amount of temporary Internet files. These files are used to download content from the Internet.

Based on the monitored location, we use the HANABot algorithm to monitor the creation of DLL and EXE files access in windows directory. Then, HANABot extracts the process ID owner of this file.

(3) Active Time of the Bots. Bots need to be always online to maintain the connection with the botnet master. If the bots are disconnected, then the botnet life cycle will be affected. As a result, the active time of the bots (operating in the background and waiting for updates) should be similar to the underlying compromised operating system. In contrast, the active time of any legitimate process is determined by the user, which is likely to be transient from one user to another.

The HANABot algorithm is used to monitor the active time of the process. It takes the creation time of the process. If its time is closer to OS starts up time, then it checks periodically to see if this process is terminated or not.

4.2.2. Process Correlation. To perform malicious activities, bots can act as follows:

(i) The bot behaves on its own process to conduct its malicious activities.

(ii) The bot divides its activities through multiple processes, so the processes collaboratively perform its assigned malicious activities. To mitigate against such sophisticated bots whose one process malicious activity is not considered as a bot, we take into account process correlation during the monitoring. We keep track of the interprocess relations and aggregate the behavior vector from the processes correlation. The obtained behavior vectors are equal for all the correlated processes like a parent process and its children processes. We can easily exclude some normal process with correlation such as windows services by maintaining a white list. Therefore, it is not hard to differentiate a set of processes behaving malicious altogether. This means that these processes have exactly the same behavior vectors.

4.2.3. Process Behavior Accumulation. The bot, during the monitoring time window, may perform either multiple malicious activities or single malicious activity. To deal with this type of bot, we collect the value of each behavior feature vector in host monitor as shown in Table 8. The behavior vector features value summation (accumulation) are those rarely seen from normal software, such as creating EXE file in system directory or creating an AutoRun key in the registry, etc. The accumulation procedure is straightforward. An example of the accumulation procedure is shown in Table 8.

Using this procedure, the process final score will not decrease if the process does one malicious activity in single time window. Every window time, the process conducts malicious activity and its score will increase. The same concept is applied to a set of processes when each one shows malicious activity in various time windows.

After the bot process is identified in the monitored host, the file associated with this process will be scanned by the administrator. This file can provide valuable information about the C&C server IP address and how the communication is established between the bot (the compromised machine) and the C&C server. Moreover, the administrator can know how the botnet is propagated and infect other machines. This knowledge permits to the administrator to, in one hand, block all the incoming and outgoing traffic with the C&C server, and, in another hand, improve the overall detection mechanism by exploiting these information to build new rules.

Applying only host analyzer technique may not be effective because host resident bots may compromise the detection technique.

5. Evaluation

5.1. Classification Algorithms. As malicious bot software spreads, the demand for an automated alternative is increased because manually generated rules become impractical. Machine learning algorithms allows the detection systems to fit fast changing malware. Moreover, it allows an automated method for learning specific patterns and properties that may be presented by botnet.

Our target is to effectively distinguish between legitimate traffic and botnet traffic. To achieve that, we consider the most memorable classification algorithm, namely, decision tree ensemble classifier and Naive Bayesian statistical classifier.

Naive Bayesian classifier (NB) represents a supervised learning method and a statistical technique for classification. It assumes an implicit probabilistic model. It allows to capture and analyze the relationship between each feature and the corresponding class to derive a conditional probability that links feature values and its class.

The decision tree classifier (DT) utilizes a decision tree as a predictive model which maps an item observation to conclusions about the item's target class. It is one of the predictive modelling approaches used in data mining and machine learning. The classification tree is a tree model where the target variable can take a finite set of values. In this tree structure, leaves represent class labels and each branch represents the feature so the conjunctions of these features lead to those class labels.

Table 8: Example of behavior vector of malicious process accumulation.

Time window	Behavior feature	Feature score	Accumulated feature score
win0	EXE file creation	1	1
win1	AutoRun creation	1	2
win2	DLL file creation	0	2

5.2. Performance Metrics. To avoid data bias, multiple metrics are measured. Accuracy, precision, recall, F-measure, and false positive rate are the five metrics used to evaluate the performance of our solution. Accuracy depends on the sample size. However, precision and recall measures are independent of the sample size.

Precision rate indicates the credibility of stated detection result, whereas Recall rate indicates the portion of hosts belonging to a specific class that can be identified. The metric F-measure is a coordinated mean of Precision and Recall. The last metric which is false positive rate indicates an error in classifying an item in botnet class whereas the right class is legitimate class.

Table 9 shows the performance results for the five presented metrics. As shown in this table, the decision tree classifier achieves high performances and outperforms the naive Bayesian classifier in host level and network level detection. Its accuracy reaches 99.6% in case of hybrid detection with 0.01 as false positive rate. That is why we choose the decision tree classifier for our solution.

