Summary of Changes

We are immensely pleased to be offered the helpful suggestions, and have revised the manuscript fully according to the reviewers' comments. In this revision, we have taken the opportunity to address reviewers' concerns that were kindly drawn to our attention. The following summarizes the latest revision conducted to the paper.

1) A statement of contribution was added to the end of Section I (introduction) to summarize the contributions of our work.
2) Section II (related work) was thoroughly revised to update our references with more latest work (around 10 most recent works were added).
3) Section III-C3 was revised to clarify the usage of clique detection in our approach.
4) Section V-A2 was thoroughly revised and Table VI was added to provide more information about our background network traces.
5) Section V-A3 was thoroughly revised and Table VII was updated to clarify the process and result of our experimental data generation.
6) Section V-B was revised and Table VIII was updated show the performance of P2P network flow detection on newly generated experimental datasets.
7) Section V-D was revised and Table X was updated to clarify the performance of botnet detection with different parameters.
8) Section V-E was divided into four subsections. Section V-E1 discuss about the effectiveness of our system. Section V-E2 was revised and Fig. 4 was added to show that Enhanced PeerHunter is scalable while running on different size of datasets. Section V-E3 and Fig. 5 were added to discuss about the effectiveness and sensitiveness of our system while using different combinations of system parameters. Section V-E4 was added to discuss about our interesting findings of the "true" false positives, which demonstrated that our system has the potential to detect other unknown botnets.
9) Section V-F was revised to discuss more about the insights on why our system is robust against the proposed attacks.
10) Section V-G was added and Table XIV was revised to compare our system with one of the state of art P2P botnet detection system Zhang et al. [6] (the upgraded version of Zhang et al. [18]).
11) Section VI was added to discuss about (1) evasions and possible solutions (Section VI-A), (2) the deployment of Enhanced PeerHunter, and (3) the potential of extending Enhanced PeerHunter to detect other botnets.

The detail on how the comments are addressed are supplied below in the order of reviewers' comments. Each comment and the written response are presented in pair.

I. Response to reviewer 1

The authors would like to express sincere thanks to the reviewer for the thorough and constructive comments. In what follows, we present detailed comments in response to the individual points raised by the reviewer and elaborate on how the manuscript has been revised.

Comment 1.1: In the paper, the authors first evaluate three components separately and then evaluate the whole system. When the authors evaluate the last two components, the best thresholds of the previous components are used to remove all the FP and keep the most TP. For example, the threshold of dd is set as 50, which keeps all the 97 P2P hosts and introduces no FP. It is fine to evaluate individual component by setting the perfect thresholds for the other components. However, using all the best thresholds together to evaluate the whole system and only showing the best results (100% DR and no FP) is unfair. The authors need to show the results when different combinations of the thresholds are used. For example, when 10 is used as the threshold of dd, how the results look like. Such evaluations are needed to show how sensitive the Enhanced PeerHunter is to the thresholds. Besides, when the Enhanced PeerHunter is applied on a different network or used to detect some other P2P botnet, the thresholds will need to be adjusted, thus having the above results will give some advice to the researchers or network admins.

Response 1.1: Thank you for the insightful comment. In this revision, we added a new section in the experimental evaluation (Section V-E3) to evaluate and discuss about the sensitivity and effectiveness of different combinations among different values of $\Theta_{dd} \in \{10, 30, 50\}$, $\Theta_{avgddr} \in \{0.2, 0.6\}$ and $\Theta_{avgmcr} \in \{0.0, 0.025, 0.05, 0.075, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$. In the following, we list a part of the new section, V-E3, in addition to the new figure, Fig. 5, which were added to address this comment.

“

3) Analyzing the Effectiveness of System Parameters: Although we have analyzed the effectiveness of Θ_{dd}, Θ_{avgddr} and Θ_{avgmcr} within the corresponding components, the effectiveness of combinations among different values of Θ_{dd}, Θ_{avgddr} and Θ_{avgmcr} has not been studied. As shown in Fig. 5, we use precision, recall and false positives to evaluate the effectiveness of different parameter combinations. As discussed in Section V-B, Θ_{dd} is used to detect P2P network flow clusters. Larger Θ_{dd} tends to result in more false negatives (lower recall), and smaller Θ_{dd} tends to result in more false positives (lower precision). For instance, changing Θ_{dd} from 30 and 50 to 10 results in 47 and 42 more false positives ($\Theta_{avgddr} = 0.15$) as shown in Fig. 5c and Fig. 5f, respectively. When $\Theta_{dd} \in \{30, 50\}$, $\Theta_{avgddr} \in [0.15, 0.35]$ and $\Theta_{avgddr} \in [0.2, 0.6]$, our system yields 100% detection rate with zero false positive. Even when

$\Theta_{dd} = 10$, our system can still work effectively with $\Theta_{avgddr} \in [0.25, 0.35]$ and $\Theta_{avgddr} \in [0.2, 0.6]$. This demonstrates our system can work effectively over several different parameter combinations.

”

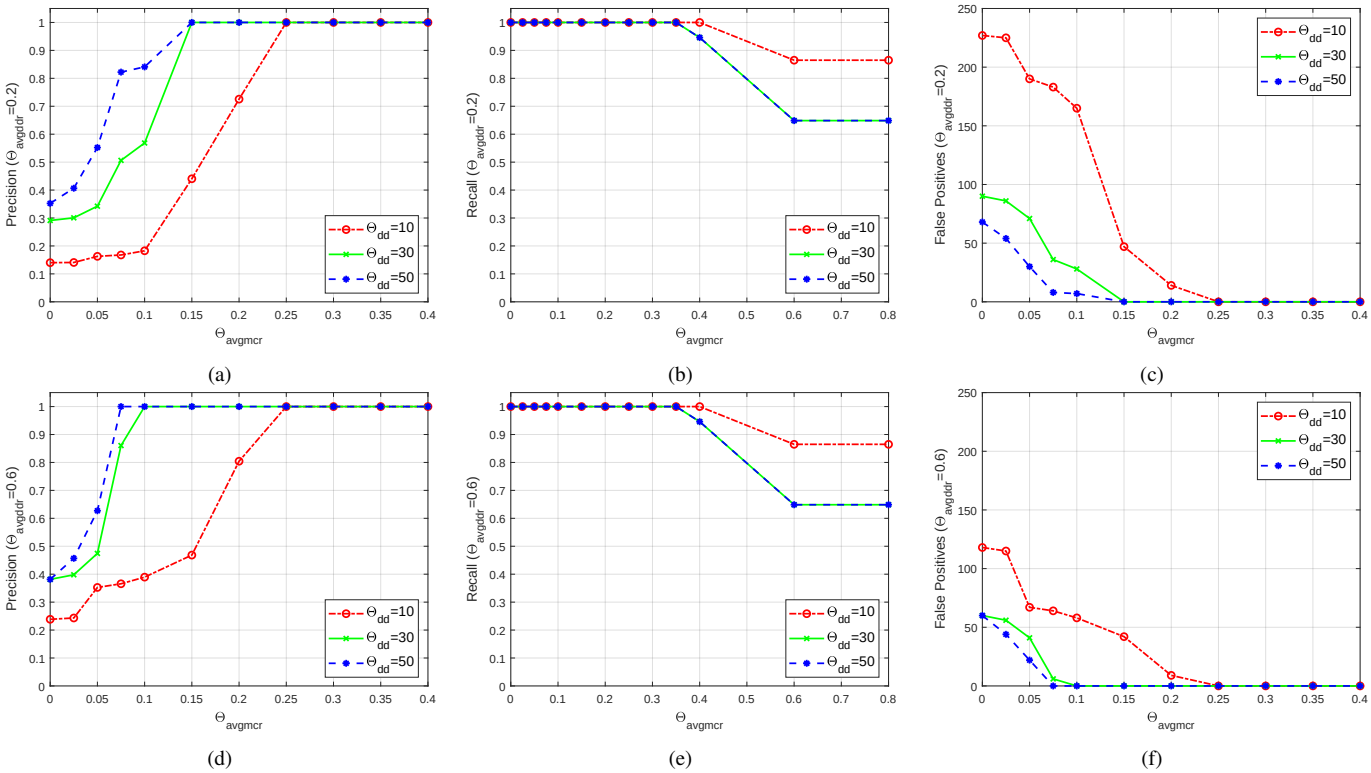

Fig. 5: Precision, recall and false positives given different Θ_{dd}, Θ_{avgddr} and Θ_{avgmcr} ($\Theta_{mcr} = 0.05$).

Comment 1.2: In the related work, the authors claim that traffic signatures (e.g., volume per hour, duration per flow) can be easily changed by botnets, then the detection system will be evaded. However, enhanced PeerHunter will also be evaded if the botnet changes the communication pattern, even very slightly. Enhanced PeerHunter clusters the traffic using 4-tuple $[ip_{src}, proto, bpp_{out}, bpp_{in}]$. Based on this definition and Algorithm 1, detecting P2P MNF and P2P host requires the host has a large number of flows that have exactly the same bpp_{out} and bpp_{in}. A P2P bot can easily evade Enhanced PeerHunter by adding paddings or junk packets when they talk to the peers. In this case, the bpp_{out} and bpp_{in} of the flows will be different, so they will not be clustered together, and Enhanced PeerHunter is evaded. Similarly, 3-tuple $[proto, bpp_{out}, bpp_{in}]$ is used to decide whether a pair of P2P network flow clusters are the same type. This will also be failed if botnet adds paddings or junk packets.

Response 1.2: Thank you for the insightful comment. We agree with the reviewer about that “enhanced PeerHunter will also be evaded if the botnet changes the communication pattern”. Therefore, we remove

the claim that Enhanced PeerHunter could easily overcome this evasion. Instead, we add a new section in the discussion (Section VI-A) to discuss about the evasions (including this one) and possible solutions. In the following, we list the new section, VI-A, which were added to address this comment.

“

A. Evasions and Possible Solutions

To avoid detection by Enhanced PeerHunter, the botmaster could use a combination of the following three approaches: (a) add randomized paddings or junk packets to influence the bytes-per-packet characterizes for network flow clustering, (b) reduce the number or rate of destination diversity, or (c) reduce the number or rate of mutual contacts. To deal with the randomized spatial-communication behavior, we could adopt more time-communication features, such as packet/flow duration and inter-packet delays, or apply more general features, such as the distribution, mean or standard deviation of bytes-per-packet. The other two evasion approaches would be the victory of our system. On one hand, to reduce the number or rate of destination diversity, a bot has to limit its communication to the network of certain locations, which degrade the P2P botnet into centralized botnet to a certain extent. On the other hand, as shown in V-F, reducing the number or rate of mutual contacts is expensive and will make botnets easier to expose themselves.

