2014 IEEE Security and Privacy Workshops

PeerShark: Detecting Peer-to-Peer Botnets by Tracking Conversations

Pratik Narang, Subhajit Ray, Chittaranjan Hota
Department of Computer Science & Information Systems
Birla Institute of Technology and Science-Pilani, Hyderabad Campus
Hyderabad, A.P., India
Email: {p2011414, f2010452, hota}@hyderabad.bits-pilani.ac.in

Venkat Venkatakrishnan
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607
Email: venkat@cs.uic.edu

***Abstract*—The decentralized nature of Peer-to-Peer (P2P) botnets makes them difficult to detect. Their distributed nature also exhibits resilience against take-down attempts. Moreover, smarter bots are stealthy in their communication patterns, and elude the standard discovery techniques which look for anomalous network or communication behavior. In this paper, we propose PeerShark, a novel methodology to detect P2P botnet traffic and differentiate it from benign P2P traffic in a network. Instead of the traditional 5-tuple 'flow-based' detection approach, we use a 2-tuple 'conversation-based' approach which is port-oblivious, protocol-oblivious and does not require Deep Packet Inspection. PeerShark could also classify different P2P applications with an accuracy of more than 95%.**

I. Introduction

The past decade saw immense rise of the peer-to-peer computing paradigm. In the beginning of the 21st Century, the P2P architecture attracted a lot of attention of developers and end-users alike, with the share of P2P over the Internet in different continents being reported to be in the range of 45% to 70% [1]. As more and more users got access to powerful processors, large storage spaces and increasing bandwidths, P2P networks presented a great opportunity to share and mobilize resources. However recent reports [2] show that the share of popular P2P applications over the Internet has declined to a mere 10%. The P2P paradigm has been plagued with the issues of privacy, security and piracy to name a few, and the last decade has seen a lot of research focus in these areas [3], [4], [5].

Peer-to-Peer overlay networks are distributed systems consisting of interconnected nodes which self-organize into network topologies. They are built with specific purposes of sharing resources such as content, CPU cycles, storage and bandwidth, and have the ability to accommodate a transient population of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or support of a global centralized server or authority [6]. The construction of P2P networks is on the top of IP layer, typically with a decentralized protocol allowing 'peers' to share resources. The runaway success of P2P applications is primarily attributed to the ease of resource sharing provided by them- be it in the form of music, videos, files (BitTorrent [7]), sharing of computing resources (SETI @ home [8]). Apart from these, P2P paradigm has also been widely deployed for IPTV (LiveStation [9]) and Voice-over-IP (Skype [10]) based services.

As P2P networks are inherently modeled without any centralized server, they lack a single point-of-failure [11]. This resilience offered by P2P networks has also attracted the attention of adversaries in the form of bot-masters (a.k.a. bot-herders). A 'bot' is a computer program which enables the operator to remotely control the infected system where it is installed. A network of such compromised end-hosts under the remote command of a master (i.e., the bot-master) is called a 'Botnet' [12]. The ability to remotely command such bots coupled with the sheer size of botnets (numbering to tens of thousands of bots) gives the bot-masters immense power to perform nefarious activities like spamming, Bitcoin mining, click-fraud scam, Distributed Denial of Service (DDoS) attacks etc. on a massive scale, and in turn generate millions of dollars per year in revenue for the bot-master[13].

Traditional botnets were known to use IRC (Internet Relay Chat), which implied a centralized architecture for their 'Command & Control' (C & C) operations. Detecting the centralized C & C server meant bringing down the entire botnet. Bot-masters have utilized the resilience offered by P2P networks to build botnets wherein bots communicate, pass on commands and update other bots in a P2P fashion [14]. Just as a P2P network is resilient to break-down if a few peers leave the network, P2P botnets have proven to be highly resilient even if a certain number of bots are identified and taken-down [15], [16].

A P2P bot's life cycle consists of the following stages:

- Infection stage, during which the bot spreads (this might happen through drive-by downloads, a malicious software being installed by the end-user, infected USB sticks, etc.).
- Rally stage, where the bot connects with a peer list in order to join the P2P network.
- Waiting stage, where the bot waits for the bot-master's command (and does not exhibit much activity otherwise).
- Execution stage, in which it actually carries out a command, such as a Denial of Service (DoS) attack, generate spam emails, etc.

To evade detection by Intrusion Detection Systems (IDSs) and Firewalls, botnets tend to keep their communication patterns (with the bot-master or other bots) quite stealthy.

DOI 10.1109/SPW.2014.25

IDSs and Firewalls which rely on anomalous communication patterns to detect malicious behavior of a host are not very successful in detecting such botnets which are very stealthy and 'lie low', since they generate little traffic and thus pass under the radars of IDSs/Firewalls.

With the advent of the Internet of things, the possibility of malware taking control of 'smart' appliances such as Television, Air-conditioners, Refrigerators etc. will not be limited to theory. In fact, there have been recent claims that a 'smart' refrigerator connected to the Internet was a part of a botnet and involved in sending thousands of malicious emails [17]. As the creators of botnets continue to adopt innovative means in creating botnets, botnet detection continues to be a challenging area of research.

In the next sections, we describe related work and our approach, labeled PeerShark. Section IV describes the system design of PeerShark. The implementation details of PeerShark are discussed in Section V. Section VI presents the results of evaluation of PeerShark with test datasets, and Section VII discusses possible methods by which bots may evade PeerShark. Conclusion and Future work are described in Section VIII.