5.3. Results and Discussion. Our technique consists of two main components: host analyzer and network analyzer. The purpose of our technique is to detect unknown botnet and infected hosts with bots. To build the ground truth, we collect the data set from the publicly available botnet datasets. To evaluate the classifier detection technique accuracy rate, we use the 10-fold cross validation method to divide our dataset into 10 random subsets. 9 subsets are used for the training, and one subset is used for the evaluation. This same process is repeated until each subset of the 10 subsets have been used as the testing set exactly once. The evaluation must show that it can detect unknown botnet that are not in the training dataset. For this purpose, we use the data set collected from our experiment as explained in Section 3. The detection accuracy rate of our network analyzer is 99.6%. We report the following:

(i) Neris bot, which belongs to IRC botnet, generates high failed ratio and performs scanning technique. Our technique is able to detect the IRC bot in the propagation phase.

(ii) The Storm bot behavior is similar to bitTorrent in term of number of failed connections. Moreover, they both implement DHT to obtain the peer list. Therefore, Storm bot can escape to the detector during the propagation stage. However, since in bitTorrent, the DTH UDP packet size is bigger than 67 bytes while the bot uses less than this size, the P2P detector will detect this bot. The detection technique achieves very high accuracy.

Table 9: Botnet detection technique result.

Measurement	Host		Network		Both levels	
	NB	DT	NB	DT	NB	DT
Precision	0.97	0.99	0.98	0.98	0.99	0.99
Recall	0.97	0.98	0.97	0.96	0.98	0.99
F-measure	0.96	0.98	0.97	0.97	0.98	0.99
False positive rate	0.04	0.03	0.04	0.04	0.05	0.01
Accuracy (%)	95.8	98.5	98.2	99.1	98.6	99.6

Table 10: Botnet detection technique result with all host features.

Decision tree classifier	Accuracy (%)
For all selected feature	85
For reduced features	98.5

(iii) From our experiment, the failed connection ratio result for some P2P application is lower than the threshold fixed for botnet detection; eMule is an example of this case. In fact, eMule generates very small failure connection rate. This is due to the use of a "bad peer source list," which will be added to the application. This list is updated with the peers IP addresses when the eMule application fails to connect with these peers IP addresses. Consider this peer source list of IP addresses as dead and block for predefined time (15 to 45 minutes). Even if eMule application behaves similar to bot in terms of failed connection rate, eMule is not detected by our technique since we check also equal small packet size feature which is not satisfied in this case.

Table 10 shows a comparison of the two classifier performances at host level. The first classifier uses all selected features. The performance of this classifier generates high false positive rates and low accuracy rate. This is due to two features that are common between normal P2P application and botnet. These features are as follows: create and set the value of AutoRun key in the registry and Active time of the bot process.

These features are not effective to differentiate between bots and legitimate P2P application detection and increase the false positive. This is due to the nature of P2P application, and it creates an AutoRun key to launch when the OS starts up and stay working online in the background (the active time of this process is equal to the active time of the OS). Botnet acts in the same way. Therefore, we evaluate the second classifier by eliminating these two features from the behavior vector. The second classier decreases the false positive rate and increases the accuracy rate.

Table 11: Performance comparison with published approaches.

Detection methods	C&C structure	False positive rate (%)	Accuracy (%)
Zhao et al. [13]	P2P, HTTP	0.2	99.1
Kirubavathi andAnitha [5]	IRC, P2P, HTTP, hybrid	0.048	99.14
Alauthaman et al. [15]	P2P	0.75	99.2
HANABot (network analyzer)	IRC, P2P, HTTP	0.04	99.1
HANABot (host and network analyzer)	—	0.01	99.6

In our experiment, when host process is infected with Rbot, the Rbot process injects the svchost process (windows service). The svchost performs DNS query to obtain the C&C server IP address. Then, the Rbot process establishes TCP connection to this address. Rbot injects the svchost process with malicious code. The host analysis does not cover any process injection methods. Therefore, the Rbot process will evade our host analysis analyzer. However, if we use both levels host analyzer and network analyzer, this bot will be detected by the network analyzer. In fact, the detection accuracy rate is increased, if we combine the network and the host analyzer. In this case, all the flows are classified correctly in the right class (botnet or legitimate).

5.4. Comparison. We compare our technique with the research works proposed in [5, 13, 15]. The performance comparison of our solution to some published approaches is shown in Table 11. The existing solution performances are computed by each developer. Zhao et al. [13] used two P2P botnets, namely, Storm and Waledac, and two HTTP botnets, namely, BlackEnergy and Weasel, to confirm their model efficiency. Kirubavathi and Anitha [5] used three botnets, namely, Zeus, Spyeye, and BlackEnergy botnets to evaluate their solution. Alauthamanet al. [15] used two P2P botnets, namely, Storm bot and Waledac bot to validate their model.

