2014 IEEE 17th International Conference on Computational Science and Engineering

PeerDigger: Digging Stealthy P2P Hosts through Traffic Analysis in Real-time

Jie He, Yuexiang Yang, Xiaolei Wang
College of Computer
National University of Defense Technology
Changsha, China
{hejie, yyx, wangxiaolei}@nudt.edu.cn

Chuan Tang, Yingzhi Zeng
Information Center
National University of Defense Technology
Changsha, China
{tangchuan, zengyingzhi}@nudt.edu.cn

***Abstract*—P2P technology has been widely applied in many areas due to its excellent properties. Some botnets also shift towards the decentralized architectures, since they provide a better resiliency against detection and takedown efforts. Besides, modern P2P bots tend to run on compromised hosts in a stealthy way, which renders most existing approaches ineffective. In addition, few approaches address the problem of real-time detection. However, it is important to detect bots as soon as possible in order to minimize their harm. In this paper, we propose PeerDigger, a novel real-time system capable of detecting stealthy P2P bots. PeerDigger first detects all P2P hosts base on several basic properties of flow records, and then distinguishes P2P bots from benign P2P hosts by analyzing their network behavior patterns. The experimental results demonstrate that our system is able to identity P2P bots with an average TPR of 98.07% and an average FPR of 1.5% within 4 minutes.**

Keywords—P2P network; bot detection; real-time; traffic analysis

I. INTRODUCTION

Peer-to-peer (P2P) technology becomes more and more popular in the past decades. Due to its high efficiency and robustness, it has been widely applied in file sharing, media streaming, and many other fields. The decentralized nature of P2P architecture overcomes the single-point-of-failure limitation of traditional centralized architecture. Modern malwares also started shifting towards P2P architecture, which provides a better resiliency against detection and take-down efforts [1-3, 7-11]. P2P botnet is now one of the most common types of P2P malwares [1]. It is a P2P network of compromised hosts (bots) that are remotely controlled by an attacker (botmaster) through a command and control (C&C) channel to perform malicious activities such as distributed denial-of-service (DDoS) attack, spamming, sensitive data theft, etc.

Identifying P2P bots is of great importance, and the primary problem is differentiating them from other benign P2P hosts. However, there are several challenges to accomplish this purpose. First, since P2P bots have the same network characteristics as benign P2P applications, their C&C traffic can be easily blended into the background P2P traffic, making it difficult to separate these two types of traffic. This challenge is further compounded by the fact that a benign P2P application may coexist with a P2P bot on an infected host. Second, in contrast to benign P2P applications, P2P bots prefer to stealthily run on the compromised hosts in the background without users' perception [4]. Furthermore, they also become stealthier in the way they perform malicious activities, which are extremely difficult to be observed in the network traffic. For example, some P2P botnets turn to social engineering as an infection vector instead of scanning, and send spam through large popular webmail services such as Hotmail [5]. We refer to a stealthy P2P host as a P2P bot whose network behaviors are stealthy from user's perspective. Third, P2P botnets often obfuscate or encrypt their communication to evade payload-based detection [6], as is in case with Storm [1-3], Waledac [7-9], Zeus [10], and Conficker [11].

In this work, we present PeerDigger, a novel real-time system capable of detecting stealthy P2P bots inside a network perimeter through traffic analysis. PeerDigger collects flow records at the border of a monitored network and identifies internal hosts that are compromised by P2P bots in a two-step scheme. In the first step, based on several basic properties of flow records, all hosts that engage in P2P communications (either benign or malicious) are identified. In the second step, PeerDigger differentiates P2P bots from other benign P2P hosts by analyzing their network behavior patterns. In contrast to existing contemporary works, PeerDigger is featured by four characteristics. First, PeerDigger focuses on the behavioral characteristics of the network traffic with no access to payload of individual packets, and so can be unaffected by encryption or obfuscation of payload contents. Second, PeerDigger is able to detect P2P bots even if their malicious activities are stealthy and non-observable. Third, different from previous traffic-based approaches, PeerDigger does not employ any complicated statistical features or any additional sophisticated algorithms such as machine learning. Besides, our system does not need any training phase, so none labeled P2P botnet traffic trace is needed which is hard to obtain. Thus, PeerDigger is capable of real-time processing because of this well cost-effective scheme and its short detection time windows. Finally, most previous approaches based on traffic analysis would fail if there were only one single bot of a botnet within a network perimeter, or when a bot coexists with a benign P2P application on the same host. However, PeerDigger performs well in these scenarios and shows good flexibility.