II. Related Work

Most prior work has either focused on P2P traffic classification from the perspective of a more general problem of Internet traffic classification [18], [19], [20], or has given special attention to detection of botnets (centralized or distributed) in Internet traffic [21], [22], [23]. The detection of P2P botnet traffic in the presence of benign P2P traffic has not received much attention. Furthermore, the challenging context of correct categorization of the exact P2P application- whether benign or malicious- running on a host has received very little attention in past works [24], [25].

Initial work on detection of P2P botnets involved signature-based and port-based approaches [26], which were easily defeated by bots which randomize their communication ports or use encryption. Although several approaches have been proposed to detect P2P botnets through the analysis of their network behavior, most of them propose a binary classification of P2P hosts (i.e., benign or malicious) [27], [22].

Some of the recent work has used supervised [24] and unsupervised [28], [25] machine learning approaches and other statistical measures [29], and have employed the standard 5-tuple categorization of network flows. Packets were classified as 'flows' based on the 5-tuple: `<source IP, source port, destination IP, destination port, protocol>`. Flows have bi-directional behavior, and the direction of the flow is decided based on the direction in which the first packet is seen. This traditional definition of flows has been greatly employed and has seen huge success in the problems of Internet traffic classification [30], and even in the early days of P2P traffic classification [31]. But since this definition relies on port number and protocol, latest P2P applications as well advanced P2P bots which change / randomize their communication port(s) and operate over TCP as well as UDP will not be well-identified by this approach. Since such a behavior is characteristic of only the latest variants of P2P applications (benign or malicious), it is obvious that past research did not touch upon this aspect.

In response to this, a recent work [23] has used the 2-tuple 'super-flows'(`<source IP, destination IP>`) with a graph-clustering technique to detect P2P botnet traffic. Although authors in [23] presented interesting insight and obtained good accuracy in detecting the traffic of two P2P botnets (Storm and Nugache), their work has several limitations. A graph-clustering approach may not scale as the network size grows. Further, their work evaluates the detection of P2P botnets only with regular web traffic (which was not analyzed for the presence or absence of regular P2P traffic). This is a serious limitation because P2P botnet traffic (quite obviously) exhibits many similarities to benign P2P traffic, and distinguishing between hosts using regular P2P applications and hosts infected by a P2P botnet would be of great relevance to network administrators protecting their network. Moreover, their approach is also limited to a binary classification.

III. Our Approach and Contributions

In this work, we present PeerShark, a 'conversation-based' approach for P2P traffic which can differentiate P2P botnet traffic from benign P2P traffic, and correctly categorize the exact P2P application running on a host inside a network. It aims to be a 'P2P-aware' assistant to network administrators wanting to segregate P2P traffic and detect P2P botnets.

PeerShark does not assume the availability of any 'seed' information of bots through blacklist of IPs, and it does not rely on Deep Packet Inspection (DPI) or signature-based mechanisms (which are rendered useless by botnets/applications using encryption). It aims to detect the stealthy behavior of P2P botnets, that is, when they lie dormant in their rally or waiting stages (in order to pass under the radars IDSs which look for anomalous communication patterns) or while they perform malicious activities (spamming, password stealing, etc.) in a manner which is not observable to a network administrator. PeerShark focuses on observing the different 'conversations' which happen between the peers, which essentially capture the idea of *who is talking to whom*. The conversations are extracted from the information obtained from packet (TCP/UDP/IP) headers. For all conversations, a set of features is extracted which quantifies the inherent 'P2P' behavior of different applications, such as- the duration of the conversation, the inter-arrival time of packets, the amount of data exchanged, etc. Further, supervised machine learning algorithms and network traces of P2P applications & botnets are used to build models which can correctly categorize different P2P applications.

PeerShark adopts a conversation-based approach instead of the traditional flow-based approaches whose demerits have been explained in the previous section. 'Conversation-based' approaches for detection of P2P botnet traffic were first seen in [32]. However, the approach of authors in [32] used 'thirty second conversations' for detection of P2P botnets. Such an approach cannot be applied to botnets which are stealthy and low-lying. Further, their work does not involve categorization of different P2P applications, and is focused towards creating a 'Decision Fusion' (traditionally referred as an 'ensemble classifier' in machine learning circles) with two machine learning approaches (namely Support vector machines and Decision trees). Another recent work [33] has seen the use of 'conversation-based' approach in the P2P domain, but for a different problem- namely, the detection of overlapping P2P communities in Internet backbone. Their work does not focus on identification of any specific P2P application- whether malicious or benign. PeerShark significantly extends past works by addressing the challenging context of detection of stealthy P2P botnets in network traffic in the presence of benign P2P applications, and categorization of the specific type of P2P application running on a host.

To summarize our contributions:

- A 'conversation-based' detection mechanism which is protocol-oblivious, port-oblivious and payload-oblivious, and relies only on the information obtained from the TCP/UDP/IP headers. Thus it does not require DPI, and cannot be evaded by payload-encryption mechanisms.
- Detection of stealthy P2P botnet traffic inside a network, and differentiating it from regular P2P traffic.
- Categorization of the specific type of P2P application (regular or botnet) running on a host (with an accuracy of more than 95%).