From the performance comparison table, we can remark that we achieve very good accuracy rate with the lowest false positive rate using HANABot with the network analyzer only. However, our proposed solution, which is hybrid solution based on network and host analysis, outperforms existing solutions in terms of accuracy rate and false positive rate.

The works in [9, 12] utilize the effectiveness of host level analysis for more accurate results. These approaches, as our approach, monitored registry and file system in each host. However, we differentiate in the feature selection of the registry such as AutoRun key creation which increases the botnet detection.

6. Conclusion

In this research, we proposed a general technique that is capable of detecting a new botnet implemented on three levels: the host level, the network level, or a combination of both. The botnet communication traffic we are interested in includes HTTP, P2P, IRC, and DNS using IP fluxing. Our proposed technique consists of three components: host analyzer, network analyzer, and detection report:

(i) The network analyzer monitors two activities: botnet propagation and botnet communication with the C&C server. The network analyzer consists of three detectors: the propagation detector, the P2P detector, and the non-P2P detector.

(ii) The host analyzer monitors three activities: the registry keys, the file system, and the duration of the active process over a period of time. The host analyzer consists of three components: process behavior analysis, process correlation, and process behavior accumulation.

(iii) The detection report is responsible for producing the final score result based on the host analysis or network analysis and provides a report of infected machines.

We developed the HANABot algorithm to preprocess host traffic data, host processes operations, and network traffic activity to detect the bots based on specific rules. All botnet flow records were successfully extracted by exploiting specific connection patterns and setting up feature vectors that is unique for botnet network traffic and its process operations. A comparison between an existing approach was given, focusing on specific features and performance.

In our future work, we will apply our solution for real-time classification. In fact, the proposed technique implements the classification algorithm in an offline mode, where the Internet network traffic traces are stored in a PCAP file before processing them. The technique can be enhanced to obtain real time classification instantaneously with a high level of accuracy. This can be achieved by making relatively short the export time period of the Netflow traces collector and by supplying the exported Netflow traces to the classification technique instantly, as they arrive. The technique can then differentiate botnet activities and take action on this basis. The processing speed at which real time classification takes place should vary, depending on various factors such as the export time interval and the size of the exported Netflow traces. Moreover, this work can be extended, in the host level, to support the detection of process injection used by botnet to exert full control over the process [35].

Data Availability

The dataset used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. Sanchez, "The 10 most common security threats explained," 2017, http://blogs.cisco.com/smallbusiness/the-10-most-common-security-threats-explained.

[2] Us.norton.com. (n.d.), "Bots and botnets—a growing threat," 2017, https://us.norton.com/botnet/.

[3] B. Cusack and S. Almutairi, "Listening to botnet communication channels to protect information systems," in *Proceedings of the Australian Digital Forensics Conference*, pp. 44–52, Joondalup, Australia, December 2014.

[4] DDoS attacks in Q1, 2018, https://securelist.com/ddos-report-in-q1-2018/85373/.

[5] G. Kirubavathi and R. Anitha, "Botnet detection via mining of traffic flow characteristics," *Computers & Electrical Engineering*, vol. 50, pp. 91–101, 2016.

[6] J. He, Y. Yang, X. Wang, Y. Zeng, and C. Tang, "PeerSorter: classifying generic P2P traffic in real-time," in *Proceedings of the 2014 IEEE 17th International Conference on Computational Science and Engineering*, pp. 605–613, Chengdu, China, December 2014.

[7] W. H. Liao and C. C. Chang, "Peer to peer botnet detection using data mining scheme," in *Proceedings of the International Conference on Internet Technology and Applications*, pp. 1–4, Wuhan, China, August 2010.

[8] T. S. Hyslip and J. M. Pittman, "A survey of botnet detection techniques by command and control infrastructure," *Journal of Digital Forensics, Security and Law*, vol. 10, no. 1, Article ID 2, 2015.

[9] F. Etemad and P. Vahdani, "Real-time botnet command and control characterization at the host leve," in *Proceedings of the Sixth International Symposium on Telecommunications*, pp. 1005–1009, Tehran, Iran, November 2012.

[10] C.-Y. Huang, "Effective bot host detection based on network Failure models," *Computer Networks*, vol. 57, no. 2, pp. 514–525, 2013.

[11] M. Rostami, B. Shanmugam, and N. Idris, "Analysis and detection of P2P botnet connections based on node behaviour," in *Proceedings of the 2011 World Congress on Information and Communication Technologie*, pp. 928–933, Mumbai, India, 2011.

