PeerHunter: Detecting Peer-to-Peer Botnets through Community Behavior Analysis

Di Zhuang*, J. Morris Chang†

Department of Electrical Engineering, University of South Florida, Tampa, Florida 33620

Email: *dizhuang@mail.usf.edu, †chang5@usf.edu

Abstract—**Peer-to-peer (P2P) botnets have become one of the major threats in network security for serving as the infrastructure that responsible for various of cyber-crimes. Though a few existing work claimed to detect traditional botnets effectively, the problem of detecting P2P botnets involves more challenges. In this paper, we present PeerHunter, a community behavior analysis based method, which is capable of detecting botnets that communicate via a P2P structure. PeerHunter starts from a P2P hosts detection component. Then, it uses mutual contacts as the main feature to cluster bots into communities. Finally, it uses community behavior analysis to detect potential botnet communities and further identify bot candidates. Through extensive experiments with real and simulated network traces, PeerHunter can achieve very high detection rate and low false positives.**

I. INTRODUCTION

A botnet is a set of compromised machines controlled by botmaster through a command and control (C&C) channel. Botnets may have different communication architectures. Classical botnets were known to use a centralized architecture, which has a single point of failure. Peer-to-peer (P2P) network happens to be modeled as a distributed architecture, where even though a certain number of peers fail to function properly, the whole network is not compromised. In this case, the most of recent botnets (e.g. Storm, Waledac, ZeroAccess, Sality and Kelihos) attempt to build on P2P network, and P2P botnets have proven to be highly resilient even after a certain number of bots being identified or taken-down [1]. P2P botnets provide a fundamental infrastructure for various cyber-crimes [2], such as distributed denial-of-service (DDoS), email spam, click fraud, etc. Therefore, detecting P2P botnets effectively is rather important for cyber security.

However, designing an effective P2P botnets detection system is extremely hard, due to several challenges. First, botnets tend to act stealthily [3] and spend most of their time in the waiting stage before actually performing any malicious activities [4], [5]. Second, botnets tend to encrypt C&C channels, which makes deep-packet-inspection (DPI) based methods fail to work. Third, botnets can randomize their communication patterns dynamically without jeopardizing any primary functions [6]–[8], which makes statistical traffic signatures based methods unable to work.

In this paper, we propose PeerHunter, a novel community behavior analysis based P2P botnet detection system, which could address all the challenges above. We consider a botnet community as a group of compromised machines that communicate with each other or connect to the botmaster through the same C&C channel, are controlled by the same attacker, and

aim to perform similar malicious activities. Due to the dynamic changes of communication behaviors of P2P botnets [9], it would be extremely hard to identify a single bot. However, bots within the same P2P botnet always work together as a community, thus, have distinct community behaviors to be identified. PeerHunter begins with a general P2P hosts detection component. Then, it builds a mutual contact graph (MCG) of the detected P2P hosts. Afterwards, it applies a community detection method on the MCG, which uses mutual contacts [10] as the main feature of P2P botnets to cluster bots within the same botnet together, and separate bots and legitimate hosts or different types of bots into different communities. Finally, it uses *destination diversity* and *mutual contacts* as the natural features to capture the "P2P behavior" and "botnet behavior" respectively of each P2P botnet community, and further identify all the P2P botnets.

Specifically, PeerHunter is capable of detecting P2P bots with the following challenges and assumptions: (a) botnets are in their waiting stage, which means there is no clear malicious activity can be observed [4]; (b) the C&C channel has been encrypted, so that no deep-packet-inspection (DPI) can be deployed; (c) no bot-blacklist or "seeds" information [10] are available; (d) none statistical traffic patterns [9] known in advance; and (e) could be deployed at network boundary (e.g. gateway), thus, do not require to monitor individual host.

In the experiments, we mixed a real network dataset from a public traffic archive [11] with several P2P botnet datasets and legitimate P2P network datasets [12]. To make the experimental evaluation as unbiased and challenging as possible, we propose a network traces sampling and mixing method to generate synthetic data. We tested our system with 24 synthetic experimental datasets that each contains 10,000 internal hosts. We implemented our P2P hosts detection component using a Map-Reduce framework, which could dramatically reduce the number of hosts subject to analysis by 99.03% and retained all the P2P hosts in our experiments. The Map-Reduce design and implementation of our system could be deployed on popular cloud-computing platforms (e.g. amazon EC2), which ensures the scalability of our system to deal with a big data. With the best parameter settings, our system achieved 100% detection rate with none false positives.

The rest of paper is organized as follows: Section II presents the related works. Section III explains the motivation and details of the features applied in our system. Section IV describes the system design and implementation details about PeerHunter. Section V presents the experimental evaluation of PeerHunter. Section VI makes the conclusion.

II. RELATED WORK

A few methods attempt to detect P2P botnets have been proposed [3], [5], [7]–[10], [12]–[18]. Host-level methods have been proposed [17]. However, in host-level methods, all the hosts are required to be monitored individually, which is impractical in real network environments. Network-level methods can be roughly divided into (a) traffic signature based methods, and (b) group/community behavior based methods.

Traffic signature based methods [3], [5], [7], [8], [12], [14]–[16] rely on a variety of statistical traffic signatures. For instance, Entelecheia [5] uses traffic signatures to identify a group of P2P bots in a super-flow graph. PeerRush [12] is a signature based P2P traffic categorization system, which can distinguish traffic from different P2P applications, including P2P botnet. Nevertheless, these methods suffer from botnets that have dynamic statistical traffic patterns. Traffic size statistical features can be randomized or modified, since they are only based on the communication protocol design of a botnet. Traffic temporal statistical features can also act dynamically without jeopardizing any primary functions of a botnet.

