PeerClear: Peer-to-Peer Bot-net Detection

Amit Kumar, Nitesh Kumar, Anand Handa(✉), and Sandeep Kumar Shukla

C3I Center, Department of CSE, Indian Institute of Technology, Kanpur, Kanpur, India
{amitkr,niteshkr,ahanda,sandeeps}@cse.iitk.ac.in

Abstract. A *bot-net* is a network of infected hosts (bots) that works independently under the control of a *Botmaster (Bot herder)*, which issues commands to bots using *command and control (C&C)* servers. Bot-net architectures have advanced over time, to evade detection and disruption. Traditionally, bot-nets used a *centralized client-server architecture* which had a single point of failure but with the advent of peer-to-peer technology, the problem of single point of failure seems to have been resolved. Gaining advantage of the decentralized nature of the P2P architecture, botmasters started using P2P based communication mechanism. *P2P bot-nets* are highly resilient against detection even after some bots are identified or taken down. P2P bot-nets provide central frameworks for different cyber-crimes which include DDoS (Distributed Denial of Service), email spam, phishing, password sniffing, etc. In this paper, we propose *PeerClear*, an approach for identifying P2P bot-nets using network traffic analysis. *PeerClear* uses a two-step process for identifying P2P bots. In the first step, the hosts involved in P2P traffic are detected and in the second step, the detected hosts are further analyzed to detect bot-nets. Our evaluation shows that our approach *PeerClear* outperformed several recent approaches and achieves a high detection rate of 99.85%. We also implement multiple new approaches reported in the literature and test on the same dataset to evaluate their relative performance.

Keywords: Bot-net · Dynamic analysis · Machine learning · Malware detection

1 Introduction

According to world Internet user statistics [1], almost 50% of the world population is connected to the Internet. Individuals use it for communication, banking transaction, information seeking, leisure purpose, etc. Organizations use it for their business, connecting with their customers, partners, suppliers, etc. Such widespread usage of the Internet leads us to the new era of cyber crimes. According to IANS (Indo-Asian News Service), cyber crimes in India rose 19 times between 2005 to 2014 and this is based only on the attacks that have been exposed. Currently, there is a combat between hackers and defense agencies.

Among all the cyber attacks, the bots seem to be one of the biggest players in many cyber crimes. Bots are infected machines under the control of an attacker and the network of such infected machines constitutes the bot-net [23]. Bot malware turns the computer into a robot that carries out tasks based on the commands sent to it over the Internet. Bot-nets provide a number of resources to the attackers such as bandwidth, computing power, IP diversity, etc., which allow attackers to commit cyber crimes on a larger scale. According to Vinton Gray Cerf known as the "Father of the Internet", one-quarter of all world computers are part of one or the other bot-net [4].

Realizing the gravity of bot-nets, researchers started studying bot-nets and methods to mitigate them. Traditionally bot-nets used client-server architecture to communicate among themselves, which has a single point of failure and is easier to detect. To make the bot-net resilient against detection, cyber criminals started using new architectures for bot-net communication. Peer-to-Peer (P2P) architecture for bot-net communication comes out to be the most prominent one, avoiding the single point of failure problem of the client-server architecture. One such bot-net is Zeus or Zbot that had become the largest bot-net in the world estimated to affect 3.6 million PCs around the globe according to Damballa [8]. In this paper, we present a new method for P2P bot-net detection and carry out experiments to compare the performance of our method against recently reported methods in the literature.

To summarize our contributions:

- We propose a novel approach, *PeerClear* to detect P2P bot-net over the network using a flow-based approach.
- We have used a two-phase P2P bot-net detection scheme. In the first phase, we identify the P2P hosts on the network and in the second phase, the identified hosts are further analysed to detect P2P bot-nets from P2P-benign applications.
- Our experimental analysis shows that *PeerClear* achieves high detection accuracy of 99.85% which is better as compared to the other authors' work.

In the next section, we discuss the background study of the research. In Sect. 3, we discuss the related work. Section 4 depicts our approach to detect P2P bot-nets. We summarise our results in Sect. 5 and lastly, Sect. 6 concludes the paper.

2 Background

Bot-net is derived from the word *"Robot"* and *"Network."* It is a network of infected hosts or zombies that run automatically and autonomously under the control of an individual or organization. Generally, a bot-net has three working components, first is the attacker, referred to as *botmaster* or *bot herder*, second is the *Bot*, and third is the *Command & Control (C&C)* server.