This paper is organized as follows. Section 2 summarizes related work. Section 3 provides an overview of our system and presents its mechanism in detail. Then, the experiments and main results are illustrated in Section 4. Finally, Section 5 concludes the paper.

II. RELATED WORK

Traditional approaches aiming at detecting P2P botnets mainly lie in two ways. Blacklist-based approaches [12-15] build publicly available blacklists of IP addresses or domain

978-1-4799-7981-3/14 $31.00 © 2014 IEEE
DOI 10.1109/CSE.2014.283

names of known botnet servers. The drawback of blacklists is apparent, since most botmasters use techniques such as fast-flux to hide the backend control servers [16]. Payload-based approaches [17-19] are another kind of traditional way to detect botnets, which identify botnets traffic by examining content of packets payload. For example, Botzilla [17] and Snort ruleset [18] identify botnets traffic using payload signatures for a set of known botnets. Wurzinger et al. [19] assume abnormal activities in network behaviors of a bot are indications of its responses to commands from the botmaster. Thus, they examine the traffic snippets before the abnormal activities, and extract common substrings in the payload that can be used as signatures. However, these approaches are easily evaded by communication obfuscation or encryption.

Consequently, recent researches have turned to detect P2P botnets through analyzing the behavioral characteristics of the network traffic. First of all, approaches such as BotHunter [20] and BotMiner [21] identify bots based on detectable malicious activities, including scanning, spamming, exploiting, DDoS, etc. Unfortunately, the malicious activities of P2P botnets are becoming stealthier and cannot be easily detected any more, thereby limiting the applicability of these approaches. Secondly, TDG-based (traffic dispersion graphs) approaches like BotGrep [22] and BotTrack [23] detect P2P botnets by analyzing the communication graphs extracted from network flows collected over multiple large networks (e.g., ISP networks). Although they do not rely on the malicious activities for bots identification, they need information about infected hosts collected from additional systems such as honeypots to bootstrap the detection and a global view of Internet traffic, which is hard to acquire. Another trend of research assumes that P2P bots belonging to the same botnets exhibit some similarities in their network behaviors. They detect P2P bots by clustering hosts exhibiting similar network behaviors, such as performing similar suspicious activities and traffic statistics [21, 24], exhibiting long active time and sharing common external destinations [25], or exhibiting similar timing patterns in traffic [26]. However, these works can only detect P2P bots when there are multiple infected bots belonging to the same botnet in the monitored network. They often become inefficient in the case that the target network has only a single bot. Moreover, another category of approaches [27-30] dedicates to detect P2P bots with the help of statistical features extracted from network traffic and machine learning algorithms such as SVM, KNN, REP trees, and so on. For example, Zhao et al. [28] select four statistical features such as variance of payload length, number of exchanged packets, size of the first packet, and number of flows per address, and apply a Bayes Network algorithm and a Decision Tree algorithm to detect traffic of P2P bots. PeerRush [29] first utilizes three statistical features to identify all P2P host inside a monitored network, and then employs 160 statistical features extracted from the distribution of bytes-per-packet and inter-packet-delays in the management flows to classify a number of benign P2P applications and three P2P botnets. The classifiers it used are trained by KNN and Gaussian machine learning algorithms. Although these statistic-based approaches are able to detect even single bot inside a network perimeter and do not rely on the observation of malicious activities, they often become invalid in the case that a bot-compromised host runs other benign P2P applications along with P2P bot at the same time. In this case, the statistical features extracted from host traffic may become ineffective since the traffic exhibits mixed patterns of both benign and malicious P2P applications. Finally, few of previous approaches have considered the ability of real-time detection. It is important to detect P2P bots before they launch attacks. Some of the statistic-based approaches [28, 30] may be capable of real-time processing because of their short detection time windows. However, they require substantial computational and memory resources due to their complicated features and sophisticated algorithms, which may make them hard to scale to high-speed networks. Yu et al. [24] propose an online botnet detection method based on Discrete Fourier Transform. However, it needs not only observation of malicious activities to confirm bot-infected hosts, but also multiple congeneric bots inside its target network.