”

Comment 1.3: It seems like the first component (P2P Network Flow Detection) is the key of the Enhanced PeerHunter as it removes all the FPs. The authors may want to use a small value (e.g., 10) as the threshold of dd that introduces some FPs, and then show whether the following two components can remove these FPs.

Response 1.3: Thank you for the comment. Our first component (P2P Network Flow Detection) is designed to remove most of the false positives resulted from the non-P2P network flows, but will not remove the false positives resulted from legitimate P2P network flows. And the following two components are designed to remove the legitimate P2P false positives, thus, detect the P2P bots. To clarify this, we updated the Table VIII, which presents the detection rate and false positives (including both P2P FPs and non-P2P FPs) with different Θ_{dd}, as follows.

We also add a new section in the experimental evaluation (Section V-E3) to evaluate botnet detection results of different combinations among different values of $\Theta_{dd} \in \{10, 30, 50\}$, $\Theta_{avgddr} \in \{0.2, 0.6\}$ and $\Theta_{avgmcr} \in \{0.0, 0.025, 0.05, 0.075, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$. In the following, we list a part of the

TABLE VIII: Detection Rate and False Positive Rate For Different Θ_{dd} (P_i is the set of P2P hosts within the background network traces that have no less than $i \times 15$ minutes active time. All the hosts of 4 legitimate P2P applications and 5 P2P botnets have 24 hours active time.)

Θ_{dd}	Detection Rate											False Positive Rate
	Bot	P2P	P_1	P_2	P_4	P_8	P_{14}	P_{20}	P_{32}	P_{48}	P_{96}	
2	37/37	60/60	667/667	325/325	180/180	66/66	38/38	26/26	21/21	13/13	4/4	1,052/9,236
5	37/37	60/60	364/667	242/325	180/180	66/66	38/38	26/26	21/21	13/13	4/4	110/9,236
10	37/37	60/60	156/667	133/325	106/180	66/66	38/38	26/26	21/21	13/13	4/4	44/9,236
30	37/37	60/60	36/667	36/325	36/180	33/66	30/38	26/26	21/21	13/13	4/4	4/9,236
50-180	37/37	60/60	15/667	15/325	15/180	15/66	15/38	15/26	15/21	13/13	4/4	0/9,236
185	37/37	60/60	6/667	6/325	6/180	6/66	6/38	6/26	6/21	6/13	4/4	0/9,236
200	29/37	60/60	4/667	4/325	4/180	4/66	4/38	4/26	4/21	4/13	2/4	0/9,236
500-1,000	21/37	60/60	1/667	1/325	1/180	1/66	1/38	1/26	1/21	1/13	1/4	0/9,236
5,000	13/37	45/60	0/667	0/325	0/180	0/66	0/38	0/26	0/21	0/13	0/4	0/9,236
10,000	0/37	18/60	0/667	0/325	0/180	0/66	0/38	0/26	0/21	0/13	0/4	0/9,236
12,500	0/37	5/60	0/667	0/325	0/180	0/66	0/38	0/26	0/21	0/13	0/4	0/9,236
13,500	0/37	0/60	0/667	0/325	0/180	0/66	0/38	0/26	0/21	0/13	0/4	0/9,236

new section, V-E3, in addition to the new figure, Fig. 5 (the same as in **Response 1.1**), which were added to address the performance of the following two components in different situations.

“

3) Analyzing the Effectiveness of System Parameters: Although we have analyzed the effectiveness of Θ_{dd}, Θ_{avgddr} and Θ_{avgmcr} within the corresponding components, the effectiveness of combinations among different values of Θ_{dd}, Θ_{avgddr} and Θ_{avgmcr} has not been studied. As shown in Fig. 5, we use precision, recall and false positives to evaluate the effectiveness of different parameter combinations. As discussed in Section V-B, Θ_{dd} is used to detect P2P network flow clusters. Larger Θ_{dd} tends to result in more false negatives (lower recall), and smaller Θ_{dd} tends to result in more false positives (lower precision). For instance, changing Θ_{dd} from 30 and 50 to 10 results in 47 and 42 more false positives ($\Theta_{avgddr} = 0.15$) as shown in Fig. 5c and Fig. 5f, respectively. When $\Theta_{dd} \in \{30, 50\}$, $\Theta_{avgddr} \in [0.15, 0.35]$ and $\Theta_{avgddr} \in [0.2, 0.6]$, our system yields 100% detection rate with zero false positive. Even when $\Theta_{dd} = 10$, our system can still work effectively with $\Theta_{avgddr} \in [0.25, 0.35]$ and $\Theta_{avgddr} \in [0.2, 0.6]$. This demonstrates our system can work effectively over several different parameter combinations.

”

Comment 1.4: The authors do not show that Enhanced PeerHunter has the capability to detect unknown botnets.

Response 1.4: Thank you for the comment. Since our system relies on unsupervised approach (e.g., community detection, thresholding), we argue that compared with botnet detection systems based on supervised machine learning algorithms (e.g., classification), our system does not require any knowledge of existing malicious behaviours and thus have the potential to detect unknown botnets. In this revision,

we added a new section in the experimental evaluation (Section V-E4) to analyze the "true" false positives when $\Theta_{dd} = 10$, which demonstrates that our system has the potential to detect other unknown botnets. In the following, we list a part of the new section, V-E4.

"

4) Analyzing the "True" False Positives when $\Theta_{dd} = 10$: In this section, we discuss about some interesting findings about the false positives result from setting $\Theta_{dd} = 10$. As discussed in Section III-C2, Θ_{avgddr} is used to capture the "P2P behavior" of network flows, and Θ_{avgmcr} is used to capture the "botnet behavior" of network flows. Hence, if we use a larger Θ_{avgddr} (i.e., 0.6) and a smaller Θ_{avgmcr} (i.e., 0.0), most of the false positives should be legitimate P2P host. For instance, in Fig. 5f, when $\Theta_{dd} = 10$, $\Theta_{avgddr} = 0.6$ and $\Theta_{avgmcr} = 0.0$, there are 115 out of 118 hosts are P2P hosts (60 from D_{p2p} and 55 from D^b_{p2p}). On the other hand, we assume that if we use a smaller Θ_{avgddr} (i.e., 0.2) and a larger Θ_{avgmcr} (i.e., 0.15), some of the false positives might come from the other types of botnets. As shown in Fig. 5c, when $\Theta_{dd} = 10$, $\Theta_{avgddr} = 0.2$ and $\Theta_{avgmcr} = 0.15$, there are 9 out of 47 that are not our known legitimate P2P hosts. We investigated these false positives, with their anonymized and payload-free network traces. It turns out that, 4 out of the 9 hosts (i.e., "180.217.2.181", "180.217.2.246", "180.217.2.248" and "180.217.2.177") are listed in the Barracuda Reputation Block List (BRBL) [47], a highly accurate list of the IP addresses known to send spam. Hence, we are convinced that those false positives are infected with virus or botnets. These findings demonstrate that our system has the potential to detect other unknown botnets.

"

We also added a new section in the discussion (Section VI-C) to discuss about the potential to extend Enhanced PeerHunter to detect other/unknown botnets, listed as follows.

"

C. Extend Enhanced PeerHunter to detect other botnets

Although Enhanced PeerHunter is designed to detect P2P botnets, our idea of using mutual contact graph has the potential to detect not only unknown botnets, but also the other types of botnets (e.g., centralized botnets, such as IRC botnets [31], mobile botnets [50]). Since bots are usually controlled by machines, rather than humans, bots from the same botnets tend to communicate with a similar set of peers or attacking targets. For instance, bots from the same IRC botnets tend to contact a similar set of C&C servers, while bots from the same mobile botnets tend to contact a similar set of satellite servers. Hence, we argue that Enhanced PeerHunter could be easily extended to detect the other types of botnets.

”

Comment 1.5: The authors use the dataset from the MAWI Working Group Traffic Archive as the background traffic. However, it is possible that the background traffic contains some legitimate P2P and/or botnet P2P traffic. If so, the Enhanced PeerHunter does not detect any of them. The authors need to show the background traffic does not contain P2P traffic, especially P2P botnet traffic.

Response 1.5: Thank you for the insightful comment. In this revision, first, we manually investigate the background network traces (WIDE, containing 24 hours anonymized and payload-free network traces), and we identified a set of P2P hosts within the background traffic using port-analysis. We updated part of Section V-A2 and made Table VI, to described our P2P hosts findings from the background traffic, as follows.

“ . . . , We investigated the background network traces, and used our best effort to separate the P2P traffic (D^b_{p2p}) from the non-P2P traffic (D^b_{p2p}). Since the WIDE dataset is anonymized and payload-free, it prevents us from using payload analysis to thoroughly check if P2P traffic, especially P2P Botnet traffic exists there. Instead, we used port analysis to manually detect P2P traffic within the dataset, which is based on the simple concept that many P2P applications have default ports on which they function (see [45] for a list of default network ports of popular P2P applications). We manually examined all the network flows of each host in the background network traces. If a host involves in more than five flows that are using any of the default P2P port values in either source port or destination port, we consider the host as a P2P host. After this procedure, we identified 667 P2P hosts.

One thing worth to be noticed is that despite the whole background network traces last for 24 hours, not all these P2P hosts are active for the entire 24 hours. P2P hosts that do not have enough active time, may not produce sufficient network flows for our system to work (as discussed in Section V-B). To ensure a fair and rigorous evaluation, we estimated the active time of each P2P host. We divide the 24 hours background network traces into 96 15-minute blocks. If a P2P host has any network flow fell in a block, we consider it is active within that block. We use the number of blocks where a P2P host is active to estimate the total active time of each P2P host. Table VI reflects the active time distribution of these P2P hosts. As shown in Table VI, even though there are 667 P2P hosts in total, only 4 of them have been active for the entire 24 hours and 26 of them have been active for no less than 5 hours.

”

TABLE VI: Active Time of P2P hosts within the Background Network Trace (P_i is the set of P2P hosts have no less than $i \times 15$ minutes active time.)

-	**# of hosts**	-	**# of hosts**	-	**# of hosts**
P_1	667	P_8	66	P_{32}	21
P_2	325	P_{14}	38	P_{48}	13
P_4	180	P_{20}	26	P_{96}	4

Second, we updated our experimental datasets by adding all 667 P2P hosts found in the background network traces to all the EDs. And we revised part of the experimental dataset generation (Section V-A3) to address this as follows.