IV. PeerShark: System Design

PeerShark relies on understanding the of 'P2P' behavior of P2P applications as well as botnets, and differentiates them based on their differing behavior. Here we explain the concepts which lie at the core of PeerShark.

A. Tracking Conversations

Once a bot-master infects a particular machine, it is in the prime interest of the bot-master to not lose connectivity with his bots. The bot-peers near to each other in the P2P overlay network maintain regular communication amongst themselves to check for updates, to exchange commands and/or to check if the peer is alive or not. If such messages are exchanged very frequently, the bot is at risk of getting detected by IDS/Firewalls protecting the network. Hence the communication between the bot-master and his bots, or that of bots amongst themselves, is expected to be low in volume (note here that this usually corresponds to the rally and waiting stages; execution stages can be aggressive or stealthy depending upon the activity for which the bots are used; DDoS attack can be quite aggressive, while password stealing may remain stealthy). Since certain botnets (and even benign P2P applications) are known to change/randomize their port numbers over which they operate, the regular 'flow' definition will not be able to give a clear picture of the activity a host is engaged in. The traditional 'flow' definition will create multiple flows out of what is actually a single conversation happening between two such peers (although happening on different ports), and thus give a false view of the communications happening in the network. To get a *bird's eye-view* of the *conversations* happening between the P2P hosts can be beneficial for a network administrator to hunt for malicious conversations between the bots. This is the approach which lies at the heart of PeerShark.

B. Categorization of P2P Applications

The other part of PeerShark concerns itself with categorization of a specific P2P application running at the end-host, which might be a regular P2P application or a bot-related activity. As discussed above, P2P bot communications will tend to be low in volume. These bots are automated by the bot-master to keep contacting each other at certain intervals of time, and thus the 'duration' of their conversations will be large. On the other hand, P2P applications are used by average users of the Internet. Such a user can be easily distinguished from an automated bot. Bots do not download and share music videos, movies etc. over P2P networks, whereas regular peers do. Although bots remain connected to each other and thus exhibit long conversations (as will be evident from our experiments), a benign peer A's conversation with another specific peer is not expected to be long since A might download one file from B in Brisbane, share another with C in Cambridge, and download another from D in Denver. Further, all P2P applications- whether malicious or benign- operate with their 'app-specific' *control messages* which are used by peers to connect to the P2P network, make file searches, leave the network, etc. Since each application has its own specific control messages, we exploit the patterns hidden in these control messages to categorize different P2P applications by considering the median value of the *inter-arrival time* of packets for each different P2P application. Moreover, as has been explained before, bot traffic tends to be stealthy. Hence bot conversations are expected to have higher inter-arrival time of packets than benign P2P conversations. Thus inter-arrival time of packets also supplements in differentiation of P2P bot traffic from benign P2P traffic. In summary, we extract four features from the datasets of different P2P applications (malicious or benign) and use them to differentiate P2P botnets from benign P2P users, and categorize the different P2P applications. The four features used in this work are:

1) The duration of the conversation.
2) The number of packets exchanged in the conversation.
3) The volume of data exchanged in the conversation.
4) The median value of the inter-arrival time of packets in that conversation.

P2P benign and malicious datasets were obtained from a recent work by the authors in [24]. Details of the data extracted for this work are:

1) Benign P2P Data : 50,000 conversations each of eMule and uTorrent.
2) Malicious P2P Data : 50,000 conversations each of Storm and Waledac.

V. Implementation details

PeerShark relies on the following four modules: (depicted in Figure 1):

Packet Filtering Module: This module takes in network logs in the form of raw packet data as input. The module reads each packet and isolates those which have a valid IPv4 header. For the purpose of data sanitization, all packets without a valid IPv4 header are deemed invalid and discarded. The packets are further filtered to keep only those packets which have a valid TCP or UDP header and a non-zero payload. From each packet, the Source IP, Destination IP, Payload length and Timestamp are extracted and stored for future use. This module is algorithmically explained in Algorithm 1.

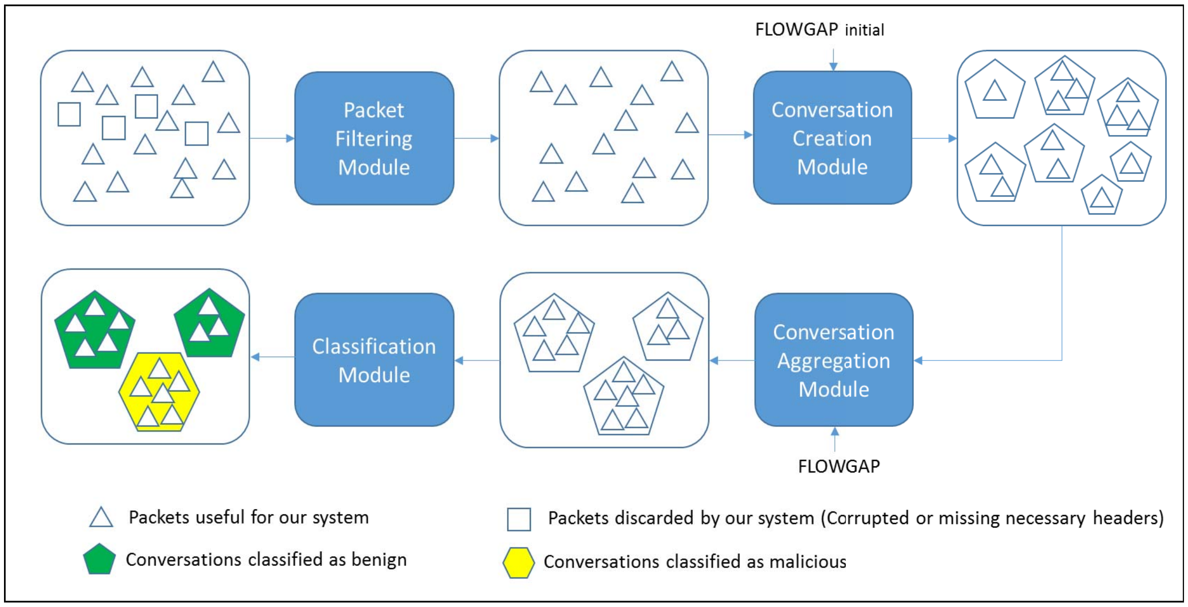
FLOWGAP initial
Packet Filtering Module
Conversation Creation Module
Conversation Aggregation Module
Classification Module
FLOWGAP
Packets useful for our system
Packets discarded by our system (Corrupted or missing necessary headers)
Conversations classified as benign
Conversations classified as malicious