In bot-nets, the Botmasters directly communicate with the bots using C&C servers. The botmasters are directly connected to the bots because they have a smaller attack domain. However, for bots having larger attack domain such as Zeus [15], Waledac [18], etc., botmaster is connected to the bots through intermediate hosts. These intermediate hosts act as a C&C servers. Nowadays most of the researchers are interested in tracking C&C servers for identifying structures of the bot-net, and these intermediate hosts complicate the process of tracing back the botmaster from detected bots. The most important component in a bot-net is the structure of its command and control channel that prevents the bot-net from being dismantled easily. The communication used among C&C servers and bots can be of the following two types:

- *Push-Based Approach:* In the push based approach, the botmaster pushes the commands into the bots to attack. The advantage of this approach is that the botmaster can instantaneously perform certain tasks through bots. Thus botmaster has higher control over the bots. The main disadvantage of this approach is that the amount of traffic generated is high and two bots infected by the same bot-net, thus having similar traffic patterns, may lead to an easier detection of bots.
- *Pull-Based Approach:* In the pull based approach, the bots periodically receive commands from the server. The server can introduce a random delay while delivering commands, so the bot-net has control over generated traffic. This prevents easy detection of bots and servers. In this approach, instantaneous execution of the command is not possible.

In this work, we are more concerned with P2P bot-net detection, which is a collection of heterogeneously distributed resources connected by a network. The most distinctive difference between client server networking and P2P networking is the existence of *servents* (host that may act both as a server and a client at the same time) [22].

Peer-to-Peer (P2P) Bot-net: P2P bot-nets are the most complex bot-net known so far. The primary objective of P2P bot-nets is to remove or minimize single point of failure problem of the IRC (Internet Relay Chat) /Web bot-nets [21]. In P2P bot-nets, all bots form a P2P network which enables them to communicate and share files across the bot-net. P2P bot-nets help the attacker to inject commands at any point in the network by routing it to all the bots. This activity needs commands to be authenticated to prevent unauthorized injection of commands. Authentication mechanisms such as public key cryptography are often used. The bots need to have access to atleast one other active node to remain connected to a P2P bot-net. For this purpose, some bot-nets use hard-coded lists of peers while some others use network scanning. Examples of P2P bot-net include storm bot-net, first identified around January 2007 [12]. This bot-net infected around 50 million systems worldwide.

3 Related Work

In 2014 Yin et al. proposed a node based detection approach for detecting P2P bot-nets [24]. They extracted network characteristics of individual hosts with time intervals of 10, 20, 30, 60, 180 min. The captured data is sampled for reducing the overhead of the defense system. They have used decision tree classifiers because of its low computational complexity and high performance. They have used only offline traffic and did not evaluate the performance of their approach on online traffic.

Rodriguez-Gomez et al. proposed an approach to detect malicious applications associated with P2P bot-nets based on resources shared by the number of peers in a P2P network [20]. The bot-net resources are the popular resources and have a shorter lifetime as compared to legitimate resources. Therefore, the main inspiration behind the approach is that the resources shared by bots in a P2P network (bot-net resources) will be accessed in a different way than the resources shared by the legitimate users in the P2P network. They trained two models, one for legitimate and other for bot-net resources. Using these models, they have found potential bot-net resources in the P2P network.

Peerminor [13], a pure behavioural system for classifying P2P bots into families, was presented by Kheir et al. They have used a two-stage classifier for this purpose. In the first stage, they have built a classifier to ignore benign P2P traffic and considered only malicious P2P traffic to reduce packet monitoring overhead. In the second stage, they have built one class classifier for known P2P bot families to classify the detected bots into respective families further. *Peerminor* used flow-based features e.g, number of packets sent and received, number of bytes sent and received, flow duration and protocols used. It is the first detection approach that gave information about the type of bot-net infecting the systems. Their training data-set has 794 benign P2P clusters and 1445 malicious P2P clusters. *Peerminor* achieves 97% accuracy for all classes collectively.

Dilon designed a P2P bot detection algorithm using live NetFlow data [10]. NetFlow is a network protocol analyzer for observing network traffic and gathering IP traffic data developed by Cisco. In this work, the Zeus bot-net network traffic was considered and aimed at the detection of individual P2P bots within a network perimeter. For this, they have filtered P2P traffic, the hosts with more than four failed connection is considered as P2P host, and others are discarded. For detecting Zeus, they have used two major features. Firstly, they have used packet ratio, up packets divided by down packets with a threshold of 0.4. Secondly, they have used the traffic patterns. The Zeus bot-net control loop periodically wakes up and contacts peers for P2P network configuration and can be detected by the traffic pattern.

Narang et al. have presented, *PeerShark* [17] for detecting P2P bot-net in the stealthier state (a state where bot-net network activity is almost negligible). *PeerShark* does not require Deep Packet Inspection (DPI). Rather than using the traditional 5-tuple (source IP, source port, destination IP, destination port, protocol) based approaches, they have used a two tuple (port oblivious and protocol oblivious) conversation based approach. They have the following four modules in their model. Firstly, Packet Filtering Module that filter IPv4 packets from network traffic. Secondly, the conversation creation module that creates a list of conversations. Then the conversation aggregation module aggregates conversations into single conversations based on some higher flow-gap value. Finally, the classification module that used supervised machine learning algorithms to train their model. Their training data-set consisted of 50,000 conversations. *PeerShark* used the packet header information of TCP/UDP/IP to extract a set of features such as the duration of the conversation, the inter-arrival time of packets, the amount of data exchanged, a median value of inter-arrival time to classify various P2P applications with an approximate accuracy of 97%.