Group or community behavior based methods [9], [10] consider the behavior patterns of a group of bots within the same P2P botnet community. For instance, Coskun et al. [10] developed a P2P botnets detection approach that start from building a mutual contact graph of the whole network, then attempt to utilize "seeds" (known bots) to identify the rest of bots within the same botnet. However, most of the time, it is hard to have a "seed" in advance. Yan et al. [9] proposed a group-level behavior analysis based P2P botnets detection method. However, they only considered to use statistical traffic features to cluster P2P hosts, which is subject to P2P botnets that have dynamic or randomized traffic patterns. Besides, their method cannot cope with unknown P2P botnets, which is the common case in botnet detection [2], because of relying on supervised classification methods (e.g. SVM).

III. BACKGROUND AND MOTIVATION

To demonstrate the features discussed in this section, we conducted some preliminary experiments using dataset shown in Table III and Table IV.Table I shows the notations and descriptions, and Table II shows the measurements of features.

TABLE I: Notations and Descriptions

Notations	Descriptions
MNF	the management network flows
AVGDD	the average # of distinct /16 MNF dstIP prefixes
AVGDDR	the average destination diversity ratio
AVGMC	the average # of mutual contacts between a pair of hosts
AVGMCR	the average mutual contact ratio

A. P2P Network Characteristics

Due to the decentralized nature of P2P network, a P2P host usually communicates with peers that distributed in a large range of distinct physical networks, which results in the destination diversity (DD) characteristic [12] of P2P management network flow (MNFs). MNF is the network flow for maintaining the function and structure of the P2P network.

TABLE II: Measurements of Features

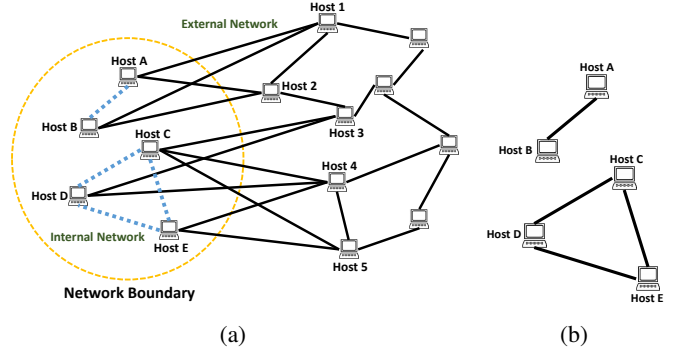| Trace	AVGDD	AVGDDR	AVGMC	AVGMCR
eMule	8,349	17.6%	3,380	3.7%
FrostWire	11,420	15.2%	7,134	4.5%
uTorrent	17,160	8.7%	13,888	3.5%
Vuze	12,983	10.1%	18,850	7.9%
Storm	7,760	25.1%	14,684	30.2%
Waledac	6,038	46.0%	7,099	37.0%
Sality	9,803	9.5%	72,495	53.2%
Kelihos	305	97.4%	310	98.2%
ZeroAccess	246	96.9%	254	100.0%

Fig. 1: Illustration of network (a) and its mutual contact graph (b).

The P2P network flow mentioned in this section and the rest only refers to P2P MNF.

We use DD as our main feature to detect P2P network flows and further identify P2P hosts. In addition, we use the number of distinct /16 IP prefixes of each host's network flows, rather than BGP prefix used in [3] to approximate DD feature of each P2P host/network flow. /16 IP prefix is a good approximation of network boundaries. For instance, it is very likely that two IP addresses with different /16 IP prefixes belong to two distinct physical networks. This is also supported by Table II, which shows the network flows in a P2P network spreading across a large number of distinct physical networks according to the number of /16 IP prefixes.

B. Mutual Contacts

The mutual contacts (MC) between a pair of hosts is a set of shared contacts between the corresponding pair of hosts. Consider the network illustrated in Fig. 1a which contains an internal network (Host A, B, C, D and E) and an external network (Host 1, 2, 3, 4 and 5). A link between a pair of hosts means they have at least one connection. In Fig. 1a, Host 1, 2 are the mutual contacts shared by Host A, B.

Mutual contacts is the natural characteristic of P2P botnet. Compared with legitimate hosts, a pair of bots within the same P2P botnet has a much higher probability to share a mutual contact [10]. Because bots within the same P2P botnet tend to receive or search for the same C&C messages from the same set of botmasters (peers) [19]. Moreover, in order to prevent peers from churning in a P2P botnet, botmaster has to check each bot periodically, which results in a convergence of contacts among peers within the same botnet [3]. However, since bots from different botnets are controlled by different botmasters, they won't share many mutual contacts. Legitimate host pairs may have a small set of mutual contacts, since nearly all hosts communicate with several extremely popular servers,

such as google.com, facebook.com [10]. Furthermore, the host pairs running the same P2P applications may also result in a decent ratio of mutual contacts, if they are accessing the same resource from the same set of peers by coincidence. However, in reality, legitimate P2P hosts with different purposes will not search for the same set of peers. Thus, we can use mutual contacts as a feature to cluster the bots within the same botnet.

The basic idea is to build a mutual contacts graph (MCG) as shown in Fig. 1, where Host A, B are linked together in Fig. 1b, since they have mutual contacts Host 1, 2 in Fig. 1a. Similarly, Host C, D, E are linked to each other in Fig. 1b, since every pair of them share at least one mutual contacts in Fig. 1a. More details about MCG is discussed in Section IV-B.

C. Community Behavior Analysis

We consider three types of community behaviors: (a) flow statistical feature, (b) numerical community feature and (c) structural community feature.