PeerDigger addresses the aforementioned problems from the following perspectives:

- Based on traffic analysis, PeerDigger is able to detect P2P bots within a monitored network in real-time, while not relying on payload of individual packets or the observation of malicious activities.
- It is able to detect even single bot inside the monitored network and dual-hosts that run both P2P bot and other benign P2P application at the same time.

III. System Overview

Given network traffic observed at the border of a monitored network, PeerDigger aims at identifying internal P2P bots. The main challenge in doing so is to distinguish them from benign P2P hosts. Thus, PeerDigger implements a two-phase system that consists of a *P2P host detection phase* and a *P2P bot identification phase*, as shown in Fig. 1. In the first phase, PeerDigger devotes to detect all P2P hosts (either benign or malicious) inside the monitored network that engage in P2P communications. To this end, we first translate the real-time packets stream into flows stream, and then discard network flows that are unlikely to be generated by P2P applications by a Flow Filter module. After that, we partition the flows stream of every host H into time windows of constant size T (e.g. T=3 minutes), and detects whether H may be running a P2P application solely based on several basic properties of flows records. In the second phase, PeerDigger distinguishes P2P bots from benign P2P clients by analyzing their network behaviors. Specifically, we investigate the distribution of re-connection number to the same destination IP in a group of aggregation flows generated by P2P host H within the considered time window to finalize bot identification.

A. P2P Host Detection

Since P2P botnets implement their command-and-control channels in a P2P fashion, P2P bots exhibit some network behaviors that are common to benign P2P applications. Thus, we first detect all hosts engaged in P2P communications through traffic analysis before identifying P2P bots.

1) Flow Extraction. A flow represents a set of IP packets exchanged between two entities. It is uniquely identified by the 5-tuple: protocol, source and destination IP address, and associated ports. Since different P2P applications prefer to different transport-layer protocols to communicate and

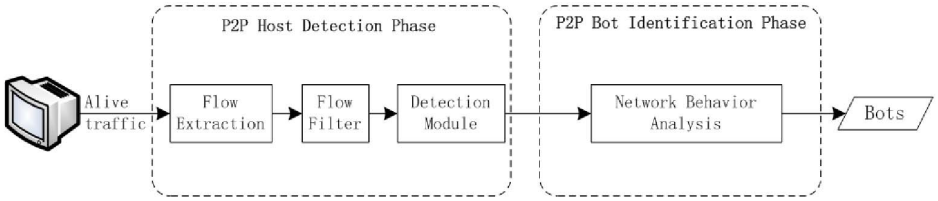

Figure 1. PeerDigger system overview.

transfer data, we consider both TCP and UDP flows. Specifically, we restrict our attention to successful TCP flows which have completed SYN, SYN/ACK, ACK handshakes and all UDP flows since UDP is a connectionless transport protocol. Some contemporary works such as [25] discard all incoming flows and UDP flows without consequent response packets (i.e., failed UDP flows). However, we note that those flows also play a non-ignorable role in P2P network activities, which will be discuss in detail in the following paragraphs. In this paper, a flow is considered expired in the following conditions:

- The flow is inactive (no new packets arrived) for 10 minutes.
- The flow is long lived and lasts longer than 30 minutes.
- A TCP flag that indicates the flow is terminated has been observed, e.g., FIN, RST flag.

To translate packets stream into flows stream, we characterize each flow using a vector of its several basic properties, namely $v(f) = \langle T_{end}, P, S_{pkts}, S_{bts}, R_{pkts}, R_{bts} \rangle$. T_{end} is the end timestamp of the flow (the timestamp of the last packet in the flow). P represents the transport-layer protocol of the flow. S_{pkts} and S_{bts} respectively represent the total amount and size of all packets sent, while R_{pkts} and R_{bts} respectively represent the total amount and size of all packets received. In this way, the real-time packets stream generated by every host H can be translated into flows stream $V(H) = \{v(f)\}$.