In another approach, Hojjat et al. [6] proposed a botnet detection based on behavioral analysis of the traffic. The proposed model detects P2P botnets in the command and control phase of the life cycle of the botnet. In this phase, the bot tries to set up a connection to its command and control server and then communicates with the botnet. The proposed model is based on the inferences that bots of a botnet have uniform traffic behavior and bears specific traffic patterns during communication. Hence, the methodology is independent of the content and can also detect P2P botnets which use encrypted traffic. The authors have considered a total of 9930 botnet traffic packets with 3296 extracted flows and 14680 normal traffic packets with 1233 extracted flows. They have used various classifiers such as Bayesian Network Classifier, Naive Bayes Classifier, Support Vector Machine, J48 Decision Tree Classifier, and Random Forest Classifier. The maximum botnet detection accuracy achieved was 99.26% using Random Forest Classifier.

Himanshi et al. [9] proposed a model based on the bot behavior. A two-tier framework was implemented to detect parasitic P2P bots. There are three stages in a P2P botnet lifecycle namely – infection stage, waiting stage, and execution stage. The proposed model detects the bot in the waiting stage i.e. before going into the execution stage. Hence, it did not require any bot signatures. The proposed model considered the features like bot's lifetime in the P2P network, search request intensities, and time correlated behavior for detection of the botnet. The authors have considered 41,941,536 malicious P2P data packets and 25,913,400 Benign Peers packets. The maximum detection accuracy achieved by the proposed model was approx 99%.

In 2016, Alauthaman et al. [5] came up with another P2P botnet detection method which implemented an adaptive multilayer feed-forward NN (Neural Network) using Decision Trees. A network traffic reduction mechanism was introduced to increase the performance. It being a connection-oriented method did not require any Deep Packet Inspection (DPI). Hence, the model was independent of payload and used only the header information of TCP control packets. For feature selection, a classification and regression tree method was used. From these features, a multilayer feed-forward NN was trained using back propagation learning algorithm. The model achieved an accuracy of 99.20%.

Although the above approaches are able to classify or detect the P2P botnet with high accuracy but none of them have sophisticated P2P traffic categorization methodology. They have used a more straight forward approach for categorizing P2P traffic like failed connections threshold, destination diversity threshold, etc. To note that *PeerShark*, the authors have used conversation based approach and they have not differentiated the P2P and non-P2P traffic that we have used in our work. Discarding non-P2P traffic have a greater impact on the computational overhead of the developed system as less traffic needs to be monitored. They have a good P2P traffic categorization method, but they are using the same approach for classifying botnet as used for P2P traffic categorization. Therefore there is a need for a system which can categorize P2P traffic using all properties shown by P2P hosts and has a separate P2P bot detection module for detecting bots using distinctive features shown by P2P botnets. Since the dataset on which these authors calibrated their methods may be different, we have implemented the models reported in [5,6,9,17] and evaluated them on our dataset to obtain a fair comparison.

4 Our Approach

In the previous section, we have discussed various P2P bot-net detection methods proposed by multiple researchers. Studying their approaches, detection methods, and future work, we propose a two-step approach to detect P2P bot-net in the stealthy state (a state where bot-net network activity is almost negligible). Firstly, we have identified all the hosts which are involved in the P2P activity and secondly, we have detected P2P bots in the identified P2P hosts as shown in Fig. 1.

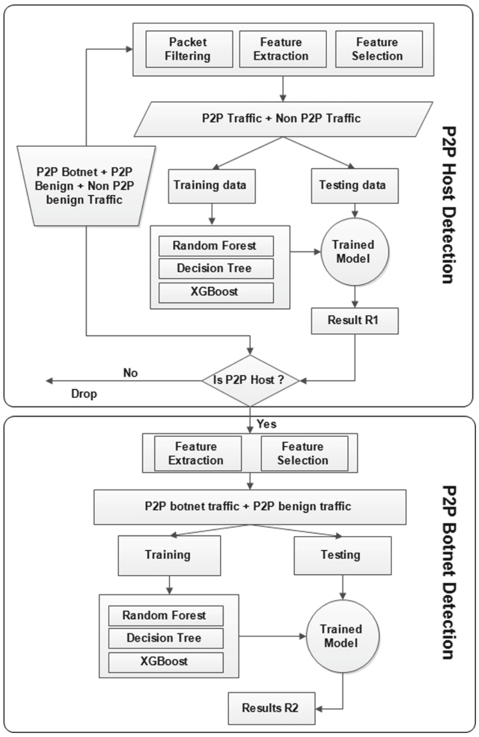

Fig. 1. Flow chart of our approach to detect P2P bot-net.

Dataset. For our experiment we have collected three types of data, i.e., *P2P-Benign*, *P2P bot-net* and *Non-P2P* network traffic.