" ..., We consider a case that contains 10,000 internal hosts. For each synthetic experimental dataset, the 667 P2P hosts in D^b_{p2p} are considered as the internal hosts. Then, another 9,333 internal hosts are sampled from D^b_{non}, where the traffic of 37 randomly selected hosts are mixed with the traffic of 37 P2P bots in D_{bot}, and the traffic of another 60 randomly selected hosts are mixed with the traffic of 60 P2P hosts in D_{p2p}. ...
"

Third, we re-did our experiments based on the new experimental datasets, and revised the evaluation on P2P network flow detection (Section V-B) and Table VIII (the same as in **Response 1.5**) to evaluate and discuss about our system's performance on detecting P2P hosts (of different durations of active time) found in the background network traces, as follows.

" We evaluate the P2P network flow detection with different Θ_{dd}. We applied this component on all 100 EDs, and Table VIII shows the average detection rate and false positives with different Θ_{dd}, ranging from 2 to 13,500. 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 $2 \leq \Theta_{dd} \leq 5$, at least 110 non-P2P hosts have been falsely identified as P2P hosts. If Θ_{dd} is set too large, all P2P hosts will be removed, which results in false negatives. For instance, when $\Theta_{dd} = 10,000$, most of the P2P hosts have been falsely discarded, and only 18 P2P hosts have been detected.

On the other hand, the effectiveness of Θ_{dd} is also subject to the active time of P2P hosts. Since if a P2P host has less active time, it tends to generate less number of P2P network flows to show enough destination diversity, so that it will not be distinguished from non-P2P network flows by our system. For instance, since all the bots and P2P hosts in D_{bot} and D_{p2p} have 24 hours active time, our system can distinguish them well from the non-P2P network flows. However, not all the P2P hosts in D^b_{p2p} are active for the entire 24 hours. As shown in Table VIII, when the active time of a P2P host is less than 5 hours

(not belonging to P_{20} the set of hosts have no lees than 20×15 minutes active time), it becomes hard for our system to detect P2P network flows from non-P2P network flows ($\Theta_{dd} < 30$). Hence, if we consider P2P hosts that have no less than 12 hours active time (P_{48}), when $30 \leq \Theta_{dd} \leq 180$, our system detects all P2P hosts with a small number of false positives ($\leq 4/9,236$), which demonstrates that our P2P network flow detection component is stable and effective over a large range of Θ_{dd} settings.
”

Last but not the least, the anonymized and payload-free WIDE traces prevent us from knowing the existence of P2P botnets. In our experimental evaluation, however, based on the findings and analysis of the true false positives (as described in **Response 1.4**), it demonstrates that our system has the potential to detect other unknown botnets.

Comment 1.6a: Details of the background data set are missing. It is hard to evaluate the results without these information.

a. Table V shows that the 24 hours background trace includes about 48 million hosts and 407 million flows. I assume most of the 48 million IPs are external IPs, otherwise, every host only has 8-9 flows during a period of 24 hours, which is too few. It would be great if the authors can list how many IPs are internal and external.

Response 1.6a: Thank you for the comment. Table V shows the general information about the original background network traces. However, in our experiments, we didn't use all of the traffic within the background traffic. Instead, we generate 100 synthetic experimental datasets by mixing network traces from the botnet datasets, legitimate P2P application dataset and part of the background datasets. We revised Section V-A3 to clarify the experimental dataset generation and Table VII shows the summaries of our experimental datasets (including the average number of internal/external hosts, the average number of flows and so forth).

“ . . . , we consider a scenario that an organization has a set of internal hosts communicating with external hosts (outside of the organization), and our system is deployed at the boundary of the organization. Since our original datasets do not maintain a internal-external network structure while collecting them, we generate synthetic experimental datasets by mixing network traces from the original datasets. We consider a case that contains 10,000 internal hosts. For each synthetic experimental dataset, the 667 P2P hosts in D^b_{p2p} are considered as the internal hosts. Then, another 9,333 internal hosts are sampled from D^b_{non},

where the traffic of 37 randomly selected hosts are mixed with the traffic of 37 P2P bots in D_{bot}, and the traffic of another 60 randomly selected hosts are mixed with the traffic of 60 P2P hosts in D_{p2p}. . . .
”

TABLE VII: Summaries of Experimental Datasets (EDs)

Descriptions	Values
the # of EDs	100
the # of bots (D_{bot}) in each ED	37
the # of legitimate P2P hosts (D_{p2p}) in each ED	60
the # of P2P hosts (D^{b}_{p2p}) in each ED	667
the # of internal hosts in each ED	10,000
the AVG # of external hosts in each ED	8,642,618
the AVG # of flow in each ED	97,640,210
the duration of each ED	24 hr

Comment 1.6b: b. I notice that the background traffic was captured on sample point F. Does this mean there are multiple points and F is only one of them? Another question is whether the captured traffic was sampled. If the data set does not include all the incoming/outgoing traffic or the traffic was sampled, then all the thresholds are not accurate. For example, when Enhanced PeerHunter runs on the un-sampled traffic, using 50 as the threshold of dd may introduce some FPs.

Response 1.6b: Thank you for the comment. 1) “sample point F” is the point that provide 24 hours traffic, and some of the other points only provide 15 minutes traffic of each day. 2) The captured traffic was not sampled and was captured at a link speed of 150Mbps. We revised part of the Section V-A2 to clarify the description of the background traffic, as follows.

“ . . . , We used a dataset from the MAWI Working Group Traffic Archive [12] as our background network traces, which contains 24 hours anonymized and payload-free network traces at the transit link of WIDE (150Mbps) to the upstream ISP on 2014/12/10 (as shown in Table VII). The dataset contains approximate 407,523,221 flows and 48,607,304 unique IPs. 79.3% flow are TCP flow and the rest are UDP flow. . . .
”

Comment 1.6c: The dataset in [16] (http://mawi.wide.ad.jp/mawi/samplepoint-F/2015/201503101400.html) contains 15 minutes traffic captured on 03/10/2015, instead of a 24 hours traffic on 12/10/2014.

Response 1.6c: Thank you for the comment. We updated it to the correct link, as http://mawi.wide.ad.jp/mawi/ditl/ditl201412/.

Comment 1.7: Some interesting results can be explained better. For example, In Table IX, the FPR is 100% when θ_{avgmcr} is in [0.0, 0.1) and θ_{avgddr} is in [0.00, 0.65]. However, the FPR decreases to 0% when the θ_{avgmcr} is 0.1 and θ_{avgddr} is in [0.00, 0.65]. This shows that the Enhanced PeerHunter might be too sensitive to the thresholds.

Response 1.7: Thank you for the insightful comment. We found that there was a misleading in our old presenting of the false positive rate (by using the base of the number of legitimate P2P hosts, rather than the total number of non-bot hosts). In this revision, we updated Table X with the number of false positives (rather than FPR) and new experiment results, and revised the evaluation on botnet detection (Section V-D) as follows.

" ..., We evaluate the botnet detection component with different parameter settings. We applied this component on the remaining network flows (100 EDs) after previous two components (with $\Theta_{dd} = 30$ and $\Theta_{mcr} = 0.1$). We assume that all the host in the background trace (D^b and D^b_{p2p}) are not malicious, and will be reported as false positives if being detected.

Table X shows the P2P botnet detection results which supports our idea that the AVGDDR of legitimate P2P network flow cluster communities is lower than most of the P2P botnets network flow cluster communities. For instance, the AVGDDR of all (60/60) legitimate P2P network flow cluster communities are higher than 0.6 , and the AVGDDR of 32 out of 37 botnets are higher than 0.8. The missing ones turned out to be 5 Sality bots, which could be detected by AVGMCR. Also, the legitimate P2P network flow clusters have lower AVGMCR than P2P bots (i.e., $\Theta_{avgmcr} \in [0.15, 0.35]$). ...
"

We also added a new section in the experimental evaluation (Section V-E3 and Fig. 5) to evaluate and discuss about the sensitivity and effectiveness of different parameter combinations (the same as described in **Response 1.1**).

TABLE X: Botnet Detection Results For Different Θ_{avgddr} and Θ_{avgmcr}. (ZeroA.: the detection rate of ZeroAccess; FP: the number of false positives.)

		Θ_{avgddr}				
Θ_{avgmcr}	-	0.0	0.2	0.4	0.6	0.8
0.0	ZeroA.	100%	100%	100%	100%	100%
	Waledac	100%	100%	100%	100%	100%
	Storm	100%	100%	100%	100%	100%
	Kelihos	100%	100%	100%	100%	100%
	Sality	100%	100%	100%	100%	0%
	Precision	28.9%	29.1%	29.3%	38.1%	34.8%
	Recall	100%	100%	100%	100%	86.5%
	FP	91	90	89	60	60
	F-score	44.8%	45.1%	45.4%	55.2%	49.6%
0.05	ZeroA.	100%	100%	100%	100%	100%
	Waledac	100%	100%	100%	100%	100%
	Storm	100%	100%	100%	100%	100%
	Kelihos	100%	100%	100%	100%	100%
	Sality	100%	100%	100%	100%	0%
	Precision	33.9%	34.2%	34.9%	47.4%	43.8%
	Recall	100%	100%	100%	100%	86.5%
	FP	72	71	69	41	41
	F-score	50.7%	51%	51.7%	64.3%	58.2%
0.1	ZeroA.	100%	100%	100%	100%	100%
	Waledac	100%	100%	100%	100%	100%
	Storm	100%	100%	100%	100%	100%
	Kelihos	100%	100%	100%	100%	100%
	Sality	100%	100%	100%	100%	0%
	Precision	56.0%	56.9%	56.9%	100%	100%
	Recall	100%	100%	81%	100%	86.5%
	FP	29	28	28	0	0
	F-score	71.8%	72.5%	72.5%	100%	92.8%
0.15-0.35	ZeroA.	100%	100%	100%	100%	100%
	Waledac	100%	100%	100%	100%	100%
	Storm	100%	100%	100%	100%	100%
	Kelihos	100%	100%	100%	100%	100%
	Sality	100%	100%	100%	100%	0%
	Precision	100%	100%	100%	100%	100%
	Recall	100%	100%	100%	100%	86.5%
	FP	0	0	0	0	0
	F-score	100%	100%	100%	100%	92.8%
0.4	ZeroA.	100%	100%	100%	100%	100%
	Waledac	100%	100%	100%	100%	100%
	Storm	84.6%	84.6%	84.6%	84.6%	76.9%
	Kelihos	100%	100%	100%	100%	100%
	Sality	100%	100%	100%	100%	0%
	Precision	100%	100%	100%	100%	100%
	Recall	94.6%	94.6%	94.6%	94.6%	78.4%
	FP	0	0	0	0	0
	F-score	97.2%	97.2%	97.2%	97.2%	87.9%
0.6-0.8	ZeroA.	100%	100%	100%	100%	100%
	Waledac	100%	100%	100%	100%	0%
	Storm	0%	0%	0%	0%	0%
	Kelihos	100%	100%	100%	100%	100%
	Sality	100%	100%	100%	100%	0%
	Precision	100%	100%	100%	100%	100%
	Recall	64.9%	64.9%	64.9%	64.9%	43.2%
	FP	0	0	0	0	0
	F-score	78.7%	78.7%	78.7%	78.7%	60.4%

Comment 1.8: In section V-E, the authors claim that the results are stable over a large range of θ_{avgddr} (i.e., [0.0,0.65]) and θ_{avgmcr} (i.e., [0.1, 0.3]). This is not quite true, when we consider avgddr and avgmcr together, (0,65-0.00)*(0.3-0.1)=0.13. This shows that these two thresholds need to be picked very carefully. Note that these results are based on the fact that the first component has already removed all the FPs.