[12] Y. Zeng, X. Hu, H. Wang, G. Shin, and A. Bose, "Containment of network worms via per-process rate-limiting," in *Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks*, pp. 1–10, Istanbul, Turkey, September 2008.

[13] 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.

[14] C. Hung and H. Sun, "A botnet detection system based on machine-learning using flow-based features," in *Proceedings of the SECURWARE 2018: The Twelfth International Conference on Emerging Security Information*, Italy, September 2018.

[15] M. Alauthaman, N. Aslam, L. Zhang, R. Alasem, and M. A. Hossain, "A P2P Botnet detection scheme based on decision tree and adaptive multilayer neural networks," *Neural Computing and Applications*, vol. 29, no. 11, pp. 991–1004, 2018.

[16] G. Zhao, K. Xu, L. Xu, and B. Wu, "Detecting APT malware infections based on malicious DNS and traffic analysis," *IEEE Access*, vol. 3, pp. 1132–1142, 2015.

[17] Y. Zeng, X. Hu, and K. G. Shin, "Detection of botnets using combined host- and network-level information," in *Proceedings of the 2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*, pp. 291–300, Chicago, IL, USA, June 2010.

[18] S. Shin, Z. Xu, and G. Gu, "EFFORT: efficient and effective bot malware detection," in *Proceedings of the 31th Annual IEEE Conference on Computer Communications (INFOCOM'12) Mini-Conference*, pp. 71–80, Orlando, FL, USA, March 2012.

[19] R. Abdullah, M. Faizal, and Z. Noh, "Tracing the P2P botnets behaviours via hybrid analysis approach," *European Journal Scientific Research*, vol. 118, no. 1, pp. 75–85, 2014.

[20] G. Sabbatel and A. Duda, "Multimedia communications, services and security," in *Proceedings of the 7th International Conference, MCSS 2014*, Krakow, Poland, June 2014.

[21] F. Haddadi, Z. Heywood, and A. Nur, "Data confirmation for botnet traffic analysis," *Foundations and Practice of Security*, pp. 329–336, Springer International Publishing, Berlin, Germany, 2015.

[22] Malware Capture Facility Project, 2016, http://mcfp.weebly.com/mcfp-dataset.html.

[23] Analysis of the Zeus Trojan, https://labs.snort.org/papers.

[24] ISCX Botnet Dataset, 2015, http://www.unb.ca/research/iscx/dataset.

[25] E. Willa, K. Anestis, L. Danielle, and H. David, *Detection of Spam Hosts and Spam Bots Using Network Flow Traffic Modeling*, USENIX Association, Berkeley, CA, USA, 2010.

[26] S. Almutairi, S. Mahfoudh, and J. S. Alowibdi, "Peer to peer botnet detection based on network traffic analysis," in *Proceedings of the 2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–4, Larnaca, Cyprus, November 2016.

[27] Identifying P2P Users Using Traffic Analysis, 2016, http://www.symantec.com/connect/articles/identifying-p2p-users-using-traffic-analysis.

[28] B. Kang, "DNS-based botnet detection," *Encyclopedia of Cryptography and Security*, pp. 362–363, Springer, Berlin, Germany, 2011.

[29] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "EXPOSURE: finding malicious domains using passive DNS analysis," in *Proceedings of the 18th Annual Network and Distributed System Security Symposium*, San Diego, CA, USA, February 2011.

[30] S. Lee and J. Kim, "Fluxing botnet command and control channels with shortening services," *Computer Communications*, vol. 36, no. 3, pp. 320–332, 2013.

[31] S. S. C. Silva, R. M. P. Silva, R. C. G. Salles, R. M. Pinto, and R. M. Salles, "Botnets: a survey," *Computer Networks*, vol. 57, no. 2, pp. 378–403, 2013.

[32] S. Yadav and A. Reddy, "Winning with DNS failures: strategies for faster botnet detection," in *Proceedings of the 7th International ICST Conference on Security and Privacy in Communication Networks*, pp. 446–459, London, UK, September 2012.

[33] P. Sinha, A. Boukhtouta, V. Belarde, and M. Debbabi, "Insights from the analysis of the mariposa botnet," in *Proceedings of the Fifth International Conference on Risks and Security of Internet and Systems*, pp. 1–9, Montreal, QC, Canada, October 2010.

[34] U. Bayer, I. Habibi, D. Balzarotti, E. Kirda, and C. Kruege, "A view on current malware behaviors," in *Proceedings of the 2Nd USENIX Conference on Large-Scale Exploits and Emergent Threats*, Boston, MA, USA, April 2009.

[35] S. Shin, Zh. Xu, and G. Gu, "EFFORT: efficient and effective bot malware detection," in *2012 Proceedings of the IEEE INFOCOM*, pp. 2846–2850, Orlando, FL, USA, March 2012.