- *P2P-Benign network Traffic:* It was collected by 11 distinct hosts which executed five different P2P benign applications (Skype, eMule, μ-Torrent, Frostwire, and Vuze.) for several days.
- *P2P bot-net Traffic:* This data was collected from *Peerrush* [2] dataset which contains the P2P bot-net traffic of Storm [12], Waledac [18], and Zeus [15] and also the P2P bot-net traffic generated from *Vinchuca* bot-net [16].
- *Non-P2P Traffic:* It was obtained from the departmental network which was being observed over five days. Network sniffing tool based on *libpcap* was used to capture the packets.

All the above data was captured in the form of a .pcap file which contains the network information.

4.1 P2P Host Detection

The main aim of this phase is to detect all the hosts which were engaged in P2P activity. It consists of four modules namely packet filter, feature extraction, feature selection and classification.

Packet Filter. In this module, unwanted packets such as multicast, broadcast and DNS generated traffic (P2P network does not use DNS) were filtered out and the rest was sent to feature extraction module as shown in Fig. 2. This filtering reduce the packet monitoring overhead and the processing time.

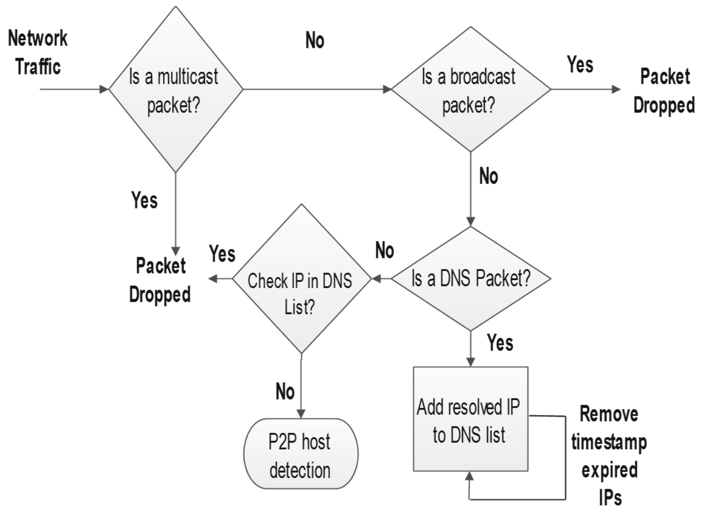

Fig. 2. Packet filter module.

Feature Extraction. To find out the prominent features for the detection of P2P hosts, the distinctive properties shown by the hosts engaged in the P2P activity as opposed to the hosts with Non-P2P activities were studied and discussed as follows:

– *Failed Connections:* In the P2P network, nodes may continuously join and leave the network. To remain connected, the peers must be connected to at least one of the peers. So peers in the P2P network continuously search for new peers. While searching for new peers, many peers may not be available in P2P network because of the continuous process of joining and leaving, so the number of failed connection attempts in the P2P network is usually higher. Regular Internet traffic did not encounter such a high number of failed connection attempts.
– *DNS filter:* Peers in the P2P network operate outside of the DNS system. Peers did not use the DNS queries to search other peers. They get it directly from the overlay network's routing table. Although for connecting to the central server, they may need to make a DNS request, which was very rare. A regular Internet user usually uses the Internet browser to visit some popular websites which were mostly resolved by DNS requests. We implemented this component in the packet filter module.
– *Destination diversity:* Since the IPs of the peers are usually scattered across many different networks, the diversity of IPs (IP domain) contacted by P2P peers in the P2P network is typically large. For all the IPs contacted by a peer, we have computed a set of/16 prefix of each destination IP. It gives an approximate idea of IP domains visited by the peer. The size of this set is the destination diversity of the peer. We have also used *destination diversity ratio* calculated by dividing destination diversity with the total number of distinct IPs contacted by the peers.

Based on the above properties, we have used *tshark* (a network protocol analyzer) [3] tool to extract the features from pcap files (Table 1).

Feature Selection. In feature extraction, we have extracted fourteen features based on the P2P host behavior. However, we found that all the features are not important while training the classifiers. There may be some features which do not affect the performance of the classification or perhaps make the results worse. Therefore, in this section, we apply the feature reduction technique to reduce the dimensionality of the feature vector. Information gain algorithm is used as a measure for feature reduction. As shown in Fig. 3, top 2 to 14 features with highest info-gain score were selected for classification. The final feature vectors consist of the extracted top 10 features because we ran the classifiers on top 2 to 14 features and then ten-fold cross validation, and the accuracy comes out to be maximum for the top 10 features which are demonstrated in Fig. 4. Fourteen features (Table 1) based on the P2P host behavior were extracted. The final feature vector used for the classification is: $<F_2, F_3, F_4, F_5, F_8, F_9, F_{11}, F_{12}, F_{13}, F_{14}, label>$