2) Flow Filter. The goal of the Flow Filter module is to filter out network flows that are unlikely generated by P2P network activities. Peers of P2P networks join and leave the network constantly. They often contact their counterparts directly by looking up IP addresses from a routing table, with no prior DNS requests. Although it is still possible to resolve domain names at bootstrap, P2P peers further communicate directly using IP addresses in the overlay network [31]. By contrast, most non-P2P applications usually need to resolve domain names before initiating flows. Therefore, we discard flows preceded with a successful DNS resolution. Specifically, the Flow Filter module keeps a list of IP addresses which resolved from domain names, and discards all flows send to or received from these IP addresses. This coarse filter can filter out a large part of non-P2P network flows, while retaining the majority of P2P network flows.

3) Detection Module. Due to the inherent nature of P2P networks, every peer within a P2P network has to frequently exchange plenty of packets with other peers to keep the effectiveness and robustness of the network [32]. These packets are generated by various P2P network activities, such as peer discovery, content request, notification, data transmission, and so on. We have noticed that the flows generated by certain network activity of certain P2P network usually share same characteristics, such as transport-layer protocol, packets amount and size. These characteristics are diverse in different network activities or different P2P protocols. In other words, if two network flows are generated by the same P2P application and correspond to the same network activity, they tend to have same transport-layer protocol, packets amount and size. Therefore, we define similar flows generated by prominent network activities as aggregation flows. There are lots of clusters of aggregation flows in the traffic generated by a P2P host, since it has a number of different network activities. Moreover, the destination IP addresses of the aggregation flows within a cluster may spread across a large amount of networks, since a peer in P2P network will exchanges packets with a lot of other peers distributed in many different networks [33]. Thus, we further define aggregation flows to be prominent flow clusters with not only same characteristics, but also a large number of distinct BGP prefixes of destination IP addresses.

PeerDigger detects all P2P hosts within the monitored network that engage in P2P communications through identifying the aforementioned aggregation flows corresponding to P2P network activities. In order to detect the aggregation flows in real-time, we partition the flows stream of every host H into time windows of constant size T (e.g. T=3 minutes) according to the end timestamp T_{end} of flow vectors. In this way, flows stream generated by host H is split into sets of flow vectors, which denoted as $V(H)_{slice} = \{v(f)'_i\}$, where $v(f)'_i = \langle P, S_{pkts}, S_{bts}, R_{pkts}, R_{bts} \rangle$ (T_{end} is excluded since it is no longer needed). At the end of each time window, flow vectors with same values are grouped into a numbers of clusters, $C_1(H), \ldots, C_m(H)$. For each of the $C_j(H)$, we consider of the set of destination IP addresses of flows in the cluster, and count its number of distinct BGP prefixes $bgp_j = BGP(C_j(H))$. After that, we discard those clusters whose bgp_j is smaller than a threshold θ_{bgp} and consider the flows in remaining clusters as aggregation flows. For convenience, we use vector $v(f)'$ in the cluster to represent this group of aggregation flows, and denote it as AF. Thus, for each host H, a set of AF can be extracted from the slice of flows stream at the end of time window, i.e., $AF(H) = \{AF_1, \ldots, AF_n\}$. We consider host H as P2P host if it generated at least one group of aggregation flows, namely $AF(H) \neq \emptyset$.

B. P2P Bot Identification

In this phase, PeerDigger aims at distinguishing P2P bots among the P2P hosts detected in the first phase. Since both P2P bots and benign P2P applications are implemented in P2P fashion, some common network behaviors patterns are shared by them. However, there definitely exist some tiny differences between the two types of P2P applications due to the different circumstances and goals behind how they utilize the P2P protocol.