Response 1.8: Thank you for the insightful comment. 1) We have clarified in **Response 1.3** that our first component (P2P Network Flow Detection) is designed to remove most of the false positives resulted from the non-P2P network flows, but will not remove the false positives resulted from legitimate P2P network flows. And the following two components are designed to remove the legitimate P2P false positives, thus, detect the P2P bots. 2) We also added a new section in the experimental evaluation (Section V-E3 and Fig. 5) to evaluate and discuss about the sensitivity and effectiveness of different parameter combinations (the same as described in **Response 1.1**). 3) We revised Section V-E1 to improve the statement of our analysis about our system effectiveness, as follows.

" . . . , We applied Enhanced PeerHunter on 100 EDs, with Θ_{dd}=30, Θ_{mcr}=0.1, Θ_{avgddr}=0.6 and Θ_{avgmcr}=0.15, and all results are averaged over 100 EDs. Using Θ_{avgddr}=0.6 and Θ_{avgmcr}=0.15 is based on our empirical study (shown in Table X). As illustrated in Table XI, our system identified all 97 P2P hosts from 10,000 hosts, and detected all 37 bots from those 97 P2P hosts, with zero false positive. It is clear that Enhanced PeerHunter is effective and accurate in detecting P2P botnets. . . .
"

Comment 1.9a: The presentations of the results need to be improved:

a. At the end of Section V-C, the authors decide to use 0.15 as the threshold of mcr because it gives the highest BAI, BSI and BLSI. However, in Table VIII, the highest BAI, BSI and BLSI fall in [0.00, 0.15), which does not include 0.15.

Response 1.9a: Thank you for the comment. In the revision, we re-did all the experiments using $\Theta_{mcr} = 0.1$, and revised the last sentence of Section V-C to "Since larger Θ_{mcr} results in less edges in the MCG, which could reduce the execution time of the community detection procedure, we use $\Theta_{mcr} = 0.1$ as our system parameter.".

Comment 1.9b: b. In Table VII, the thresholds of dd are not continuous, it is not clear why the authors select 2-10, 15, 20-25, instead of 2-10, 11-19, 20-25. Besides, when the dd is 5, the FP should be a fixed

value, rather than $\leq 8/9,903$

Response 1.9b: Thank you for the comment. Since the range of Θ_{dd} in Table VIII is between 2 and 13,500, it's very hard to provide a continuous analysis of Θ_{dd} in such a large range. Thus, we select several typical threshold results based on our empirical study. And the P2P detection results (i.e., detection rate and false positive rate) are an average over 100 experimental datasets. The following shows the revised version of Table VIII.

TABLE VIII: Detection Rate and False Positive Rate For Different Θ_{dd} (P_i is the set of P2P hosts within the background network traces that have no less than $i \times 15$ minutes active time. All the hosts of 4 legitimate P2P applications and 5 P2P botnets have 24 hours active time.)

Θ_{dd}	Detection Rate											False Positive Rate
	Bot	**P2P**	P_1	P_2	P_4	P_8	P_{14}	P_{20}	P_{32}	P_{48}	P_{96}	
2	37/37	60/60	667/667	325/325	180/180	66/66	38/38	26/26	21/21	13/13	4/4	1,052/9,236
5	37/37	60/60	364/667	242/325	180/180	66/66	38/38	26/26	21/21	13/13	4/4	110/9,236
10	37/37	60/60	156/667	133/325	106/180	66/66	38/38	26/26	21/21	13/13	4/4	44/9,236
30	37/37	60/60	36/667	36/325	36/180	33/66	30/38	26/26	21/21	13/13	4/4	4/9,236
50-180	37/37	60/60	15/667	15/325	15/180	15/66	15/38	15/26	15/21	13/13	4/4	0/9,236
185	37/37	60/60	6/667	6/325	6/180	6/66	6/38	6/26	6/21	6/13	4/4	0/9,236
200	29/37	60/60	4/667	4/325	4/180	4/66	4/38	4/26	4/21	4/13	2/4	0/9,236
500-1,000	21/37	60/60	1/667	1/325	1/180	1/66	1/38	1/26	1/21	1/13	1/4	0/9,236
5,000	13/37	45/60	0/667	0/325	0/180	0/66	0/38	0/26	0/21	0/13	0/4	0/9,236
10,000	0/37	18/60	0/667	0/325	0/180	0/66	0/38	0/26	0/21	0/13	0/4	0/9,236
12,500	0/37	5/60	0/667	0/325	0/180	0/66	0/38	0/26	0/21	0/13	0/4	0/9,236
13,500	0/37	0/60	0/667	0/325	0/180	0/66	0/38	0/26	0/21	0/13	0/4	0/9,236

II. Response to Reviewer 2

The authors would like to express sincere thanks to the reviewer for the thorough and constructive comments. In what follows, we present detailed comments in response to the individual points raised by the reviewer and elaborate on how the manuscript has been revised.

Comment 2.1: There are many studies concerning p2p botnet detection. This paper proposed enhanced PeerHunter based on community behavior analysis. The authors proposed mutual-contacts graph to capture the intrinsic nature of p2p botnet indistinguishable from legitimate p2p applications. Following such mutual-contacts graph based netflow level traffic analysis, enhanced PeerHunter could certainly detect p2p botnets in the waiting stage, in the face of encrypted traffic, without the need to label the seeds, at the network boundary, etc.

Overall, the idea makes sense. The paper is well organized and written, thus easy to follow. The authors take full advantage of mutual-contacts graph and achieve high detection rate with very few false positives.

Response 2.1: We thank the reviewer for the positive comments.

Comment 2.2: However, the practical problem below, which I believe is critical in p2p botnet detection, should be further addressed.

When the monitored internal network is too small, it would be challenging to build mutual-contacts graph due to lack of sample nodes and edges. As the monitored internal network becomes large, such a challenge gradually disappears. However, when the monitored internal network is too large, new challenges appear. Specifically, some mutual contacts may be within the internal network, hence invisible to the monitor at the network boundary. In addition, the performance scalability of traffic processing would be a big concern when the monitored internal network is too large, since the traffic would be several hundreds GB/s. Therefore, where should one deploy enhanced PeerHunter in the face of a p2p botnet that might be (sparsely) distributed all over the Internet?

Response 2.2: Thank you for the insightful comment. In this revision, we added a new section, VI-B, to discuss about the possible issues and solutions involved in the deployment of Enhanced PeerHunter, as follows.

“

A. The deployment of Enhanced PeerHunter

In the previous description, we simply assume that our system is deployed at the boundary of a single organization. In this section, we discuss about the deployment of Enhanced PeerHunter in three more realistic scenarios.

1) The number of bots within an organization is too small: It would be challenging to build the MCG of botnet communities (i.e., the number of bots belonging to the same botnet is less than 3). In this case, we can deploy multiple Enhanced PeerHunter systems at the boundaries of multiple organizations, and correlate the network flows collected by those multiple Enhanced PeerHunter systems to build an appropriate size of MCG to detect botnet communities.

2) The number of bots within an organization is too large: The mutual contacts of certain bots might be within the organization internal network, hence invisible to the single system monitored at the network boundary. In this case, we can deploy multiple Enhanced PeerHunter systems within the organization, that divide the organization network into several appropriate size of sub-internal networks. Each system is responsible for one sub-internal network.

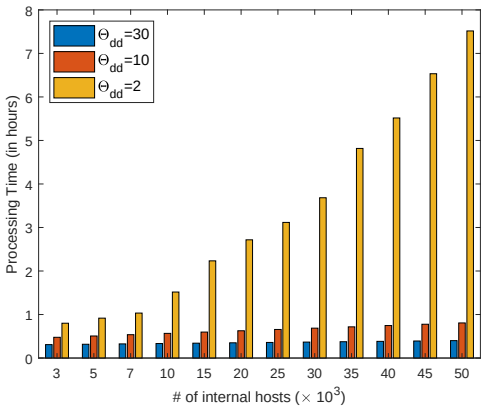

Fig. 4: Processing time with different data size and Θ_{dd}.

3) The botmaster knows the system deployment location: In this way, the botmaster could assign the location of bots or control the communications of the bots based on the knowledge of the system deployment location to evade our system. For instance, the botmaster could assign bots into different sub-internal networks, and urge most of the bots communicate with the others within the same sub-internal network. In this case, we could use the concept and idea of Moving Target Defense (MTD) [49] to develop a strategy that makes it more difficult for botmasters to learn the deployment locations of our systems, by dynamically changing the settings or deployments of single/multiple Enhanced PeerHunter systems.
”

We also tested our system using different sizes (i.e., different number of internal hosts) of EDs. In the following, we list a new part of section, V-E2, in addition to the new figure, Fig. 4, which were added to address the stability of our system while deploying on different size on networks.