Table 1. Extracted network traffic features for P2P host detection

Feature_Id	Feature	Description
F_1	ret_count	Retransmitted packet count
F_2	diversity	Destination diversity
F_3	diversity_ratio	Destination diversity ratio
F_4	no_pkt_out	Number of connection attempts made on distinct port
F_5	distinct_ip	Number of distinct IPs contacted
F_6	reset	Reset packets count
F_7	out_of_ord	Out of order packets count
F_8	icmp	ICMP destination unreachable packets count
F_9	flows	Number of packets sent and received
F_10	byte_in	Bytes per packet in forward direction
F_11	byte_out	Bytes per packet in backward direction
F_12	dis_ret_count	Average retransmitted packets per host count
F_13	dup_ack	Duplicate ack packets count
F_14	ctrl_pkt	Total number of control packets (packet without data) sent and received

Classification. The collected traffic data were categorised into two groups, first group is used to train the classifier, and the other group is used to test the classifier. The training group consisted of 70% of the instances, and our testing group consisted of 30% of the instances. For the selection of classification model, we have used 10-fold cross validation on Random forest [11], Decision Tree [19] and XGBoost [7] classifiers to detect P2P hosts from the captured traffic.

For P2P host detection, the data were extracted for three different time windows of 10 min duration and the results are shown in Table 2. The results in terms of true-positive rate (TPR), false-positive rate (FPR), precision and accuracy are summarized in Table 2.

Table 2. P2P host detection results (R1)

Classifier	TPR	FPR	Precision	Accuracy
Random forest	99.91%	0.003%	99.98%	99.93%
Decision tree	99.89%	0.001%	99.92%	99.88%
XGBoost	99.73%	0.001%	99.92%	99.78%

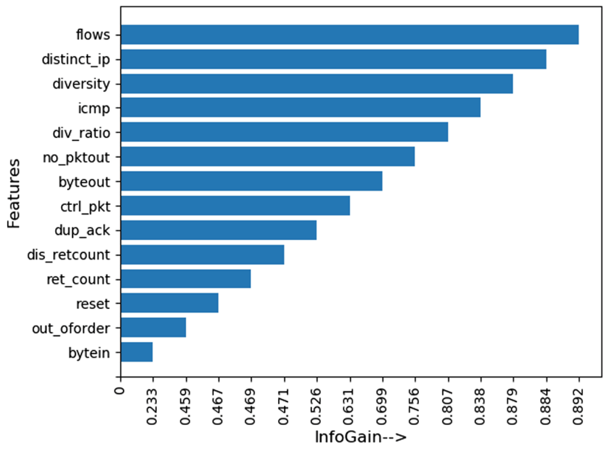

Fig. 3. Information gain

4.2 P2P Bot-net Detection

After the identification of P2P hosts over the network, in this phase, the bot-net was detected from the identified P2P hosts. This phase consists of three modules, i.e., feature extraction, feature selection and classification.

Feature Extraction. For the P2P bot-net detection, the flow-based approach was used and the data was extracted for 1-hour time window, to trace the stealthier nature of P2P bots. A network flow is a set of packets exchanged between two hosts. Network traffic flow is uniquely identified by five tuples ⟨source IP, source port, destination IP, destination port, protocol⟩. The conversation is defined by the help of binary tuple ⟨source IP, destination IP⟩ and vice versa. All the conversations are categorized as port and protocol oblivious. In this work, we have used only flow-based features.

P2P protocols use transport layer protocols to share the files, so both the TCP and UDP traffic are captured for our experimental analysis. To distinguish between the P2P benign traffic and the P2P bot-net traffic, we have focused on *management flows*, i.e., the network traffic which is used to maintain the updated information about the network. Once the bot-net infects any host, in order to remain connected to that host, bot-net continuously sends the control packets as keep-alive messages to the bot. Bots' communication in the waiting state is quite stealthy. These control packets provide useful insights into the bot-net communication pattern.

Moreover, the management flow depends on the protocol design whereas the data flow depends on the user. The data flows are usually regulated by the user interaction with the P2P applications. The usage of the P2P applications varies from user to user. Relying on the management flows allow more universally, user-independent P2P bot-net detection approach. We have not completely discarded the data flows. Some of the features were also obtained from the data flows as well.

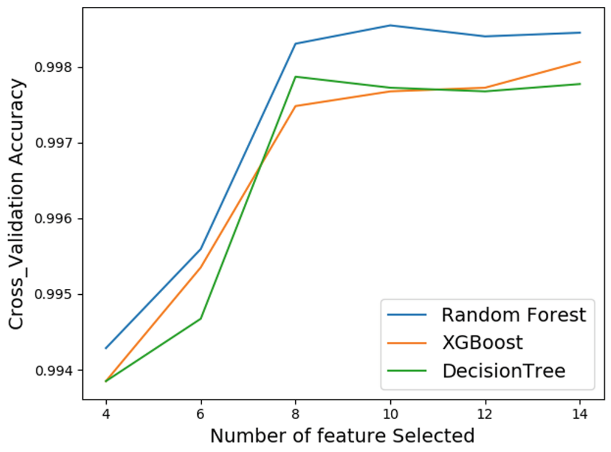

Fig. 4. Ten-fold cross validation

Now the question arises, how to separate the management and data packets. This is because management packets are generally embedded inside the data packets, and sometimes are sent separately. Below are few heuristics considered to separate management from data packets.