1) Flow Statistical Feature: Botnet detection methods using flow statistical features, have been widely discussed [7], [8], [20], [21]. We use the statistical features of P2P MNFs, which are usually generated through the same P2P protocol for a specific P2P application, and some of the statistical patterns of P2P MNFs fully depend on protocols. However, the other network flows, such as data-transfer flows, are usually situation-dependant, which vary a lot even in the same P2P network. In this work, we use the ingoing and outgoing bytes-per-packets (BPP) of network flows in one P2P network as the community flow statistical feature.

2) Numerical Community Feature: We consider two types of features: average destination diversity ratio (AVGDDR) and average mutual contacts ratio (AVGMCR).

Average Destination Diversity Ratio: This captures the "P2P behavior" of P2P botnet communities. The destination diversity (DD) of a P2P host is the number of distinct /16 IP prefixes of each host's network flows. The destination diversity ratio (DDR) of each host is its DD divided by the total number of distinct destination IPs of its network flows.

Due to the decentralized nature of P2P networks, P2P network flows tend to have higher DDR than non-P2P network flows. Furthermore, network flows from P2P botnet communities usually have higher average DDR (AVGDDR) than network flows from legitimate network communities. Network flows from bots within the same botnet tend to have similar DDR, since those bots are usually controlled by machines, rather than humans. However, the destinations of legitimate P2P network flows are usually user-dependant, which result in their DDR varying greatly from user to user. Besides, our botnet community detection method aims to cluster bots within the same botnets together, rather than clustering the same legitimate P2P hosts together. Legitimate communities might contain both P2P hosts and non-P2P hosts, leading to lower AVGDDR than botnet communities.

Table II shows the number of distinct destination IP /16 prefixes in MNFs of each type of P2P host, where both legitimate hosts and bots spread across a large number of distinct networks. However, most of the botnets communities have higher AVGDDR than legitimate communities, except Sality. We could combine the next feature to identify Sality.

Average Mutual Contacts Ratio: This captures the "botnet behavior" of P2P botnet communities. The mutual contacts ratio (MCR) between a pair of hosts is the number of mutual contacts between them, divided by the number of total distinct contacts of them. This idea is based on three observations: (a) P2P botnet communities are usually formed by at least two bots, otherwise they cannot act as a group, (b) MCR between a pair of bots within the same botnet is much higher than that between a pair of legitimate hosts or bots from different botnets, and (c) each pair of bots within the same botnet has similar MCR. Thus, we consider the average MCR (AVGMCR) among all pairs of hosts within one network community as another numerical community feature.

Table II shows the average number of mutual contacts between a pair of hosts within the same community, where both botnets and certain legitimate network communities have a considerable number of mutual contacts. That is because those legitimate communities have much more contacts than botnets. However, botnets has much higher AVGMCR.

3) Structural Community Feature: This captures the structural characteristics of a botnet. The basic idea is that, every pair of bots within the same botnet tends to have a considerable number or ratio of mutual contacts. Therefore, if we consider each hosts as a vertex and link an edge between a pair of hosts if they have a certain amount or ratio of mutual contacts, the bots within the same botnet tend to form certain complete graphes (cliques). On the contrary, the contacts of different legitimate hosts usually tend to diverge into different physical networks. Thus, the probability that legitimate communities form certain cliques is relatively low. Then, we can consider P2P botnets detection as a clique detection problem, which detects cliques from a given network with certain requirements. However, since clique detection problem is NP-complete, we cannot just apply such method to detect botnets. Therefore, we use all three botnet community behaviors.

IV. SYSTEM DESIGN

PeerHunter has three components, that work synergistically to (a) detect P2P hosts, (b) construct mutual contact graph, and (c) detect bots. Fig. 2 illustrates the framework of PeerHunter.

A. P2P Hosts Detection

This component is responsible for detecting hosts engaged in P2P communications. The input is a 5-tuple network flow $[ip_{src}, ip_{dst}, proto, bpp_{out}, bpp_{in}]$, where ip_{src} is source IP, ip_{dst} is destination IP, $proto$ is tcp or udp, and bpp_{out} and bpp_{in} are outgoing and ingoing BPP of network flows. First, we cluster all network flows $F = \{f_1, f_2, \ldots, f_k\}$ based on the 4-tuple $[ip_{src}, proto, bpp_{out}, bpp_{in}]$ into flow clusters $FC = \{FC_1, FC_2, \ldots, FC_m\}$. Then, we calculate the number of distinct /16 prefixes of ip_{dst} (destination diversity) associated with each flow cluster, $dd_i = DD(FC_i)$. If dd_i is greater than a pre-defined threshold Θ_{dd}, we consider FC_i as a P2P MNF cluster, and the corresponding source hosts as P2P hosts.

Fig. 2: System Overview

Algorithm 1 P2P Hosts Detection

1: **function** MAP($[ip_{src}, ip_{dst}, proto, bpp_{out}, bpp_{in}]$)
2:     $Key \leftarrow [ip_{src}, proto, bpp_{out}, bpp_{in}]$
3:     $Value \leftarrow ip_{dst}$
4:     **output** $(Key, Value)$
5: **end function**
6: **function** REDUCE($Key, Value[\ ]$)
7:     $k \leftarrow Key$
8:     $dd_k = \varnothing$
9:     **for** $v \in Value[\ ]$ **do**
10:         $dd_k \leftarrow dd_k \cup \{v\}$
11:     **end for**
12:     **if** $|dd_k| \geq \Theta_{dd}$ **then**
13:         **for** $v \in Value[\ ]$ **do**
14:             **output** (k, v)
15:         **end for**
16:     **end if**
17: **end function**

As shown in Algorithm 1, we design this component using a MapReduce framework [22]. For a mapper, the input is a set of 5-tuple network flows, and the output is a set of key-value pairs. For a reducer, the input is the set of key-values pairs. Then, the reducer aggregates all values with the same key to calculate the DD of each flow cluster, and finally output the detected P2P MNFs based on Θ_{dd}.