Fig. 1. Flow Diagram

Algorithm 1 Packet Filtering Module

```
procedure FILTERPACKETS(packetCapture)
    $ArrayList < ModifiedPkt > filteredPkts$;
    for Packet p in packetCapture do
        $timestamp \leftarrow p.getTimestamp()$;
        if p has IPHeader then
            $ip \leftarrow p.getIPHeader()$;
            $IP1 \leftarrow ip.getSourceIP()$;
            $IP2 \leftarrow ip.getDestIP()$;
            if p has TCPheader or UDPheader then
                $header \leftarrow p.getTransportHeader()$;
                $pSize \leftarrow header.getPayloadSize()$;
                if payloadSize not null or zero then
                    $nextPkt \leftarrow ModifiedPkt(IP1, IP2,$
                                        $pSize, timestamp)$;
                    $filteredPkts.add(nextPkt)$;
                end if
            end if
        end if
    end for
    return $filteredPackets$;
end procedure
```

Conversation Creation Module: The output of the Packet Filtering module is fed as input to the conversation creation module. This module creates a list of conversations by aggregating packets received from the previous module. Each conversation is identified by the binary tuple `<IP1,IP2>` and an initial `FLOWGAP` value. The initial `FLOWGAP` is used to create conversations: while iterating through packets, if a packet is encountered which belongs to the IP pair of the conversation and whose time-stamp lies within `FLOWGAP` time from the beginning or end of the conversation, the packet is added to the conversation and the attributes of the conversation are modified accordingly. This is explained as follows in Algorithm 2.

Algorithm 2 Conversation Creation Module

```
procedure CREATECONVERSATIONS(filteredPackets)
    $ArrayList < Conversation > initConvList$;
    $ArrayList < PacketGroup > pgList$;
    $pgList \leftarrow filteredPkts.groupPktsByIPpair()$;
    for PacketGroup pg in pgList do
        sort packets in $pg$ by $timestamp$;
        $nextConv \leftarrow Conversation(NULL)$;
        for Packet p in pg do
            if p.timestamp between
                (nextConv.start - FLOWGAP) &&
                (nextConv.end + FLOWGAP) then
                    $nextConv.addPacket(p)$;
            else
                    $nextConv \leftarrow Conversation(p)$;
                    $initConvList.add(nextConv)$;
            end if
        end for
    end for
    return $initConvList$;
end procedure
```

Conversation Aggregation Module: The conversations created in the creation module are aggregated for a higher `FLOWGAP` value as desired by a network administrator. Here, the network administrator is given the flexibility to mine data for the time-period desired by him- say 2 hours, 24 hours etc., thus giving him visibility into the network logs as desired by him. Such flexibility is especially valuable for bots which are *extremely stealthy* in their communication patterns and exchange as low as a few packets every few hours. From the dataset obtained by the authors of [24], we observed that the ZeuS botnet demonstrated such behavior. Such behavior demands a special attention and evaluation, and thus we do