- *Inter packet time:* The management packets are exchanged periodically whereas the data packets are sent continuously one after another. Therefore, inter packet time between the data packets are usually very small. On the other hand in the management packets inter packet time is large. We consider only those flows in which inter packet time was greater than a particular threshold θ. For example, consider a packet P_i and packets seen before and after P_i as P_{i-1} and P_{i+1}. Now say inter packet time between packets P_i and P_{i-1} was Δ_{i-1} and that of between P_i and P_{i+11} was Δ_{i+1} then we consider P_i as management packet if Δ_{i-1} and Δ_{i+1} both are greater than θ.
- *Duration of Flow:* P2P network flows are generally long-lasting. Instead of creating a new connection, peers exchange the management packets to keep the connection alive. Same as in the case of the P2P bot-net, to prevent losing connection with a bot, they periodically exchange control messages.

Our main concern is to extract those features which distinguish the P2P botnet from P2P benign traffic. The communications used by P2P bot are low in volume because the bots are controlled by the bot-master and they continuously communicate with each other to remain connected. Hence, the duration of this communication is large. For P2P benign applications like μ-Torrent, users generally download large files such as music, videos, etc. On the other hand, Bots do not download such large files. Rather they continuously send information to the bot-master. Also, the inter-arrival time between the packets for bots is more as compared to P2P benign applications, because of the reasons discussed above. Therefore by using these features, we can prominently distinguish P2P bot-net traffic from P2P-benign traffic. Table 3 shows the used features captured by *pyshark* [14].

- *Host Access Features:* These features were used to capture the host accessing pattern of the bot-nets. These include features like inter-arrival time of packets, maximum inter-arrival time, minimum inter-arrival time, etc. to capture the distribution of inter-arrival time of flow at any host.
- *Flow Size Features:* These features were used to capture the distribution of both incoming and outgoing flows at a specific host. These include features like packets or bytes sent and received in the flow, to capture their distribution of the flow at any host. Other examples of flow size features include the number of bytes sent or received in the flow, smallest packet seen in the flow, the largest packet seen in the flow, etc.

Table 3. Extracted network traffic features for P2P bot-net detection

ID	Features	Description
F1	mean_inter_time	Mean of the Inter-arrival time between packets
F2	fwd_pkt	Number of packets sent in flow
F3	bkd_pkt	Number of packets received in flow
F4	frwd_bytes	Number of Bytes sent in flow
F5	bkd_bytes	Number of Bytes Received in flow
F6	total_data	Total data sent and received in flow including headers
F7	small_pkt	Smallest packet in flow
F8	large_pkt	Largest packet in flow
F9	max_inter_time	Maximum Inter-arrival time between any two packets in flow
F10	min_inter_time	Minimum Inter-arrival time between any two packets in flow
F11	total_duration	Total duration of flow
F12	pkt_frequency	Packet frequency (flow duration/ number of packets in flow)
F13	mean_fwd_inter_time	Mean inter-time between packets sent in forward direction
F14	mean_bkd_inter_time	Mean inter-time between packets sent in backward direction
F15	max_fwd_inter_time	Maximum inter-time between packets sent in forward direction
F16	min_frwd_inter_time	Minimum inter-time between packets sent in forward direction
F17	max_bkd_inter_time	Maximum inter-time between packets sent in backward direction
F18	min_bkd_inter_time	Minimum inter-time between packets sent in backward direction

Feature Selection. The extracted 18 features based on host access patterns and flow size features were further reduced for bot-net detection. We have used information gain feature selection algorithm to reduce the dimensionality of the feature vector. The top 2 to 18 features with highest info-gain scores were selected (Fig. 5) and respectively used for the classification.

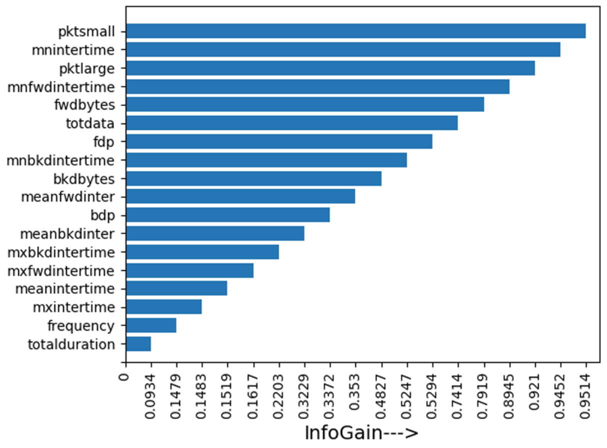

Fig. 5. Information gain

Bot-net Classification. For P2P bot-net detection, we have used the same classification algorithm used in P2P host detection (Random forest, Decision tree, and XGBooost) for the classification of P2P botnet detection. To find the best number of features for the best performance of all selected classifiers, we ran classifiers on 2 to 18 features with the highest info-gain score and obtained results are shown in Fig. 6 and Table 4. The observation of results shows that all the classifier performs with more than 99% accuracy and among them, Random forest outperformed with 99.99% accuracy while using only top 6 features.