B. Mutual Contact Graph Extraction

This component is responsible for extracting mutual contact graph (MCG) through mutual contacts. The input is a list of detected P2P hosts, $H=\{h_1,\ h_2,\ \ldots,\ h_{|H|}\}$, and their corresponding P2P MNFs, $F=\{f_1^1,\ f_2^1,\ \ldots,\ f_{n_1}^1,\ f_1^2,\ f_2^2,\ \ldots,\ f_{n_2}^2,\ \ldots,\ f_1^{|H|},\ f_2^{|H|},\ \ldots,\ f_{n_{|H|}}^{|H|}\}$, where f_i^j is flow i from h_j. The output is a MCG, $G_{mc} = (V, E)$, where each vertex $v_i \in V$ contains a DDR score ddr_i of h_i's MNFs, and each edge $e_{ij} \in E$ contains a nonnegative MCR weight mcr_{ij} between h_i and h_j. Algorithm 2 shows the main steps in this component.

First, for each host h_i, we generate a contact set C_i, that contains all the destination IPs in its MNFs. Each host h_i also contains a flow statistical pattern set S_i, which contains all $[proto, bpp_{out}, bpp_{in}]$ 3-tuple in its MNFs. Let $DD(C_i)$ be the set of distinct /16 prefixes of all the IPs in C_i. Then, ddr_i and mcr_{ij} can be calculated as below.

$$ddr_i = \frac{\|DD(C_i)\|}{\|C_i\|} \qquad mcr_{ij} = \frac{C_i \cap C_j}{C_i \cup C_j} \qquad (1)$$

Furthermore, as discussed in Section III-C1, MNFs from different hosts within the same network communities should have similar statistical patterns. Thus, for each pair of input hosts, say h_i and h_j, we calculate the intersection between S_i

and S_j. If $S_i \cap S_j = \varnothing$, then there is no edge between h_i and h_j in MCG. Otherwise, they share at least one MNF statistical pattern, and we calculate mcr_{ij} as shown in (1). Let Θ_{mcr} be a pre-defined threshold. Then, if $mcr_{ij} > \theta_{mcr}$, there is an edge between h_i and h_j, with weight mcr_{ij}. Otherwise, there is no edge between h_i and h_j ($mcr_{ij} = 0$).

Algorithm 2 Mutual Contact Graph Extraction

input: H, F, Θ_{mcr}
output: $G_{mc} = (V, E)$
1: $E = \varnothing, V = \varnothing$
2: **for** $h_i \in H$ **do**
3:     $C_i = \varnothing$
4:     $S_i = \varnothing$
5: **end for**
6: **for** $f_i^j \in F$ **do**
7:     $C_j \leftarrow C_j \cup \{ip_{dst}\}$
8:     $S_j \leftarrow S_j \cup \{[proto, bpp_{out}, bpp_{in}]\}$
9: **end for**
10: **for** $h_i \in H$ **do**
11:     $ddr_i \leftarrow \frac{\|DD(C_i)\|}{\|C_i\|}$
12:     $vertex\ v_i \leftarrow <ddr_i>$
13:     $V \leftarrow V \cup \{v_i\}$
14: **end for**
15: **for** $\forall\ h_i, h_j \in H$ and $i < j$ **do**
16:     **if** $S_i \cap S_j \neq \varnothing$ **then**
17:         $mcr_{ij} \leftarrow \frac{C_i \cap C_j}{C_i \cup C_j}.$
18:         **if** $mcr_{ij} > \Theta_{mcr}$ **then**
19:             $edge\ e_{ij} \leftarrow <mcr_{ij}>$
20:             $E \leftarrow E \cup \{e_{ij}\}$
21:         **end if**
22:     **end if**
23: **end for**
24: **return** $G_{mc} = (V, E)$

C. P2P Botnet Detection

This component is responsible for detecting P2P bots from given MCG. First, we cluster bots into communities. Then, we detect botnet communities using numerical community behavior analysis. In the end, we perform structural community behavior analysis to further identify or verify each bot candidates. Algorithm 3 shows the main steps.

1) Community Detection: In a MCG $G_{mc} = (V, E)$, $\forall$ $e_{ij} \in E$, we have $mcr_{ij} \in [0, 1]$, where $mcr_{ij} = 1$ means all contacts of h_i and h_j are mutual contacts and $mcr_{ij} = 0$ means there is no mutual contacts between h_i and h_j. Furthermore, bots within the same botnet tend to have a large number/ratio of mutual contacts. Then, the bots clustering problem can be considered as a classical community detection problem. Various community detection methods have been

discussed in [23]. In this work, we utilize Louvain method, a modularity-based community detection algorithm [24], due to (a) its definition of a good community detection result (high density of weighted edges inside communities and low density of weighted edges between communities) is perfect-suited for our P2P botnet community detection problem; (b) it outperforms many other modularity methods in terms of computation time [24]; and (c) it can handle large network data sets (e.g. the analysis of a typical network of 2 million nodes only takes 2 minutes [24]).