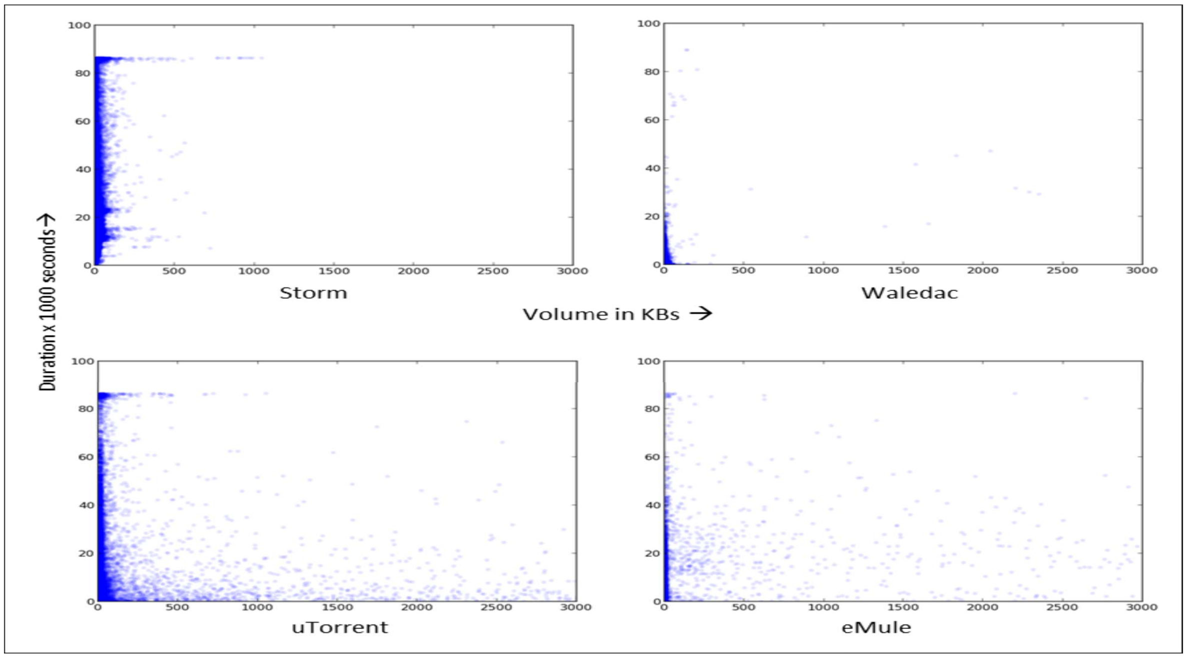

Fig. 2. Comparion of network traces of Storm, Waledac, uTorrent and eMule

not include ZeuS in our present evaluation. For this evaluation, the value being used is 1 hour.
The resultant conversations are then used to train our classification model. The attributes of each conversation which are analyzed are: Number of packets, Conversation volume (summation of payload lengths), Conversation duration and the Median value of Inter-arrival time of packets in the conversation. The reasons behind choosing these features have already been explained in the previous section.
The median of Inter-arrival time of packets was observed to be a better metric than the mean because PeerShark aggregates several conversations into a single conversation as per the `FLOWGAP` value supplied. In such a scenario, it is quite possible that conversation A and conversation B get merged into a single conversation while the last packet of conversation A and first packet of conversation B occur several minutes (or even hours) apart. This will skew the mean value, and the use of median value was found to be more suitable from our experiments.
The conversation aggregation module is explained in Algorithm 3.

Classification Module: The Classification module uses supervised machine learning algorithms (using Weka [34]) for training its model and classifying the test data. To validate our approach, models were built using a number of algorithms, namely Bayesian networks [35], Decision trees and Boosted REP trees.

Figure 2 provides a comparison of data of P2P botnets and applications for one hour FLOWGAP duration in the form of scatter plots. Each point on the plot denotes a conversation, with the intensity of the color denoting higher number of conversations at that point. The X axis has the Volume (in Kilobytes), while Duration (in thousands of seconds) is plotted on the Y axis. The data (for each application) corresponds to a 24 hour period, which means 86,400 seconds. Hence 86.4 forms the upper limit on the Y axis.

Algorithm 3 Conversation Aggregation Module

```
1: procedure AGGCONV(initConvList, FLOWGAP)
2:     $ArrayList < Conversation > finalConvList;$
3:     $ArrayList < ConversationGroup > cgList;$
4:     $cgList \leftarrow initConvList.groupConvByIPpair();$
5:     for ConversationGroup cg in cgList do
6:         sort conversations in cg by timestamp;
7:         $nextConv \leftarrow Conversation(NULL);$
8:         if c.timestamp between
9:             (nextConv.start - FLOWGAP) &&
10:            (nextConv.end + FLOWGAP) then
11:                $nextConv.addConv(c);$
12:        else
13:                $nextConv \leftarrow Conversation(c);$
14:                $finalConvList.add(nextConv);$
15:        end if
16:    end for
17:    return $finalConvList;$
18: end procedure
```

From the scatter plots, it is evident that bots tend to 'lie low' and remain stealthy. Their traffic is low-volume and high-duration, with only very few conversations being high in volume. In clear contrast, most conversations seen in benign P2P traffic (eMule and uTorrent) do not exhibit the trend of low-volume and high-duration. Rather the points are widely spread over the entire length and breadth of the plot. The few high-duration conversations seen in benign P2P traffic can be attributed to the fact that this is dataset was generated at the University of Georgia [24] by continuously running P2P applications (with sharing, downloading, etc.) over several days. Since the applications were running continuously, high durations conversations are present. Such a pattern is not expected to be seen with an average user of the Internet.

VI. Evaluation & Results

PeerShark was evaluated using network trace datasets obtained from the University of Georgia [24]. Data from two P2P applications (eMule and uTorrent) and two P2P botnet applications (Waledac and Storm) were used for this work. As described in the previous section, network traces were parsed to create and then further aggregate 'conversations'. The data so obtained was labeled to create a 'labeled training dataset' for each application. It is important to note that in the traces of Storm and Waledac, the number of known 'malicious hosts' are 13 and 3 respectively. However, it is not known whether the other IP addresses seen in the network traces are benign or malicious[1]. Hence, for the sake of creating a 'ground truth' for our evaluation, we treat a conversation as 'malicious' even if either of the IPs (either source or destination) is known to be 'malicious'.