“ We also tested our system using different sizes (i.e., different number of internal hosts) of EDs. For each size, we generate 10 EDs, and record the average processing time of our system with different Θ_{dd}. As shown in Fig. 4, compared with the size of datasets, Θ_{dd} has more influence on the system scalability. Because in our P2P network flow detection component, Θ_{dd} has an impact on n (the number of P2P network flow clusters subject to analysis), and larger Θ_{dd} leads to smaller n, thus less processing time. For instance, when $\Theta_{dd} = 10$ and $\Theta_{dd} = 30$, the increase of processing time because of increasing data size is much less than when $\Theta_{dd} = 2$. Therefore, our system is very scalable on different sizes of data with an appropriate parameter ($\Theta_{dd} = 30$). Also, by tuning Θ_{dd}, our system has the potential to deal with different size of datasets in a reasonable time.
”

Comment 2.3: The idea of mutual-contacts graph might be applicable for other types of botnets (e.g., centralized botnets like IRC botnet [1], mobile botnets [2]), since, for example, bots of a centralized botnet are likely to contact a similar set of C&C or satellite servers. The authors should discuss the possibilities to extend the idea to detect these botnets.

[1] X. Ma et al., "A Novel IRC Botnet Detection Method Based on Packet Size Sequence," 2010 IEEE International Conference on Communications, Cape Town, 2010

[2] Shuang Zhao, Patrick P. C. Lee, John C. S. Lui, Xiaohong Guan, Xiaobo Ma, and Jing Tao. 2012. Cloud-based push-styled mobile botnets: a case study of exploiting the cloud to device messaging service. In Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC '12). ACM, New York, NY, USA, 119-128

Response 2.3: Thank you for the insightful comment. In this revision, we added a new section, VI-C, to discuss about the potential of extending Enhanced PeerHunter to detect other botnets, as follows. And the two recommended papers are referred in this section.

"

B. Extend Enhanced PeerHunter to detect other botnets

Although Enhanced PeerHunter is designed to detect P2P botnets, our idea of using mutual contact graph has the potential to detect not only unknown botnets, but also the other types of botnets (e.g., centralized botnets, such as IRC botnets [31], mobile botnets [50]). Since bots are usually controlled by machines, rather than humans, bots from the same botnets tend to communicate with a similar set of peers or attacking targets. For instance, bots from the same IRC botnets tend to contact a similar set of C&C servers, while bots from the same mobile botnets tend to contact a similar set of satellite servers. Hence, we argue that Enhanced PeerHunter could be easily extended to detect the other types of botnets.

"

Comment 2.4: Moreover, the mutual-contacts graph, in concept, is constructed based on bipartite graphs via similarity measures, thereby similar to the coexistence graph in [3]. The authors should also reference [3] in related work.

[3] Xiaobo Ma, Junjie Zhang, Jing Tao, Jianfeng Li, Jue Tian, and Xiaohong Guan. 2014. DNSRadar: Outsourcing Malicious Domain Detection Based on Distributed Cache-Footprints. Trans. Info. For. Sec. 9, 11 (November 2014), 1906-1921.

Response 2.4: Thank you for the comment. In this revision, we added the recommended reference into the related work section, II, as "Similar to the idea of using mutual contacts graph, Ma et al. [35] proposed to use the coexistence of domain cache-footprints distributed in networks that participate in the outsourcing service (i.e., coexistence graph) to detect malicious domains."

III. Response to reviewer 3

The authors would like to express sincere thanks to the reviewer for the thorough and constructive comments. In what follows, we present detailed comments in response to the individual points raised by the reviewer and elaborate on how the manuscript has been revised.

Comment 3.1: This paper describes an approach to detect P2P botnet based on network-flow level botnet community behavior. The basic idea is to start from a P2P network flow detection component, and use the natural botnet behavior mutual contacts as the main feature to cluster bots into communities. Then network-flow level botnet community behavior analysis is used to detect potential botnet communities. The evaluation results show that this approach can achieve a relative high accuracy. In addition to the conference version, authors added some new contents and aim at providing a robust approach against the mimicking legitimate P2P application attacks.

#Strength:

1. The writing of this paper is clear and easy to follow.
2. The analysis on community behavior is full.
3. The additions upon conference version are necessary.

Response 3.1: We thank the reviewer for the positive comments.

Comment 3.2: The designed method only use mutual contacts graph (MCG) as the baseline and use Louvain method to divide communities and further implement P2P botnet detection. In my opinion, the MCG method is too simple, and it doesnt take full account of other characters of a community other than mutual contacts. If they dont find mutual contacts from the traffic data, this method will be frustrated.

Response 3.2: Thank you for the comment. We agree with the reviewer about "If they dont find mutual contacts from the traffic data, this method will be frustrated.". However, based on our literature and empirical study, mutual contacts is the natural characteristic of P2P botnets. In the following, we list a part of the revised Section III-B, Section III-C, Table I and Table II, which were revised to address this comment.

“ . . . , Mutual contacts are 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 [11]. Because bots within the same P2P botnet tend to receive and search for the same C&C messages from the same set of botmasters (peers) [39]. Moreover, in order to prevent peers from churning in a P2P botnet, the botmaster must check each bot periodically, which results in a convergence of contacts among peers within the same botnet [6]. 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 some very popular servers, such as google.com, facebook.com [11]. 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 practice, legitimate P2P hosts with different purposes will not search for the same set of peers. As such, we can use mutual contacts to cluster the bots within the same botnet, and distinguish between P2P botnets and legitimate P2P applications. . . .

”

“ . . . , **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) the MCR of a pair of bots within the same botnet is much higher than the MCR of a pair of legitimate applications or a pair of 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 have much higher AVGMCR. . . .

”

TABLE I: Notations and Descriptions

Notations	Descriptions
MNF	the management network flow
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 contacts ratio

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%

Other than using mutual contact, we also used the concept of "destination diversity" to capture the "P2P behavior" of P2P botnet communities, which has been described in part of Section III-C, as follows.

" . . . , **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 its network flow. The destination diversity ratio (DDR) of each host is its DD divided by the total number of distinct destination IPs of its network flow.

Due to the decentralized nature of P2P networks, P2P network flow tend to have higher DDR than non-P2P network flow. Furthermore, network flow from P2P botnet communities usually have higher average DDR (AVGDDR) than network flow from legitimate network communities. Network flow 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 flow are usually user-dependent, 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.

Table II shows the number of distinct destination IP /16 prefixes in MNF of each type of P2P host, where both legitimate hosts and bots spread across many 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.. . .

"

Comment 3.3: The idea of using group or community behavior to detect P2P botnets is not new. There are many papers in the past few years discussing methodologies to detect P2P botnets. The paper did not thoroughly discuss the difference between their method with related works in the literature. The lack of comparison with other previous works about P2P botnet detection makes the effects of the methods adopted in this paper are not persuasive. It can be further improved in the methodology and evaluation.

Response 3.3: Thank you for the insightful comment. In this revision, we added new section, V-G, to compare our system with one of the state of art P2P botnet detection system Zhang et al. [6] (the upgraded version of Zhang et al. [18]). In the following, we list the new section, V-G, in addition to the revised Table XIV, which were added to address this comment.

“

G. Comparison to Zhang et al. [6]

We compare our system to one of the state of art P2P botnet detection system Zhang et al. [6] (the upgraded version of Zhang et al. [18]). Zhang et al. [6] propose a scalable botnet detection system capable of detecting stealthy P2P botnets (i.e., in the waiting stage), where no knowledge of existing malicious behaviour is required in advance. The system first applies a two-step flow clustering approach to create the fingerprints of hosts that have engaged in P2P activities. Afterwards, it applies two layers of filtering to detect potential P2P bots: a coarse-grained filtering to detect “persistent” P2P hosts that have longer active time of P2P behaviors, and a fine-grained filtering that applies hierarchical clustering to group pairs of P2P hosts that have less distance between their fingerprints. Our system shares many similarities with Zhang et al. [6]. For instance, both systems are (a) using network flow-based approach, (b) using unsupervised approach (i.e., no knowledge of existing malicious behaviours are required and have the potential to detect unknown botnets), (c) claiming to work while the botnet traffic are overlapped with the legitimate P2P traffic on the same host, (d) designing to have the built-in scalability, and (e) deployed at the network boundary (e.g., gateway), thus could be evaluated on the same datasets.

The main differences between our system and Zhang et al. [6] are listed as follows. First, two systems are using different network flow features. Zhang et al. [6] uses the absolute number of bytes and packets of each flow; Enhanced PeerHunter uses the bytes-per-packet rate of each flow. Second, two systems are using different approach to cluster network flows (at different granularity). Zhang et al. [6] uses a two-step distance-based clustering (i.e., k-means, BIRCH) to cluster network flows of similar feature values; Enhanced PeerHunter clusters the network flows that have exactly the same feature value. Third, two systems apply the botnet detection step at different levels (i.e., host-level or network-flow-level). Zhang et

al. [6] uses the distance between each pair of hosts to detect bots; Enhanced PeerHunter uses the distance between each pair of network flows to detect botnet network flow communities and then further identify the corresponding bots. Last but not least, two systems are using different heuristics to detect botnets. Zhang et al. [6] uses an threshold on the height of the hierarchical clustering dendrogram to detect bot clusters, which is very sensitive to the experimental datasets (as shown in Table XIV); Enhanced PeerHunter uses network-flow level community behavior analysis (i.e., AVGDDR and AVGMCR) to identify botnet (network flow) communities, which is more robust to the proposed attacks and can also be extended to other/new community behaviors.

We implement a prototype system of Zhang et al. [6], since Zhang et al. [6] does not have an public available implementation. Most of our implementations follow the description as in [6], other than the system parallelization, which has zero impact on the system effectiveness evaluation. The experimental datasets used in both works are also different. For instance, we evaluate our system on 100 synthetic experimental datasets (of different background traffic and different topology, as described in Section V-A3) and take the average results; Zhang et al. [6] was evaluated on single customized dataset. Furthermore, even though both datasets are using the same 24 hours time window, our datasets have much more internal hosts (i.e., 10,000 vs. 953), higher legitimate P2P hosts to P2P bots ratio (i.e., 727:37 vs. 8:16), and more types of botnets (i.e., 5 vs. 2). To summarize, our experimental datasets is more challenging and comprehensive.

We applied our implementation Zhang et al. [6] on the same experimental datasets as Enhanced PeerHunter under two circumstances (i.e., No Attack and PMMKL). We follow the same value for most of the system parameters as described in [6], such as $\Theta_{BGP} = 50$, $\Theta_{p2p} = 0.5$, $K = 4,000$, $\lambda = 0.5$. Since the default value of Θ_{bot} (i.e., 0.95) used by the original paper, did not perform well on our dataset, we evaluated Zhang et al. [6] using two other different well selected values of Θ_{bot} (i.e., 0.6 and 0.8) that shows better results.