Given $G_{mc} = (V, E)$ as input, Louvain method outputs a set of communities $Com = \{com_1, com_2, \ldots, com_{|Com|}\}$, where $com_i = (V_{com_i}, E_{com_i})$. V_{com_i} is a set of hosts in com_i. E_{com_i} is a set of edges, where $\forall e_{jk} \in E_{com_i}$, we have $e_{jk} \in E$ and $v_j, v_k \in V_{com_i}$.

2) Botnet Communities Detection: Given a set of communities Com, for each community $com_i \in Com$, we start from calculating $avgddr_i$ and $avgmcr_i$, as shown below.

$$avgddr_i = \frac{\sum_{v_j \in V_{com_i}} ddr_j}{\|V_{com_i}\|} \quad (2)$$

$$avgmcr_i = \frac{2 \times \sum_{\forall e_{jk} \in E_{com_i}} mcr_{jk}}{\|V_{com_i}\| \times (\|V_{com_i}\| - 1)} \quad (3)$$

We define two thresholds Θ_{avgddr} and Θ_{avgmcr}. Then, $\forall\ com_i \in Com$, if $avgddr_i \geq \Theta_{avgddr}$ and $avgmcr_i \geq \Theta_{avgmcr}$, we consider com_i as a botnet community.

3) Bot Candidates Detection: Recall from Section III-C3, the MCG of a botnet usually has a structure of one or several cliques. Therefore, we utilize a maximum clique detection method $CliqueDetection$ to further identify or verify each bot candidates from botnet communities. Each time it tries to detect one or several maximum cliques on the given MCG of botnet communities. If maximum clique (at least contains 3 vertices) has been found, we consider the hosts in that clique as bot candidates, remove those hosts from the original MCG, and run the maximum clique detection algorithm on the remaining MCG, until no more qualified maximum cliques to be found, then return the set of bot candidates.

V. EXPERIMENTAL EVALUATION

A. Experiment Setup

1) Experiment Environment: The experiments are conducted on one single PC with an 8 core Intel i7-4770 Processor, 32GB RAM, 400GB SSD and 4TB HHD, and on the 64-bit Ubuntu 14.04 LTS operating system.

2) Data Collection and Analysis Tool: The dataset contains three categories: (a) ordinary P2P network traces, (b) P2P botnets network traces, and (c) background network traces. In practice, all the network traces could be collected at a network boundary (e.g. firewall, gateway, etc.).

Ordinary P2P network traces (D_1)**:** We used the dataset obtained from the University of Georgia [12] as our ordinary P2P network traces, which collected the network traces of 4 different popular P2P applications for several weeks. There are 16 eMule hosts, 16 FrostWire hosts, 14 uTorrent hosts and 14

Algorithm 3 P2P Botnet Detection

input: G_{mc}, Θ_{avgddr}, Θ_{avgmcr}
output: S_{bot}
1: $S_{botnetCom} = \varnothing$, $S_{bot} = \varnothing$
2: $Com \leftarrow \textbf{Louvain}(G_{mc})$
3: **for** $com_i \in Com$ **do**
4: $\qquad avgddr_i \leftarrow \dfrac{\sum_{v_j \in V_{com_i}} ddr_j}{\|V_{com_i}\|}$
5: $\qquad avgmcr_i \leftarrow \dfrac{2 \times \sum_{\forall e_{jk} \in E_{com_i}} mcr_{jk}}{\|V_{com_i}\| \times (\|V_{com_i}\| - 1)}$
6: $\qquad$ **if** $avgddr_i \geq \Theta_{avgddr}$ and $avgmcr_i \geq \Theta_{avgmcr}$ **then**
7: $\qquad\qquad S_{botnetCom} \leftarrow S_{botnetCom} \cup \{com_i\}$
8: $\qquad$ **end if**
9: **end for**
10: **for** $com_i \in S_{botnetCom}$ **do**
11: $\qquad S_{bot} \leftarrow \textbf{CliqueDetection}(com_i)$
12: **end for**
13: **return** S_{bot}

TABLE III: Traces of Ordinary P2P Networks (24 hrs)

Trace	# of hosts	# of flows	# of dstIP	Size
eMule	16	4,181,845	725,367	42.1G
FrostWire	16	4,479,969	922,000	11.9G
uTorrent	14	10,774,924	2,326,626	57.1G
Vuze	14	7,577,039	1,208,372	20.3G

TABLE IV: Traces of P2P Botnets (24 hrs)

Trace	# of bots	# of flows	# of dstIP	Size
Storm	13	8,603,399	145,967	5.1G
Waledac	3	1,109,508	29,972	1.1G
Sality	5	5,599,440	177,594	1.5G
Kelihos	8	122,182	944	343.9M
ZeroAccess	8	709,299	277	75.2M

TABLE V: Traces of Background Network

Date	Dur	# of hosts	# of flows	Size
2014/12/10	24 hrs	48,607,304	407,523,221	788.7G

Vuze hosts, and we randomly selected 24 hours network traces of each host. More details about D_1 are shown in Table III.

P2P Botnet network traces (D_2)**:** Part of our botnet network traces is also from the University of Georgia dataset [12], which contains 24 hours network traces of 13 hosts infected with Storm and 3 hosts infected with Waledac. We also collected 24 hours network traces of another three infamous P2P botnets, Sality, Kelihos and ZeroAccess. These network traces were collected from the hosts intentionally infected by Kelihos, ZeroAccess, and Sality binary samples obtained from [25]. Furthermore, all malicious activities have been blocked with the same settings as shown in [12]. We collected the network traces of 8 Kelihos bots, 8 ZeroAccess bots and 5 Sality bots. More details about D_2 are shown in Table IV.