Further, to build machine learning models with a balanced dataset, 50,000 conversations of each application were used. In order to not over-estimate the accuracy of PeerShark, we validated our results using multiple machine learning algorithms with ten-fold cross-validation. Due to space limitations, we present our evaluation only with three algorithms- Decision trees, Boosted REP trees and Bayesian Networks.

Decision trees are simple to train and fast algorithms. However, they tend to create complex tree structures and over-fit the data. Although this gives high accuracy on the training data and even over the test data when tested in a controlled lab environment, such botnet detection models may not generalize to a real-world scenario. Hence, apart from using the J48 implementation of Decision trees in Weka, we also used the implementation of Reduced-error pruning trees, a.k.a. REP trees, and limited the maximum depth of the tree to 8. Although we sacrifice on the training accuracy by limiting the maximum depth, this helps in obtaining a generalized model. Boosting [36] was used with REP trees to increase the accuracy obtained from a single classifier. We used the AdaBoost meta-classifier of Weka, and set it to use 10 REP trees (with each tree limited to a maximum depth of 8).

As a choice for the third classifier, we use Bayesian networks [35] which are probabilistic graphical models that can identify relationships among variables of interest. The model stores dependencies among all variables and can handle the missing data and outliers quite well. It also provides insight into a problem domain by providing relationships between various features of data.

Table I gives the results for ten-fold cross validation performed with the J48 Decision trees, Boosted REP trees and Bayesian networks. TPR refers to the True Positive Rate (the fraction of true positives out of the total actual positives) per class, FPR refers to the False Positive Rate (the fraction of false positives out of the total actual negatives) per class, and AUROC refers to the Area under the ROC (Receiver Operator Characteristic) Curve per class, which is created by plotting the TPR versus FPR for that class.

[1]Personal communication: Babak Rahbarinia [24], November 2013

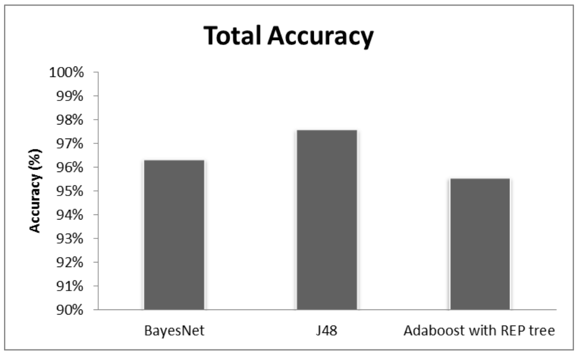

Fig. 3. Total Accuracy of each algorithm

As Table I shows, PeerShark could consistently give more than 97% accuracy in detection of the two P2P botnet applications, and at least 92% accuracy in the detection of the two P2P applications, with very low false positives. Figure 3 gives the total accuracy (weighted average of per-class accuracy) of the three classifiers. As an overall average, PeerShark's accuracy stands above 95%.

VII. Possible Evasions & Discussions

PeerShark is able to correctly detect and categorize P2P applications- whether malicious or benign- with high accuracy. Here we consider the limitations of PeerShark. PeerShark is presently designed with a multi-class classification approach, where the output will always be *one* of the four classes as in the training phase. Such an approach does not fare well with new or unknown P2P applications since they do not belong to either of the four classes. PeerShark could be extended to add an 'unknown' class label which will account for all P2P data that does not match the classification criteria of any of the other classes.

If peer A and peer B are engaged in P2P file sharing and both of them are also a part of a botnet, PeerShark will see their communications as a single 'conversation'. Because of an overlap between the botnet data and application data, PeerShark will not be able to correctly classify the kind of botnet or application running on peers A & B. But with regard to the present day botnets, such a scenario may not happen frequently. This can be explained as follows: in most cases, bots are covertly installed on the system of an unsuspecting victim (through drive-by downloads, infected USB sticks, etc.). Considering two such random victims to be A and B, it is not very likely that the actual owners of A and B will also be engaged in P2P file-sharing with each other. However, we must argue on the case of 'smarter', futuristic bots which may try to evade the detection mechanism of PeerShark. Botmasters could configure their bots to engage in occasional file-sharing activity amongst each other in a regular P2P network (like eMule, uTorrent etc.). With such benign-like activity, bots will no longer display a 'low-volume' behavior. Since such a behavior is one of the main detection criterion used by PeerShark, PeerShark is likely to mis-classify such bots. But, since occasional file-sharing by bots involves network

TABLE I. P2P BOTNET DETECTION WITH BAYESNET, J48 AND BOOSTED REP TREES

	Bayesian Network			J48 Decision Trees			Adaboost with REP trees		
	TPR	FPR	AUROC	TPR	FPR	AUROC	TPR	FPR	AUROC
eMule	0.929	0.012	0.996	0.964	0.012	0.987	0.93	0.021	0.993
Storm	0.988	0.009	0.999	0.986	0.003	0.996	0.979	0.004	0.999
Waledac	0.989	0.01	0.999	0.988	0.005	0.995	0.97	0.009	0.998
uTorrent	0.947	0.019	0.996	0.965	0.012	0.989	0.943	0.025	0.994

bandwidth usage (and, say, accompanying monetary charges), such an activity has the likelihood of getting noticed by the owner of the system or a network administrator, and is thus fraught with risks for the bot-master. Nonetheless, we admit that it is possible for bot-masters to design smarter bots which mimic benign-like behavior and/or add noise (or randomness) to their communication patterns, and thus evade the present detection mechanism of PeerShark.