From the experimental results (Table XIV), we achieve several observations as follows. First, Zhang et al. [6] is more sensitive to the experimental dataset. For instance, Zhang et al. [6] is reported to achieve 100% detection rate and 0.2% false positive rate on their own datasets (using $\Theta_{bot} = 0.95$), while could not achieve similar results on our datasets using either the default parameter ($\Theta_{bot} = 0.95$) or the well selected parameter ($\Theta_{bot} = 0.6$ or $\Theta_{bot} = 0.8$). Second, as discussed in Section V-E, our system is stable and effective over a large range of system parameters (Θ_{avgddr} and Θ_{avgmcr}), while Zhang et al. [6] is more sensitive to its system parameter (Θ_{bot}). For instance, Zhang et al. [6] has higher precision (lower false positives) and lower recall (higher false negatives) while using $\Theta_{bot} = 0.6$ comparing with using $\Theta_{bot} = 0.8$. Third, our system outperforms Zhang et al. [6] in terms of the detection rate of different

TABLE XIV: Comparison of the botnet detection results under no attack and PMMKL attack. (* detection rate)

-	PeerHunter [10]		Enhanced PeerHunter		Zhang et al. [6] ($\Theta_{bot} = 0.6$)		Zhang et al. [6] ($\Theta_{bot} = 0.8$)	
	No Attack	PMMKL	No Attack	PMMKL	No Attack	PMMKL	No Attack	PMMKL
ZeroAccess*	100%	0%	100%	100%	100%	82.5%	100%	90%
Waledac*	100%	0%	100%	100%	100%	0%	100%	60%
Storm*	100%	37.5%	100%	100%	97.8%	61.5%	100%	95.4%
Kelihos*	100%	0%	100%	100%	85.5%	45%	85.5%	77.5%
Sality*	100%	79.2%	100%	100%	89.6%	80%	96.8%	88%
Precision	100%	99.1%	100%	100%	100%	100%	60.8%	62.7%
Recall	100%	23.9%	100%	100%	94.7%	60%	96.4%	86.5%
FP	0/9,963	39/9,926	0/9,963	0/9,926	0/9,963	0/9,926	23/9,963	19/9,926
F-score	100%	38.5%	100%	100%	97.3%	75%	74.6%	72.7%

botnets, the overall precision, recall and false positives. For instance, our system achieve 100% detection rate with zero false positives under different circumstances, while Zhang et al. [6] fails to detect all the bots under both well selected parameters. At last, our system is more robust to PMMKL attack. For instance, PMMKL attack has no impact on the effectiveness of our system, while decreasing the F-score of Zhang et al. [6] from 97.3% to 75% ($\Theta_{bot} = 0.6$) or from 74.6% to 72.7% ($\Theta_{bot} = 0.8$).

”

Comment 3.4: Most of the references are prior to 2015, and there are few references for recent years. It is recommended that the authors update their references and add latest works.

Response 3.4: Thank you for the comment. According to your recommendation, we updated our references for the most recent years in the related work. For instance, we added the following recent works into our related work section.

[29] S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao, and M. Conti, Detecting android malware leveraging text semantics of network flows, IEEE Transactions on Information Forensics and Security, vol. 13, no. 5, pp. 10961109, 2018.

[32] S. Khanchi, A. Vahdat, M. I. Heywood, and A. N. Zincir-Heywood, On botnet detection with genetic programming under streaming data label budgets and class imbalance, Swarm and evolutionary computation, vol. 39, pp. 123140, 2018.

[49] M. Albanese, S. Jajodia, and S. Venkatesan, Defending from stealthy botnets using moving target defenses, IEEE Security & Privacy, vol. 16, no. 1, pp. 9297, 2018.

[21] J. Wang and I. C. Paschalidis, Botnet detection based on anomaly and community detection, IEEE Transactions on Control of Network Systems, vol. 4, no. 2, pp. 392404, 2017.

[22] S. Venkatesan, M. Albanese, A. Shah, R. Ganesan, and S. Jajodia, Detecting stealthy botnets in a resource-constrained environment using reinforcement learning, in Proceedings of the 4th ACM Workshop on Moving Target Defense, 2017, pp. 7585.

[23] S. Karuppayah, L. Bock, T. Grube, S. Manickam, M. Muhlhauser, and M. Fischer, Sensorbuster: On identifying sensor nodes in p2p botnets, in Proceedings of the 12th International Conference on Availability, Reliability and Security. ACM, 2017, p. 34.

[24] F. Haddadi and A. N. Zincir-Heywood, Botnet behaviour analysis: How would a data analytics-based system with minimum a priori information perform? International Journal of Network Management, vol. 27, no. 4, p. e1977, 2017.

[36] W. Chen, X. Luo, and A. N. Zincir-Heywood, Exploring a service-based normal behaviour profiling system for botnet detection, in Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on. IEEE, 2017, pp. 947952.

[33] F. Haddadi and A. N. Zincir-Heywood, Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification, IEEE Systems journal, vol. 10, no. 4, pp. 13901401, 2016.

[34] J. Jianguo, B. Qi, S. Zhixin, Y. Wang, and B. Lv, Botnet detection method analysis on the effect of feature extraction, in Trustcom, 2016 IEEE. IEEE, 2016, pp. 18821888.

Comment 3.5: The author's analysis of network traffic is too coarse-grained. It is recommended that authors add some references on malware detection using traffic flow to explain the meaning of network-flow level behavior analysis. In addition, it is suggested that the authors add some new methods discussion on botnet behavior analysis and defense.

—-Wang S, Yan Q, Chen Z, et al. Detecting Android Malware Leveraging Text Semantics of Network Flows[J]. IEEE Transactions on Information Forensics & Security, 2018, 13(5):1096-1109.

—- Albanese M, Jajodia S, Venkatesan S. Defending from Stealthy Botnets Using Moving Target Defenses[J]. IEEE Security & Privacy, 2018, 16(1):92-97.

—- Haddadi F, Zincir-Heywood A N. Botnet behaviour analysis: How would a data analyticsbased system with minimum a priori information perform? [J]. International Journal of Network Management, 2017:e1977.

Response 3.5: Thank you for the comment. According to your recommendation, we revised the related work by discussing more about flow-based malware/botnet detection, and using different features to capture the botnet behavior. All three recommended references have been added to the related work. In the following, we list a part of the revised Section II to address this comment.

" . . . , Flow-based systems [6], [8][11], [13], [16], [18], [19], [21][24], [31], [32] use header information of network packets (i.e., network flow characteristics) to capture botnets behaviors. Compared with payload-

based systems, flow-based systems use less information from the network packets. However, since most recent stealthy botnets tend to use encryption to hide the payload information [24], [33], designing flowbased systems becomes more important and necessary. Some flow-based systems apply one or several different supervised machine learning algorithms on a set of well extracted network flow features to model the botnets behaviors [34]. For instance, Jianguo et al. [34] applied three supervised machine learning algorithms (i.e., SVM, Logistic Regression and Neural Network) on network flow features extracted from Netmate and Tranalyzer to detect botnet. They obtained very high performance metrics, while employing a fully labelled dataset. Khanchi et al. [32] proposed an approach using genetic programming and ML on data streams to detect botnet flows. However, since most of the supervised ML-based approaches usually generate models that are focusing on specific types of botnets (existing in the training data), those approaches will not be effective to detect botnets not appeared in the training data (unknown botnets).

Some other most recent flow-based systems tend to use a combination of different heuristics to model P2P botnets behaviors. For instance, Botgrep [16] proposed to detect P2P botnets through localizing structured communication graphs, where they found that the communication graph of P2P applications have fast convergence time of random walks to a stationary distribution. However, their method can only identify structured communication subgraphs, rather than ensure those subgraphs contain P2P botnets. Entelecheia [8] proposed to use a synergistic graph-mining approach on a super-flow graph built from network flow features (i.e., volume per hour, duration per flow) to identify a group of P2P bots, where they claimed that P2P botnet network flows tend to have low volume and long duration. Group or community behavior based methods [9][11], [21] consider the behavior patterns of a group of bots within the same P2P botnet community. Coskun et al. [11] developed a P2P botnets detection approach that start from building a mutual contacts graph of the whole network, then attempt to use “seeds” (known bots) to identify the rest of bots within the same botnet. However, it is impractical to have a seed in advance. Similar to the idea of using mutual contacts graph, Ma et al. [35] proposed to use the coexistence of domain cache-footprints distributed in networks that participate in the outsourcing service (i.e., coexistence graph) to detect malicious domains. Yan et al. [9] proposed a group-level behavior analysis based P2P botnets detection method, where they start from clustering P2P hosts into groups, and then use supervised machine learning methods (e.g., SVM) to identify bots through a set of group-level behavior features. Since their approach relies on supervised classification methods (e.g., SVM) which requires to train the model of each botnet on fully labelled dataset in advance, it will be hard for their method to cope with detecting unknown P2P botnets, which is the common case in botnet detection [2]. Chen et al. [36] applied three unsupervised machine learning algorithms, self-organising map (SOM), local outlier factor (LOF), and k-NN outlier, to build a normal behaviour profile to be used for botnet detection. They could obtain a very

high detection rate (91.3rates due to the nature of the unsupervised ML algorithms employed. PeerHunter [10], our previous work, propose to use the host level community behavior analysis to detect P2P botnets, which does not consider the scenario that the P2P bots and legitimate P2P applications are ruining on the same compromised host at the same time. As such, this previous work does not very robust to the mimicking legitimate P2P application attacks. Zhang et al. [6] propose a scalable botnet detection system capable of detecting stealthy P2P botnets (i.e., in the waiting stage), where no knowledge of existing malicious behaviour is required in advance. Zhang et al. [6] also claimed to work in the scenario that the botnet traffic are overlapped with the legitimate P2P traffic on the same host. However, their experimental dataset is kind of biased and less challenging. For example, in their dataset, the number of bots (i.e., 16) is twice as many as the number of legitimate P2P hosts (i.e., 8), which is easier for bots to form clusters and harder for legitimate P2P hosts to form clusters.

In this work, we propose Enhanced PeerHunter (i.e., extended version of PeerHunter [10]), a network-level flowbased system that relies on community behavior analysis to detect P2P botnets. We compare Enhanced PeerHunter with PeerHunter [10] and Zhang et al [6] on a more challenging and comprehensive experimental datasets, and show that our system outperforms both systems in terms of detection rate, false positives and performance under the mimicking legitimate P2P application attacks (Section V).

. . .

”

IV. Response to reviewer 4

The authors would like to express sincere thanks to the reviewer for the thorough and constructive comments. In what follows, we present detailed comments in response to the individual points raised by the reviewer and elaborate on how the manuscript has been revised.