Background network traces (D_3)**:** We used the dataset downloaded from the MAWI Working Group Traffic Archive [11] as background network traces, as shown in Table V. This dataset contains 24 hours anonymized network traces at the transit link of WIDE (150Mbps) to the upstream ISP on 2014/12/10 (sample point F). This network traces contains approximate 407,523,221 flows and 48,607,304 unique IPs. 79.3% flows are TCP flows and the rest are UDP flows. We

TABLE VI: Summaries of Experimental Datasets (EDs)

Descriptions	Values
the # of EDs	24
the # of bots in each ED	37
the # of ordinary P2P hosts in each ED	60
the # of internal hosts in each ED	10,000
the AVG # of external hosts in each ED	6,607,714
the AVG # of flows in each ED	91,240,099
the duration of each ED	24 hr

TABLE VII: Detection Rate and False Positive Rate For Different θ_{dd}

θ_{dd}	DR	FP	θ_{dd}	DR	FP
2-10	97/97	$\geq 450/9,903$	500-1,000	81/97	0
15	97/97	$\geq 8/9,903$	5,000	60/97	0
20-25	97/97	$\leq 1/9,903$	10,000	18/97	0
30-185	97/97	0	12,500	5/97	0
200	89/97	0	13,500	0	0

TABLE VIII: Community Detection Results For Different Θ_{mcr}

Θ_{mcr}	FLCR	FBCR	FBSR
0-0.25	0	0	0
0.5	0	0	2.8
1.0	0	0	6.4

utilize ARGUS [26] to process and cluster network traces into the 5-tuple format tcp/udp flows.

3) Experimental Dataset Generation: To evaluate our approach, we generate 24 experimental datasets by mixing the network traces from D_1 and D_2 into different sub-datasets of D_3. Table VI illustrates the summaries of experimental datasets (EDs). Each experimental datasets contains 10,000 internal hosts sampled from D_3, where the network traces of 37 randomly selected hosts are mixed with D_2, and the network traces of another 60 randomly selected hosts are mixed with D_1. To make the experimental evaluation as unbiased and challenging as possible, below we propose two criterions.

Maintain a bipartite network structure. Our system aims to deploy at a network boundary (e.g. firewall, gateway, etc.), where the network forms a bipartite structure, and only network flows within the connections between internal hosts and external hosts could be captured. Then, the network in each experimental dataset should maintain a bipartite network structure, where any pair of internal hosts should not have any communications to each other.

Keep the connectedness of mutual contacts graph. The easiest way to obtain a list of background hosts is to sample the hosts randomly from D_3, with the respect of bipartite structure. However, since D_3 contains an extremely large number of hosts, simply sampling hosts randomly will result in that most of the sampled background hosts do not have a mutual contact with the other background hosts, which is much easier for PeerHunter to identify botnet communities. Because less number of mutual contacts among legitimate hosts means more disconnected legitimate communities in the corresponding MCG, which is in favor of Louvain method to detect strongly connected botnet communities. Therefore, we need to sample a list of internal hosts in a way that every internal host should have at least one mutual contact with at least one another internal host.

To follow the criterions described above without making our evaluation tasks any easier, we propose the following experimental dataset generation procedure:

• Utilize a two-coloring approach to sample the network traces of 10,000 background hosts from D_3 without jeopardize the bipartite network structure and the connectedness of mutual contacts graph: (a) initialize two counters, C_{black} and C_{white}, to count the number of hosts colored in black and white respectively; (b) coloring a random host h_i as black, and C_{black} plus one; (c) coloring all contacts of h_i as white, and increase C_{white} by the number of hosts colored as white in this round; (d) for each new colored host, color its contacts

with the opposite color, and adjust the counters repeatedly, until we have $C_{black} \geq 10,000$ and $C_{white} \geq 10,000$; (e) select the colored host set with exactly 10,000 hosts as the internal hosts, the hosts in the other colored host set will be the external hosts; and (f) extract the network traces of the 10,000 internal hosts from D_3. Then, it forms a bipartite graph, where each colored host set forms a bipartite component, and each host shares at least one mutual contacts with some other hosts from its own bipartite component.

• To maintain a bipartite network structure of botnets and ordinary P2P network traces, we eliminate all communications among bots in D_2 and legitimate P2P hosts in D_1.

• To mix D_1 and D_2 with D_3, each time we randomly select 97 internal hosts from one sub-datasets sampled from D_3, map those IPs to 37 bots' IP in D_2 and 60 legitimate P2P hosts' IP in D_1, and merge the corresponding network traces.

To evaluate our system, 24 synthetic experimental datasets have been created by running this procedure repeatedly.

B. Evaluation on P2P Host Detection

We evaluate the P2P host detection with different parameter settings. This component uses a pre-defined threshold θ_{dd} (Section IV-A) to detect P2P hosts. We applied this component on all 24 experimental datasets, and Table VII shows the experimental results with different θ_{dd}, ranging from 2 to 13500. If θ_{dd} is set too small, non-P2P hosts are likely to be detected as P2P hosts, which results in many false positives. For instance, when $2 \leq \theta_{dd} \leq 10$, there are, on average more than 450 non-P2P hosts have been falsely identified as P2P hosts. In contrast, if θ_{dd} is set too large, all P2P hosts will be removed, which results in false negatives. For instance, when $\theta_{dd} = 5000$, there are, on average 37 P2P hosts have been falsely discarded, and when $\theta_{dd} \geq 12000$, nearly all hosts are removed. When $20 \leq \theta_{dd} \leq 185$, it detects all P2P hosts with a very small number of false positives ($\leq 1/9903$), which demonstrates that our P2P hosts detection component is stable and effective over a large range of θ_{dd} settings. The effectiveness of θ_{dd} is also subject to the time window of the collected data. In our experiment, we used 24 hrs network traces. The destination diversity (DD) of P2P hosts tends to grow over time. Then, θ_{dd} will be effective in a even larger range, if the time window increase.