Table 4. P2P bot-net detection result (R2)

Classifiers	TPR	FPR	Precision	Accuracy
Random forest	99.98%	0.002%	99.99%	99.99%
Decision tree	99.97%	0.004%	99.97%	99.97%
XGBoost	99.77%	0.024%	99.97%	99.88%

We have also trained the model for P2P bot-net detection using traffic from three bot-nets namely Waledac, Vinchuca, and Zeus. For P2P benign applications, we have used traffic from Skype, eMule, Frostwire, and Vuze. The model

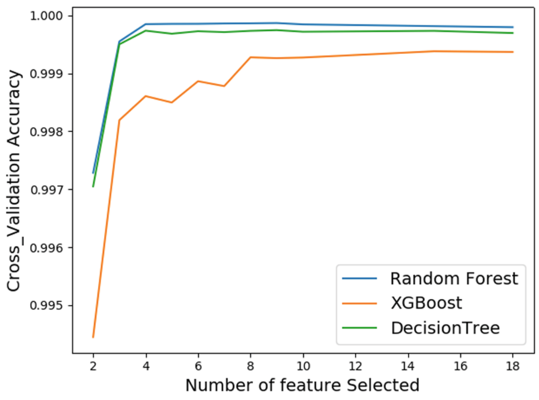

Fig. 6. P2P botnet detection accuracy of selected classifiers

is tested using unseen traffic from a different bot-net i.e. Storm for P2P bot-net and μ-Torrent for P2P benign applications. The model achieves an accuracy of 97% for above-mentioned experiments. This leads us to believe that the approach can be generalized for other bot-net detection. This is perhaps due to the fact that traffic flow features of most P2P bot-nets are very similar.

5 Results and Comparison with Past Work

The overall performance of our system is determined by passing the entire traffic into P2P host detection module. The entire traffic consists of non P2P traffic and P2P traffic. The first module filters P2P hosts from non-P2P hosts. The filtered P2P hosts can be P2P bot-net hosts or P2P benign hosts. This traffic is now fed into the other module i.e. P2P bot-net detection module. P2P Bot-net detection module differentiates P2P bot-nets from P2P benign traffic. The overall accuracy of the system as a whole is 99.85%. The results show that our approach outperforms the results reported previously in the literature. We have also performed experiments using other proposed approaches on our traffic flows to check how much better is our proposed model in terms of accuracy. It shows that our model performs better as compared to their models and achieves better accuracy. We have considered the most recent models discussed in the literature. The results presented in their papers and the ones obtained by using their approach on our dataset are summarised in Table 5.

In our proposed model, there are 7,11,149 P2P botnets, and 8,15,659 benign P2P traffic flows taken into account. Table 5 shows the exact amount of traffic flows, conversations, and packets considered by the different authors. In [9] and [5], the authors notified the packets but not the flows. Similarly, in [17] authors notified the number of conversations. We want to mention here that there were few other work discussed in Sect. 3, but we did not compare them as the information about the features, or the statistical methods was missing from their papers to faithfully reimplement them (Fig. 7).

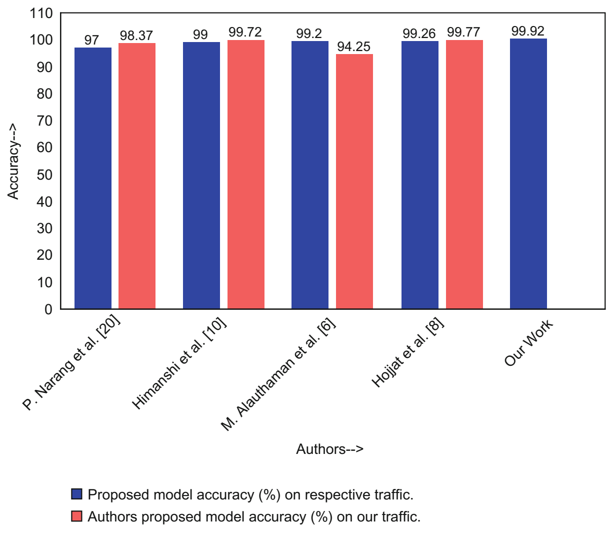

Fig. 7. Accuracy(%) of Authors' proposed model on their's and as well as on our traffic.

Table 5. Accuracy (%) comparison.