Comment 4.1: This paper studies the problem of P2P botnet detection. The authors utilize network traffic flow to form mutual contact graph, which is then used for community detection and bot detection. The used P2P network features and mutual contact features are not new. This paper is an extended version of their conference paper. However, the difference between the conference version and journal version is too minor.

Comparing Section III and IV in the conference and journal version of the paper, the difference lies in the host-based MCG and network-flow based MCG. All the algorithms are almost identical, except the minor difference of host and flow cluster. I dont see any improvement except this minor modification.

Response 4.1: Thank you for the comment. After this revision, we extended our conference paper from 8

pages to 16 pages. Other than what have been stated in our "Summary of Changes" in the latest revision, the following summarizes the extensions/enhancements we had made beyond our conference paper.

1) We use the network-flow level mutual contacts instead of the host level mutual contacts to build the mutual contact graph, which relates the P2P management network flow to the P2P applications instances generating them instead of the host running those applications instances. By doing this, Enhanced PeerHunter can separate the P2P botnets (management network flow) and legitimate P2P applications (management network flow), even if the P2P bots and legitimate P2P applications are ruining on the same compromised host/machine at the same time.
2) Because of using the network-flow level mutual contacts, in the community detection results, the communities could be overlapped on the same set of hosts. As such, we propose three new criterions (i.e., Bot Separation Index, Bot Aggregation Index and Bot-Legitimate Separation Index) to evaluate the community detection performance.
3) We proposed two mimicking legitimate P2P application attacks (one passive attack and one active attack). Both attacks aim to make the P2P bots behavior more like legitimate P2P applications. Compared with our previous work, this work is much more robust facing those attacks. For instance, the passive attack could reduce the accuracy of PeerHunter, but has no effects on Enhanced PeerHunter. And, although Enhanced PeerHunter could not completely mitigate the active attack, but it will make it rather expensive and exposed for the adversaries to conduct the active attack.
4) We did extra extensive experiments in our experimental evaluation to better evaluate our system. For instance, Section V-E3 and Fig. 5 were added to discuss about the effectiveness and sensitiveness of our system while using different combinations of system parameters. Section V-E4 was added to discuss about our interesting findings of the "true" false positives, which demonstrated that our system has the potential to detect other unknown botnets. Section V-G was added and Table XIV was revised to compare our system with one of the state of art P2P botnet detection system Zhang et al. [6].
5) A discussion section (Section VI) has been added to discuss about (1) evasions and possible solutions (Section VI-A), (2) the deployment of Enhanced PeerHunter, and (3) the potential of extending Enhanced PeerHunter to detect other botnets.

Comment 4.2: The new addition compared with conference paper is only the two evasion attacks. Yet, the authors did not provide any insights on why the proposed method is robust against these attacks. In fact, the active attack is very effective in dropping its performance. This makes the new extension of the paper insignificant.

Response 4.2: Thank you for the comment. We revised Section V-F, to discuss more about the insights on why our system is robust against these attacks. The following list the part of our revision in Section V-F1 to address the robustness of our system against PMMKL (i.e., the passive attack).

" As such, the botnet traffic will be overlapped with the legitimate P2P traffic. Since during most of the time, P2P botnets will be acting stealthily, the legitimate P2P traffic will dominate the host level behavior. Hence, the attack could effectively evade the host level group behavior based methods [9], [10]. Also, the attack does not require the botnets to generate more or new types of network flows, and just need to monitor the legitimate P2P application activities, which can evade certain anomaly-based methods. Since our detection algorithm is based on network-flow level mutual contacts graph, which could differentiate the network flows coming from different P2P applications, it is capable of detecting P2P bots while the bots traffic and the legitimate P2P traffic are overlapped on the same host.
"

In AMMKL (i.e., the active attack), to evade our system, botnets need to conduct the attack with the sacrifice of its stealthiness and efficiency, which would be the victory of our system. The following list the part of our revision in Section V-F2, Table XV, and the new Section VI-A to address the robustness of our system against AMMKL.

" Table XV shows the analysis of all 5 botnets. Take Storm for instance, to affect the detection of Storm, each P2P network flow cluster of Storm needs to communicate with at least an extra 40% of its current destination peer IPs, and in order to completely evade our system, γ needs to be increased to 80%. Consider the fact that each Storm host generates an average of 67 P2P network flow clusters in 24 hours, and each network flow cluster communicates to an average of 740 destination peer IPs. As such, to completely evade our system, each Storm host must communicate with at least an extra of $67 \times 740 \times 80\% \approx 39,664$ peers, which makes it less stealthy, less efficient and more exposed to trigger anomaly-based P2P botnet detection [48]. In conclusion, although our system could not completely mitigate AMMKL, it makes the botnets rather expensive and exposed to perform AMMKL.
"

TABLE XV: Effort needed for different P2P botnets to completely evade our system under AMMKL.

-	# of P2P flow clusters	# of peers per flow cluster	# of peers per host	γ	extra # of peers needed
ZeroAccess	3	686	2,058	220%	4,528
Waledac	171	244	41,724	180%	75,104
Storm	67	740	49,580	80%	39,664
Kelihos	15	252	3,780	200%	7,560
Sality	1,158	918	1,063,044	80%	850,436

“

A. Evasions and Possible Solutions

To avoid detection by Enhanced PeerHunter, the botmaster could use a combination of the following three approaches: (a) add randomized paddings or junk packets to influence the bytes-per-packet characterizes for network flow clustering, (b) reduce the number or rate of destination diversity, or (c) reduce the number or rate of mutual contacts. To deal with the randomized spatial-communication behavior, we could adopt more time-communication features, such as packet/flow duration and inter-packet delays, or apply more general features, such as the distribution, mean or standard deviation of bytes-per-packet. The other two evasion approaches would be the victory of our system. On one hand, to reduce the number or rate of destination diversity, a bot has to limit its communication to the network of certain locations, which degrade the P2P botnet into centralized botnet to a certain extent. On the other hand, as shown in V-F, reducing the number or rate of mutual contacts is expensive and will make botnets easier to expose themselves.

”

Comment 4.3: Some additional comments:

Figure 1 still shows a host-level MCG, while the paper is trying to provide a network flow-level MCG.

Response 4.3: Thank you for the comment. Fig. 1 is used to illustrates the general concept of mutual contact, and Fig. 3 (shown below) gives an example of network-flow level mutual contacts graph extraction and community detection. We also clarified this by revising the last sentence of Section III-B as “More details about network-flow level MCG is discussed in Section IV-B.”.

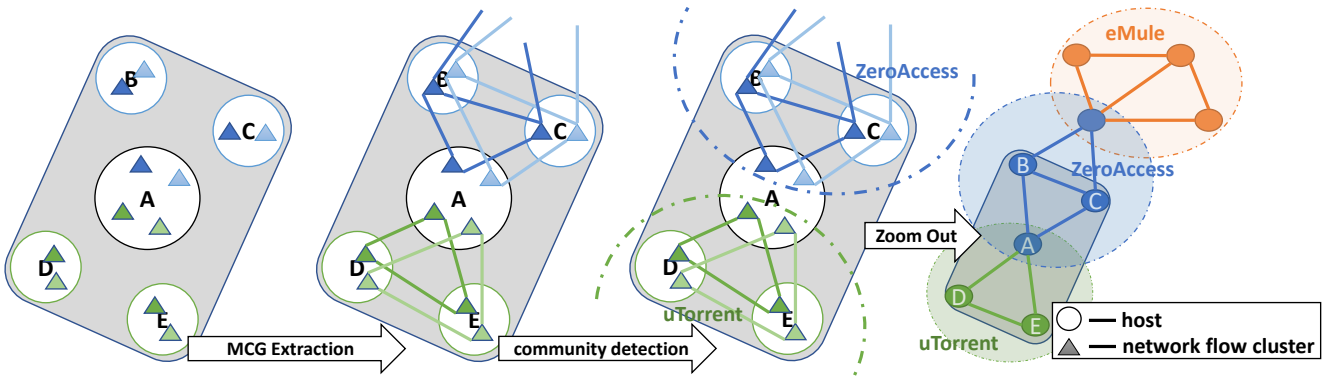

Fig. 3: An example of network-flow level mutual contacts graph extraction and community detection. Each triangle represents a network flow cluster, and the same color triangles represent the same type of network flow clusters. The areas divided by the dash-dot line with different color represents different communities.

Comment 4.4: The authors mentioned clique detection is NP-complete, and they cannot apply such method. Yet, the method is used in Algorithm 3, which is very confusing.

Response 4.4: Thank you for the comment. To resolve the confusion, we revised and clarified the explanation about why we could not using clique detection **directly** to the whole dataset, and combining the other community behaviors could reduce the input size of the clique detection problem. The basic idea is that after applying the other components (i.e., the P2P host detection component, and the community detection in the botnet detection component), the number of host in each community would be reduced a lot, thus it would be more feasible to use clique detection (even though its still NP-complete). The following shows a part of the revised Section III-C3.

" . . . , 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 directly apply such method to detect botnets, without any preprocessing. Therefore, we propose to combine all three botnet community behaviors, and use the previous two community behaviors as the "preprocessing" to reduce the input size of the clique detection problem. . . .
"

Comment 4.5: The language of the paper can be further improved.

Response 4.5: Thank you for the comment. The paper has been revised thoroughly to improve the writing.

V. Response to reviewer 5

The authors would like to express sincere thanks to the reviewer for the thorough and constructive comments. In what follows, we present detailed comments in response to the individual points raised by the reviewer and elaborate on how the manuscript has been revised.

Comment 5.1: The paper studies the problem of identification and detection of P2P bots. Authors presented a network-flow level botnet community behavior analysis based method to detect botnets that communicate via P2P overlay networks, named Enhanced PeerHunter. Compared to the previous version of the published conference paper, I see some improvement, but not significant. The paper presented well and written soundly.

Response 5.1: We thank the reviewer for the positive comment.

Comment 5.2: My main sustainable negative comments are that 1) the paper doesnt include any recent studies from after 2015 (except for 1 conference paper in 2017), that gives indication that the authors is not aware of the recent developments in this fast grown field.

The review of related work misses several important papers and fails to establish how the proposed approach differs in any significant way.

Response 5.2: Thank you for the comment. According to your recommendation, we updated our references for the most recent years in the related work. For instance, we added the following recent works into our related work section.

[29] S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao, and M. Conti, Detecting android malware leveraging text semantics of network flows, IEEE Transactions on Information Forensics and Security, vol. 13, no. 5, pp. 10961109, 2018.