C. Evaluation on Community Detection

We evaluate the performance of community detection with different parameter settings. We applied this component on the

remain network flows (24 experimental datasets) after the P2P host detection (with $\theta_{dd} = 50$). For each experimental dataset, this component generates a MCG $G_{mc} = (V, E)$ with a pre-defined threshold Θ_{mcr}, where each edge $e_{ij} \in E$ contains a weight $mcr_{ij} \in [0, 1]$. Then, we applied Louvain method (with default resolution 1.0) on the MCG for community detection. The choice of Θ_{mcr} has an influence on the community detection results.

We evaluated the community detection performance in terms of (a) the ability to separate bots and legitimate hosts, (b) the ability to separate bots from different botnets, and (c) the ability to cluster bots within the same botnet. Let *falsely-clustered hosts* denote the number of legitimate hosts that have been clustered with bots into the same community, *cross-community bots* denote the number of bots of different types that have been clustered into the same community, and *split-communities botnets* denote the number of detected communities that contain bots, subtract the number of ground truth botnets (e.g. 5 in our experiments). Then, we propose three evaluation criterions: (a) False Legitimate Cluster Rate (FLCR), which is *falsely-clustered hosts* divided by the total number of legitimate hosts during community detection; (b) False Bot Cluster Rate (FBCR), which is *cross-community bots* divided by the total number of bots during community detection; (c) False Botnet Split Rate (FBSR), which is *split-communities botnets* divided by the total number of ground truth botnets.

Table VIII shows the results with different Θ_{mcr}, ranging from 0 to 1. If Θ_{mcr} is set too small, there will be more non-zero weight edges, which might result in less but larger communities. In contrast, if Θ_{mcr} is set too large, most of the vertices will be isolated, which results in more but small communities. As shown in Table VIII, when $\Theta_{mcr} \leq 0.25$, FBSR also remains 0, which means no botnets have been falsely split into different communities. However, as Θ_{mcr} increasing from 0.5 to 1, FBSR is also increasing, which means bots within the same botnets have been clustered into different communities. This reflects that most of the MCG edge weighs between bots are less than 0.5. If $\Theta_{mcr} \geq 0.5$, bots even within the same botnets will be isolated. FLCR and FBCR are always 0 no matter how Θ_{mcr} has been changed. FLCR is 0 means that all bots are successfully separated from legitimate hosts. FBCR is 0 means none of the communities contains more than one type of bots. This results demonstrate that our system is very effective and robust in separating bots and legitimate hosts, and separating different types of bots.

D. Evaluation on Botnet Detection

We evaluate the botnet detection component with different parameter settings. We applied this component on the remain network flows (24 experimental datasets) after previous two components (with $\theta_{dd} = 50$ and $\Theta_{mcr} = 0.03125$). Table IX shows the results with different $\theta_{avgddr} \in [0, 1]$ and $\theta_{avgmcr} \in [0, 1]$. The results support our idea that the AVGDDR of legitimate host communities is lower than most of the P2P botnets. For instance, the AVGDDR of all (60/60) legitimate

TABLE IX: DR and FPR For Different θ_{avgddr} and θ_{avgmcr}

| θ_{avgmcr} | | θ_{avgddr} | | | | |
	-	0-0.0625	0.125	0.25	0.5	1
0-0.03125	DR	37/37	32/37	32/37	16/37	0/37
	FP	60/60	32/60	0/60	0/60	0/60
0.0625-0.25	DR	37/37	32/37	32/37	16/37	0/37
	FP	0/60	0/60	0/60	0/60	0/60
0.5	DR	21/37	16/37	16/37	16/37	0/37
	FP	0/60	0/60	0/60	0/60	0/60
1	DR	0/37	0/37	0/37	0/37	0/37
	FP	0/60	0/60	0/60	0/60	0/60

host communities are less than 0.25, but the AVGDDR of 32 out of 37 botnets are higher than 0.25. The missing ones turned out to be 5 Sality bots, which could be detected by AVGMCR. As shown in Table IX, legitimate P2P hosts have lower AVGMCR than P2P bots (e.g. $\theta_{avgmcr} = 0.0625$). This experimental results demonstrate that our botnet detection component is effective (detection rate equals to 100 % with zero false positives) and stable over a large range of θ_{avgddr} (e.g. $[0, 0.0625]$) and θ_{avgmcr} (e.g. $[0.0625, 0.25]$).

E. Evaluation on PeerHunter

We evaluate our system according to effectiveness and scalability. Effectiveness is to evaluate the capability of our systems to detect P2P botnets, and scalability is to evaluate the practicality of our systems to deal with the real world big data. We applied PeerHunter on 24 experimental synthetic datasets, with θ_{dd}=50, Θ_{mcr}=0.03125, θ_{avgddr}=0.0625 and θ_{avgmcr}=0.25, and all results are averaged over 24 datasets.

We use detection rate and false positive rate to measure the effectiveness. As shown in Table X, our system identified all 97 P2P hosts from 10,000 hosts, and detected all 37 bots from those 97 P2P hosts, with zero false positives. It is clear that PeerHunter is effective and accurate in detecting P2P botnets.