Authors	Dataset	Approach	Author reported accuracy %	Authors proposed model accuracy on our traffic
Narang et al. [17]	P2P Botnet Conversations - 50000 P2P Benign Conversations - 50000	Conversation-based	Approx 97%	98.37%
Hojjat et al. [6]	P2P Botnet Flows - 3296 P2P Benign Flows - 1233	Flow based	99.26%	99.77%
Himanshi et al. [9]	P2P Botnet Packets - 41941536 P2P Benign packets - 25913400	Flow based	Approx 99%	99.72%
Alauthaman et al. [5]	P2P Botnet Control packets - 114087 P2P Benign Control packets - 331526	Flow based	99.20%	94.25%
Our approach	P2P Botnet flows - 711149 P2P Benign flows - 815659	Flow based	99.85%	N/A

6 Conclusion

In this work, we have discussed *PeerClear*, an approach for detecting P2P bot-nets using network traffic analysis. The detection of bots was done in two steps, i.e., P2P host detection and P2P bot-net detection. In P2P host detection phase, we have performed packet filtering, feature extraction, and classification. Packet filtering module filters unwanted packets which were not contributing to the classification. Feature extraction module converts network traffic into the host-based feature vectors. Classification module uses decision tree for classification of extracted feature vectors into P2P or non-P2P. After the identification of P2P hosts, we have detected P2P bot-net by three other modules i.e., feature extraction, feature selection, and classification. The feature extraction module extracts the flow-based feature vectors from the network traffic. The feature selection module selects essential features based on the information gain ratio algorithm. Finally in the classification module, Random Forest, XGBoost and decision tree, classified the P2P bot with more than 99% of accuracy. The overall accuracy of *PeerClear* is (99.85%). We also implemented methods for P2P bot-net detection reported in other papers an evaluated them on our dataset. The results indicate our approach does better than others on the same dataset.

Acknowledgement. This work was partially funded by Science and Engineering Research Board, Government of India.

References

1. Internet world stats (2018). https://www.internetworldstats.com/stats.htm
2. Peerrush (2018). http://peerrush.cs.uga.edu/peerrush/
3. Tshark - Dump and Analyze Network Traffic, March 2018. https://www.wireshark.org/docs/man-pages/tshark.html
4. Vint Cerf: One Quarter of All Computers part of a Botnet (2018). http://www.tmttlt.com/archives/5289/
5. Alauthaman, M., Aslam, N., Zhang, L., Alasem, R., Hossain, M.A.: A P2P botnet detection scheme based on decision tree and adaptive multilayer neural networks. Neural Comput. Appl. **29**(11), 991–1004 (2018)
6. Beiknejad, H., Vahdat-Nejad, H., Moodi, H.: P2P botnet detection based on traffic behavior analysis and classification. Int. J. Comput. Inf. Technol. **6**(1), 01–12 (2018)
7. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794. ACM (2016)
8. Comodo: Latest malware attacks, May 2018. https://enterprise.comodo.com/blog/tag/latest-malware-attacks/
9. Dhayal, H., Kumar, J.: Peer-to-Peer botnet detection based on bot behaviour. Int. J. Adv. Res. Comput. Sci. **8**(3), 172–175 (2017)
10. Dillon, C.: Peer-to-Peer botnet detection using NetFlow. Master's thesis, University of Amsterdam (2014)
11. Donges, N.: The Random Forest Algorithm (2018). https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd
12. Holz, T., Steiner, M., Dahl, F., Biersack, E., Freiling, F.: Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. In: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats (2008)
13. Kheir, N., Han, X., Wolley, C.: Behavioral fine-grained detection and classification of P2P bots. J. Comput. Virol. Hacking Tech. **11**(4), 217–233 (2015)
14. KimiNewt: Python wrapper for tshark, allowing python packet parsing using wireshark dissectors, June 2018. https://github.com/KimiNewt/pyshark

15. Lelli, A.: Zeusbot/Spyeye P2P Updated, Fortifying the Botnet (2018). https://www.symantec.com/connect/blogs/zeusbotspyeye-p2p-updated-fortifying-botnet
16. Lontivero: A Resilient Peer-to-Peer Botnet Agent in.NET, April 2017. https://github.com/lontivero/vinchuca
17. Narang, P., Ray, S., Hota, C.: PeerShark: detecting peer-to-peer botnets by tracking conversations. In: IEEE Security and Privacy Workshops (2014)
18. Nunnery, C., Sinclair, G., Kang, B.B.: Tumbling down the rabbit hole: exploring the idiosyncrasies of botmaster systems in a multi-tier botnet infrastructure. In: Proceedings of the 3rd USENIX Conference on Large-Scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More (2010)
19. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
20. Rodriguez-Gomez, R.A., Macia-Fernandez, G., García-Teodoroa, P., Steiner, M., Balzarotti, D.: Resource monitoring for detection of parasite P2P botnets. Comput. Netw. **70**, 302–3011 (2014)
21. Saiyod, S., Chanthakoummane, Y., Benjamas, N., Khamphakdee, N., Chaichawananit, J.: Improving intrusion detection on snort rules for botnet detection. Softw. Netw. **2018**(1), 191–212 (2018)
22. Schollmeier, R.: A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In: First International Conference on Peer-to-Peer Computing (2002)
23. Singh, S.C.: High-tech and computer crimes: global challenges, global responses. In: Nirmal, B., Singh, R. (eds.) Contemporary Issues in International Law, pp. 413–437. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-6277-3_30
24. Yin, C.: Towards accurate node-based detection of P2P botnets. Sci. World J. **2014**, 10 p. (2014)