Another similar approach by which bots could evade PeerShark is by changing their 'high-duration conversation' behavior. That is, a bot-master may design his botnet in such a way that bots do not maintain proper connectivity with other bot-peers. Again, since 'high-duration conversation' behavior of P2P bots lies at the heart of the detection algorithm of PeerShark, this technique can surely defeat its detection mechanism. But this will come at a very heavy cost for the bot-masters. If the bots do not maintain connectivity with their peers, propagating Command & Control messages and/or updates will involve much higher latencies. And this will result into the usability (and thus the profitability) of the botnet itself comes into question.

Furthermore, assume the case of a peer A which is engaged in P2P file sharing with a benign peer B, but is also covertly a part of a botnet and is engaged in exchanging Command & Control with a malicious peer C. PeerShark will see these as two conversations, namely A to B and A to C. Since PeerShark regards a conversations as 'malicious' even if either of the IPs (either source or destination) is malicious, A to C is identified as 'malicious' without hesitation. But since the conversation between A and B also involves one malicious peer (namely A), this conversation will also be tagged as malicious. Although it is a limitation on the part of PeerShark to regard that peer B is engaged in a malicious conversation, it is not a serious shortcoming since peer B is, as a matter of fact, conversing with a peer which has been compromised, and thus it runs high chances of being infected in future. Thus, raising an alarm for peer B (apart from A and C) is not completely unwarranted.

Finally, as described in the Packet Filtering Module (Section V), PeerShark discards all packets having a zero payload. This was necessary to remove corrupted packets and sanitize the network traces obtained from authors in [24]. However, such an approach has an inherent drawback of dropping all legitimate packets with zero payloag- such as TCP connection establishment (SYN) packets. Further, being protocol-oblivious, PeerShark does not differentiate between TCP and UDP packets. This can be exploited by an active adversary who may use zero payload TCP packets (SYN or ACK packets) to exchange simple commands between bots. Some solutions for the limitations are proposed in the next section.

VIII. CONCLUSION & FUTURE WORK

In this paper, we presented PeerShark, a novel approach which extends the previous efforts for P2P botnet detection. The main contribution of PeerShark is the detailed evaluation of a conversation-based approach which was clearly shown to be advantageous over traditional flow-based approaches. Using the simple and novel conversation-based features, PeerShark could also correctly categorize different kinds of P2P applications- whether malicious or benign- with high accuracy.

However, the accuracy obtained with classification of benign P2P applications is relatively lower as compared to the accuracy of detection of P2P botnets. Accurate detection of these applications is an important area of future work. The present evaluation was also limited to two benign P2P applications. We plan to add greater variety of benign P2P applications to our dataset and experiment with several other features which can help in correctly categorizing P2P traffic- such as the control (or management) traffic information of P2P applications [24].

A few limitations of the present implementation of PeerShark were discussed in the previous section. Most of these limitations are tethered to the fact that PeerShark's present approach gives a *bird's eye-view* of the conversations happening in the network. Being flow-oblivious (i.e., port and protocol-oblivious), many lower-level details (such as the Transport layer protocol) are neglected. We plan to address these limitations by modifying PeerShark to begin with a flow-aware approach followed by clustering of similar flows (clustered on the basis of, say, inter-arrival time, average payload length, etc.). Then, the flow-oblivious conversation aggregation will be performed for all the flows belonging to a particular cluster only. If more than one P2P application is running between two peers (either benign or malicious), the flows from different applications are expected to get separated into different clusters (because of the different nature of flows seen in different application). Work on this approach is already in progress, and we expect it to address the limitations and possible evasions addressed previously.

As a part of work-in-progress, we are extending our approach with a distributed model for data collection where data collectors sit closer to the nodes inside the network (say at wi-fi access points). This will give greater visibility of the network traffic which occurs over LAN and never touches the backbone router of an enterprise. Such insight can be very valuable for detecting P2P bots inside a network perimeter which maintain connectivity with each other over LAN in a P2P fashion, but limit the conversation with the outside world via one or two designated peers only.

Acknowledgment

This work was supported by Grant number 12(13)/2012-ESD for scientific research under Cyber Security area from the Department of Information Technology, Govt. of India, New Delhi, India.

References

[1] "Ipoque internet study 2008/2009," http://www.ipoque.com/en/resources/internet-studies, accessed on 4 January 2014.

[2] "Sandvine global internet phenomena report 2013," https://www.sandvine.com/trends/global-internet-phenomena/, accessed on 4 January 2014.

[3] R.-A. Shang, Y.-C. Chen, and P.-C. Chen, "Ethical decisions about sharing music files in the p2p environment," *Journal of Business Ethics*, vol. 80, no. 2, pp. 349–365, 2008.

[4] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson, "Privacy-preserving p2p data sharing with oneswarm," in *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4. ACM, 2010, pp. 111–122.

[5] J.-T. Kim, H.-K. Park, and E.-H. Paik, "Security issues in peer-to-peer systems," in *Advanced Communication Technology, 2005, ICACT 2005. The 7th International Conference on*, vol. 2. IEEE, 2005, pp. 1059–1063.