Our system has a scalable design based on efficient detection algorithm and distributed/parallelized computation. Out of three components in our system, the P2P botnet detection component (community detection and botnet detection as shown in Table X) has a negligible processing time compared with the other two components. This is due to previous two components are designed to reduce a huge amount of the hosts subject to analysis (e.g. 99.03% in our experiments). The P2P host detection component has linear time complexity, since it scans all the input flows only once to compute the flow clusters and further identify P2P flows. However, since it is the very first component to process the input data, which could be large, it still costs the highest processing time (as shown in Table X). To accommodate the growth of a real world input data (big data), we designed and implemented the P2P host detection component using a Map-Reduce framework, which could be deployed in distributed fashion on scalable cloud-computing platforms (e.g. amazon EC2). The MCG extraction component requires pairwise comparison to calculate edges weights. Let n be the number of hosts subject to analysis and m be the maximum number of distinct contacts of a host. We implemented the comparison between each pair of hosts parallelly to handle the growth of n. If we denote k as the number of threads running parallelly, the time complexity

TABLE X: Number of hosts identified by each component

-	Before P2P detection	After P2P detection	After Community detection	After Bot detection
# of hosts	10,000	97	97	37

TABLE XI: PeerHunter Execution Time

-	P2P Host Detection	MCG Extraction	Community Detection	Bot Detection	Total
Processing Time	15 minutes	5 minutes	18 milliseconds	11 milliseconds	20 minutes

of MCG extraction is $O(\frac{n^2 m}{k})$. For a given ISP network, m grows over time. Since our system uses a fixed time window (24 hours), for a given ISP network, m tends to be stable and would not cause a scalability issue. Besides, since the percentage of P2P hosts of an ISP network is relatively small (e.g. 3% [3]), and an ISP network usually has less than 65,536 (/16 subnet) hosts, n would be negligible compared with m. Furthermore, even if n and m are both big numbers, our system could use an as large as possible k to adapt the scale of n and m. In a nutshell, PeerHunter is scalable to handle the real world big data.

VI. CONCLUSION

In this work, we present a novel community behavior analysis based P2P botnet detection system, PeerHunter, which operates under several challenges: (a) botnets are in their waiting stage; (b) the C&C channel has been encrypted; (c) no bot-blacklist or "seeds" are available; (d) none statistical traffic patterns known in advance; and (e) do not require to monitor individual host. We propose three types of community behaviors that can be utilized to detect P2P botnets effectively. In the experimental evaluation, we propose a network traces sampling and mixing method to make the experiments as unbiased and challenging as possible. Experiments and analysis have been conducted to show the effectiveness and scalability of our system. With the best parameter settings, our system can achieved 100% detection rate with none false positives.

REFERENCES

[1] 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.

[2] P. Wang, B. Aslam, and C. C. Zou, "Peer-to-peer botnets," in *Handbook of Information and Communication Security*. Springer, 2010, pp. 335–350.

[3] 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.

[4] X. Wang and D. Ramsbrock, "The botnet problem," *Computer and Information Security Handbook*, pp. 119–132, 2009.

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

[6] E. Stinson and J. C. Mitchell, "Towards systematic evaluation of the evadability of bot/botnet detection methods." *2nd USENIX Workshop on Offensive Technologies (WOOT)*, vol. 8, pp. 1–9, 2008.

[7] 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*, vol. 5, no. 2, 2008, pp. 139–154.

[8] 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.

[9] Q. Yan, Y. Zheng, T. Jiang, W. Lou, and Y. T. Hou, "Peerclean: Unveiling peer-to-peer botnets through dynamic group behavior analysis," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 316–324.

[10] B. Coskun, S. Dietrich, and N. Memon, "Friends of an enemy: identifying local members of peer-to-peer botnets using mutual contacts," in *Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC)*. ACM, 2010, pp. 131–140.

[11] "Mawi working group traffic archive," Mar. 2015. [Online]. Available: http://mawi.wide.ad.jp/mawi/samplepoint-F/2015/201503101400.html

[12] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li, "Peerrush: Mining for unwanted p2p traffic," *Journal of Information Security and Applications*, vol. 19, no. 3, pp. 194–208, 2014.

[13] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, and W. Lee, "Bothunter: Detecting malware infection through ids-driven dialog correlation." in *Usenix Security Symposium*, vol. 7, 2007, pp. 1–16.

[14] J. Felix, C. Joseph, and A. A. Ghorbani, "Group behavior metrics for p2p botnet detection," in *Information and Communications Security*. Springer, 2012, pp. 93–104.

[15] J. Zhang, X. Luo, R. Perdisci, G. Gu, W. Lee, and N. Feamster, "Boosting the scalability of botnet detection using adaptive traffic sampling," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*. ACM, 2011, pp. 124–134.

[16] 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.

[17] B. Soniya and M. Wilscy, "Fuzzy inference system based on entropy of traffic for bot detection on an endpoint host," in *Data Science & Engineering (ICDSE), 2014 International Conference on*. IEEE, 2014, pp. 112–117.

[18] 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.

[19] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. C. Freiling, "Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm." *First USENIX Workshop on Large-Scale Exploits and Emergent Threats*, vol. 8, no. 1, pp. 1–9, 2008.

[20] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, and P. Hakimian, "Detecting p2p botnets through network behavior analysis and machine learning," in *Privacy, Security and Trust (PST), 2011 Ninth Annual International Conference on*. IEEE, 2011, pp. 174–180.

[21] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Usilng machine learning technliques to identify botnet traffic," in *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*. IEEE, 2006, pp. 967–974.

[22] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[23] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.

[24] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008.

[25] "Malware sample sources for researchers," Mar. 2015. [Online]. Available: https://zeltser.com/malware-sample-sources/

[26] "Argus: Auditing network activity," Mar. 2015. [Online]. Available: http://qosient.com/argus/