[32] S. Khanchi, A. Vahdat, M. I. Heywood, and A. N. Zincir-Heywood, On botnet detection with genetic programming under streaming data label budgets and class imbalance, Swarm and evolutionary computation, vol. 39, pp. 123140, 2018.

[49] M. Albanese, S. Jajodia, and S. Venkatesan, Defending from stealthy botnets using moving target defenses, IEEE Security & Privacy, vol. 16, no. 1, pp. 9297, 2018.

[21] J. Wang and I. C. Paschalidis, Botnet detection based on anomaly and community detection, IEEE Transactions on Control of Network Systems, vol. 4, no. 2, pp. 392404, 2017.

[22] S. Venkatesan, M. Albanese, A. Shah, R. Ganesan, and S. Jajodia, Detecting stealthy botnets in a resource-constrained environment using reinforcement learning, in Proceedings of the 4th ACM Workshop on Moving Target Defense, 2017, pp. 7585.

[23] S. Karuppayah, L. Bock, T. Grube, S. Manickam, M. Muhlhauser, and M. Fischer, Sensorbuster: On identifying sensor nodes in p2p botnets, in Proceedings of the 12th International Conference on Availability, Reliability and Security. ACM, 2017, p. 34.

[24] F. Haddadi and A. N. Zincir-Heywood, Botnet behaviour analysis: How would a data analytics-based system with minimum a priori information perform? International Journal of Network Management, vol. 27, no. 4, p. e1977, 2017.

[36] W. Chen, X. Luo, and A. N. Zincir-Heywood, Exploring a service-based normal behaviour profiling system for botnet detection, in Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on. IEEE, 2017, pp. 947952.

[33] F. Haddadi and A. N. Zincir-Heywood, Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification, IEEE Systems journal, vol. 10, no. 4, pp. 13901401, 2016.

[34] J. Jianguo, B. Qi, S. Zhixin, Y. Wang, and B. Lv, Botnet detection method analysis on the effect of feature extraction, in Trustcom, 2016 IEEE. IEEE, 2016, pp. 18821888.

Comment 5.3: 2) Compare methods with similar ones on the literature is important.

As I mentioned, comparison to other approaches is superficial and limited.

Response 5.3: Thank you for the comment. In this revision, we added new section, V-G, to compare our system to one of the state of art P2P botnet detection system Zhang et al. [6] (the upgraded version of Zhang et al. [18]). In the following, we list the new section, V-G, in addition to the revised Table XIV, which were added to address this comment.

“

G. Comparison to Zhang et al. [6]

We compare our system to one of the state of art P2P botnet detection system Zhang et al. [6] (the upgraded version of Zhang et al. [18]). Zhang et al. [6] propose a scalable botnet detection system capable of detecting stealthy P2P botnets (i.e., in the waiting stage), where no knowledge of existing malicious behaviour is required in advance. The system first applies a two-step flow clustering approach to create the fingerprints of hosts that have engaged in P2P activities. Afterwards, it applies two layers of filtering to detect potential P2P bots: a coarse-grained filtering to detect “persistent” P2P hosts that have longer

active time of P2P behaviors, and a fine-grained filtering that applies hierarchical clustering to group pairs of P2P hosts that have less distance between their fingerprints. Our system shares many similarities with Zhang et al. [6]. For instance, both systems are (a) using network flow-based approach, (b) using unsupervised approach (i.e., no knowledge of existing malicious behaviours are required and have the potential to detect unknown botnets), (c) claiming to work while the botnet traffic are overlapped with the legitimate P2P traffic on the same host, (d) designing to have the built-in scalability, and (e) deployed at the network boundary (e.g., gateway), thus could be evaluated on the same datasets.

The main differences between our system and Zhang et al. [6] are listed as follows. First, two systems are using different network flow features. Zhang et al. [6] uses the absolute number of bytes and packets of each flow; Enhanced PeerHunter uses the bytes-per-packet rate of each flow. Second, two systems are using different approach to cluster network flows (at different granularity). Zhang et al. [6] uses a two-step distance-based clustering (i.e., k-means, BIRCH) to cluster network flows of similar feature values; Enhanced PeerHunter clusters the network flows that have exactly the same feature value. Third, two systems apply the botnet detection step at different levels (i.e., host-level or network-flow-level). Zhang et al. [6] uses the distance between each pair of hosts to detect bots; Enhanced PeerHunter uses the distance between each pair of network flows to detect botnet network flow communities and then further identify the corresponding bots. Last but not least, two systems are using different heuristics to detect botnets. Zhang et al. [6] uses an threshold on the height of the hierarchical clustering dendrogram to detect bot clusters, which is very sensitive to the experimental datasets (as shown in Table XIV); Enhanced PeerHunter uses network-flow level community behavior analysis (i.e., AVGDDR and AVGMCR) to identify botnet (network flow) communities, which is more robust to the proposed attacks and can also be extended to other/new community behaviors.

We implement a prototype system of Zhang et al. [6], since Zhang et al. [6] does not have an public available implementation. Most of our implementations follow the description as in [6], other than the system parallelization, which has zero impact on the system effectiveness evaluation. The experimental datasets used in both works are also different. For instance, we evaluate our system on 100 synthetic experimental datasets (of different background traffic and different topology, as described in Section V-A3) and take the average results; Zhang et al. [6] was evaluated on single customized dataset. Furthermore, even though both datasets are using the same 24 hours time window, our datasets have much more internal hosts (i.e., 10,000 vs. 953), higher legitimate P2P hosts to P2P bots ratio (i.e., 727:37 vs. 8:16), and more types of botnets (i.e., 5 vs. 2). To summarize, our experimental datasets is more challenging and comprehensive.

We applied our implementation Zhang et al. [6] on the same experimental datasets as Enhanced

TABLE XIV: Comparison of the botnet detection results under no attack and PMMKL attack. (* detection rate)

-	PeerHunter [10]		Enhanced PeerHunter		Zhang et al. [6] ($\Theta_{bot} = 0.6$)		Zhang et al. [6] ($\Theta_{bot} = 0.8$)	
	No Attack	PMMKL	No Attack	PMMKL	No Attack	PMMKL	No Attack	PMMKL
ZeroAccess*	100%	0%	100%	100%	100%	82.5%	100%	90%
Waledac*	100%	0%	100%	100%	100%	0%	100%	60%
Storm*	100%	37.5%	100%	100%	97.8%	61.5%	100%	95.4%
Kelihos*	100%	0%	100%	100%	85.5%	45%	85.5%	77.5%
Sality*	100%	79.2%	100%	100%	89.6%	80%	96.8%	88%
Precision	100%	99.1%	100%	100%	100%	100%	60.8%	62.7%
Recall	100%	23.9%	100%	100%	94.7%	60%	96.4%	86.5%
FP	0/9,963	39/9,926	0/9,963	0/9,926	0/9,963	0/9,926	23/9,963	19/9,926
F-score	100%	38.5%	100%	100%	97.3%	75%	74.6%	72.7%

PeerHunter under two circumstances (i.e., No Attack and PMMKL). We follow the same value for most of the system parameters as described in [6], such as $\Theta_{BGP} = 50$, $\Theta_{p2p} = 0.5$, $K = 4,000$, $\lambda = 0.5$. Since the default value of Θ_{bot} (i.e., 0.95) used by the original paper, did not perform well on our dataset, we evaluated Zhang et al. [6] using two other different well selected values of Θ_{bot} (i.e., 0.6 and 0.8) that shows better results.

From the experimental results (Table XIV), we achieve several observations as follows. First, Zhang et al. [6] is more sensitive to the experimental dataset. For instance, Zhang et al. [6] is reported to achieve 100% detection rate and 0.2% false positive rate on their own datasets (using $\Theta_{bot} = 0.95$), while could not achieve similar results on our datasets using either the default parameter ($\Theta_{bot} = 0.95$) or the well selected parameter ($\Theta_{bot} = 0.6$ or $\Theta_{bot} = 0.8$). Second, as discussed in Section V-E, our system is stable and effective over a large range of system parameters (Θ_{avgddr} and Θ_{avgmcr}), while Zhang et al. [6] is more sensitive to its system parameter (Θ_{bot}). For instance, Zhang et al. [6] has higher precision (lower false positives) and lower recall (higher false negatives) while using $\Theta_{bot} = 0.6$ comparing with using $\Theta_{bot} = 0.8$. Third, our system outperforms Zhang et al. [6] in terms of the detection rate of different botnets, the overall precision, recall and false positives. For instance, our system achieve 100% detection rate with zero false positives under different circumstances, while Zhang et al. [6] fails to detect all the bots under both well selected parameters. At last, our system is more robust to PMMKL attack. For instance, PMMKL attack has no impact on the effectiveness of our system, while decreasing the F-score of Zhang et al. [6] from 97.3% to 75% ($\Theta_{bot} = 0.6$) or from 74.6% to 72.7% ($\Theta_{bot} = 0.8$).

”

Comment 5.4: 3) no statement of contribution, authors should state how their contribution are different to others.

Response 5.4: Thank you for the comment. According to your recommendation, we added the statement of contribution at the end of introduction section, as follows.

" To summarize, our work has the following contributions:

• We present a novel, effective and efficient network-flow level community behavior analysis based P2P botnet detection system, Enhanced PeerHunter, which is capable of detecting P2P bots under the following challenges: (a) botnets are in their waiting stage (no malicious activity); (b) the C&C channel has been encrypted (no deep-packet-inspection); (c) the botnet traffic are overlapped with legitimate P2P traffic on the same host; and (d) none statistical traffic patterns are known in advance (unsupervised).

• We proposed three types of community behaviors (i.e., flow statistical features, numerical community features and structural community features) that can be used to detect P2P botnets effectively.

• We experimented our system with a wide range of parameter settings. With the best parameter settings, our system achieved 100% detection rate with zero false positive.

• We proposed two evasion attacks (i.e., passive and active mimicking legitimate P2P application attacks), where we assume the adversaries know our techniques in advance, and they attempt to evade our system via instructing P2P bots to mimic the behavior of legitimate P2P applications. The experiment results show that Enhanced PeerHunter will not be affected by the passive attack, and will make it very hard and expensive for the adversaries to conduct the active attack.

• We compared Enhanced PeerHunter with PeerHunter [10] (i.e., our previous work) and Zhang et al. [6]. Extensive experiments have been conducted to show that (a) our system outperforms Zhang et al. [6] in terms of the detection rate of different botnets, the overall precision, recall and false positives, and (b) our system is more robust to MMKL attack compared with PeerHunter [10] and Zhang et al. [6].

"