[6] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Computing Surveys (CSUR)*, vol. 36, no. 4, pp. 335–371, 2004.

[7] "Bittorrent," http://www.bittorrent.com/, accessed on 17 December 2013.

[8] "Seti @ home," http://setiathome.berkeley.edu/, accessed on 22 January 2014.

[9] "Livestation," http://www.livestation.com/, accessed on 22 January 2014.

[10] "Skype," http://www.skype.com/en/, accessed on 17 December 2013.

[11] J. Buford, H. Yu, and E. K. Lua, *P2P networking and applications*. Morgan Kaufmann, 2009.

[12] P. Narang, J. M. Reddy, and C. Hota, "Feature selection for detection of peer-to-peer botnet traffic," in *Proceedings of the 6th ACM India Computing Convention*, 2013, pp. 16:1–16:9.

[13] C. Kanich, N. Weaver, D. McCoy, T. Halvorson, C. Kreibich, K. Levchenko, V. Paxson, G. M. Voelker, and S. Savage, "Show me the money: Characterizing spam-advertised revenue." in *USENIX Security Symposium*, 2011, pp. 15–15.

[14] D. Dittrich and S. Dietrich, "P2p as botnet command and control: a deeper insight," in *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*. IEEE, 2008, pp. 41–48.

[15] C. Rossow, D. Andriesse, T. Werner, B. Stone-Gross, D. Plohmann, C. J. Dietrich, and H. Bos, "Sok: P2pwned-modeling and evaluating the resilience of peer-to-peer botnets," in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 97–111.

[16] D. Andriesse, C. Rossow, B. Stone-Gross, D. Plohmann, and H. Bos, "Highly resilient peer-to-peer botnets are here: An analysis of gameover zeus," in *Malicious and Unwanted Software:" The Americas"(MALWARE), 2013 8th International Conference on*. IEEE, 2013, pp. 116–123.

[17] "Hacked fridge part of botnet attack," http://asia.cnet.com/hacked-fridge-part-of-botnet-that-sent-750000-spam-emails-62223486.htm, Accessed on 12th February 2014.

[18] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of p2p traffic using application signatures," in *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004, pp. 512–521.

[19] J. Li, S. Zhang, Y. Lu, and J. Yan, "Real-time p2p traffic identification," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*. IEEE, 2008, pp. 1–5.

[20] M. Iliofotou, H.-c. Kim, M. Faloutsos, M. Mitzenmacher, P. Pappu, and G. Varghese, "Graph-based p2p traffic classification at the internet backbone," in *INFOCOM Workshops 2009, IEEE*. IEEE, 2009, pp. 1–6.

[21] G. Gu, R. Perdisci, J. Zhang, W. Lee *et al.*, "Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection." in *USENIX Security Symposium*, 2008, pp. 139–154.

[22] J. François, S. Wang, R. State, and T. Engel, "Bottrack: Tracking botnets using netflow and pagerank," in *Proceedings of the 10th International IFIP TC 6 Conference on Networking - Volume Part I*, 2011, pp. 1–14.

[23] H. Hang, X. Wei, M. Faloutsos, and T. Eliassi-Rad, "Entelecheia: Detecting p2p botnets in their waiting stage," in *IFIP Networking Conference, 2013*, 2013, pp. 1–9.

[24] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li, "Peerrush: Mining for unwanted p2p traffic," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 7967, pp. 62–82.

[25] J. Zhang, R. Perdisci, W. Lee, X. Luo, and U. Sarfraz, "Building a scalable system for stealthy p2p-botnet detection," *Information Forensics and Security, IEEE Transactions on*, vol. 9, no. 1, pp. 27–38, 2014.

[26] R. Schoof and R. Koning, *Detecting peer-to-peer botnets*, University of Amsterdam, 2007, technical report.

[27] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov, "Botgrep: Finding p2p bots with structured graph analysis." in *USENIX Security Symposium*, 2010, pp. 95–110.

[28] J. Zhang, R. Perdisci, W. Lee, U. Sarfraz, and X. Luo, "Detecting stealthy p2p botnets using statistical traffic fingerprints," in *Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*. IEEE, 2011, pp. 121–132.

[29] T.-F. Yen and M. K. Reiter, "Are your hosts trading or plotting? telling p2p file-sharing and bots apart," in *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*. IEEE, 2010, pp. 241–252.

[30] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "Blinc: multilevel traffic classification in the dark," in *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4. ACM, 2005, pp. 229–240.

[31] T. Karagiannis, A. Broido, M. Faloutsos *et al.*, "Transport layer identification of p2p traffic," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM, 2004, pp. 121–134.

[32] S. Zhang, "Conversation-based p2p botnet detection with decision fusion," Master's thesis, Fredericton: University of New Brunswick, 2013.

[33] L. Li, S. Mathur, and B. Coskun, "Gangs of the internet: Towards automatic discovery of peer-to-peer communities," in *Communications and Network Security (CNS), 2013 IEEE Conference on*. IEEE, 2013, pp. 64–72.

[34] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[35] R. R. Bouckaert, "Bayesian network classifiers in weka for version 3-5-7," *Artificial Intelligence Tools*, vol. 11, no. 3, pp. 369–387, 2008.

[36] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Thirteenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann, 1996, pp. 148–156.