We noticed that P2P bots are more likely to contact the same external hosts than benign P2P applications. Specifically, flows in AF (i.e., aggregation flows corresponding to the same P2P network activity) generated by P2P bots prefer to establish connections to the destination IP addresses that have been contacted by other flows in the same AF, than those generated by benign P2P applications. In other words, aggregation flows generated by P2P bots tend to re-connect the same hosts to communicate. The reasons for this phenomenon may lie in the following two aspects. On the one hand, P2P bots are likely to experience less peer churn (i.e., the dynamics of peers joining and leaving the P2P network) than benign P2P clients [26]. In order to receive and execute commands from the botmaster, P2P bots have motivation to keep up persistent communications with each other and maintain the connectivity of the botnet. By contrast, the peer membership of benign P2P applications, such as file-sharing systems and IPTV platforms, is very dynamic due to the availability of the desired files and their short duration lives [31]. They tend to contact more new peers whoever own requested content to maintain a steady download speed. On the other hand, most P2P bots store a list of known peers with which to communicate, both for bootstrapping itself into the botnet [1-3], and to limit the number of peers it contacts to evade detection. All these characteristics make it more likely for P2P bots to contact the same hosts than benign P2P applications.

In order to leverage this characteristic to identify P2P bots in real-time, we inspect the re-connection distribution of the aggregation flows generated by detected P2P hosts at the end of time window. For every $AF(H) = \{AF_1, \dots, AF_n\}$ extracted from each P2P host H, we first translate it into $\Gamma(H) = \{\Gamma_1, \dots, \Gamma_n\}$, where Γ_j is the set of destination IP addresses of the flows in AF_j, i.e., $\Gamma_j = \{IP_1, \dots, IP_q\}$. We define the re-connection number (RCN) of AF_j as the number of repeated elements in Γ_j, and denoted as,

$$\mathrm{RCN}(\Gamma_j) = \sum_{x=1}^{q}(count(\mathrm{IP}_x) - 1) \quad (1)$$

After that, we define Re-Connection Ratio (RCR) of host H as the maximum re-connection number of each AF.

$$\mathrm{RCR(H)} = \max\{\mathrm{RCN}(\Gamma_1), \dots, \mathrm{RCN}(\Gamma_n)\} \quad (2)$$

Since P2P bots prefer to communicate with the same hosts, they will produce a large value of $\mathrm{RCR}(H)$. After computing the RCR for every detected P2P host, we label host H as P2P bot if its $\mathrm{RCR}(H)$ is greater than a threshold θ_{RCR}.

This simple yet effective mechanism greatly saves the overhead of detection, and makes it possible for PeerDigger to process real-time detection even in very busy networks.

TABLE I. ALGORITHM OF PEERDIGGER

Algorithm 1: PeerDigger(H)

Input: alive traffic of host H.

01 $V(H) \leftarrow \{v(f)_i = \langle T_{end}, P, S_{pkts}, S_{bts}, R_{pkts}, R_{bts}\rangle\}$ /* Flow extraction */

02 $Filter\_List(IP) \leftarrow IP_{resolved}$ /* Add IP address resolved from DNS to filter list */

03 $\mathrm{Filter}(V(H))$ /* Filter flow vectors */

04 for every time window /* Split flow stream */

05 $V(H)_{slice} \leftarrow \{v(f)'_i = \langle P, S_{pkts}, S_{bts}, R_{pkts}, R_{bts}\rangle\}$ /* Exclude T_{end} out of flow vector */

06 $\{C_1(H), \dots, C_j(H), \dots, C_m(H)\} \leftarrow V(H)_{slice}$ /* Group flows with same vector */

07 for every $C_j(H)$ /* Discard clusters with small bgp */

08 if $(\mathrm{BGP}(C_j(H)) > \theta_{bgp})$ $AF_j \leftarrow C_j(H)$

09 $AF(H) = \{AF_1, \dots, AF_n\}$

10 if $(AF(H) = \emptyset)$ /* P2P host detection */

11 Output("This is not a P2P host.")

12 if $(AF(H) \neq \emptyset)$

13 $\{\Gamma_1, \dots, \Gamma_n\} \leftarrow \{AF_1, \dots, AF_n\}$ /* Extract destination IP address for every AF */

14 for every Γ_j /* Compute re-connection number of each AF */

15 $\mathrm{RCN}(\Gamma_j) = \sum_{x=1}^{q}(count(\mathrm{IP}_x) - 1)$

16 $\mathrm{RCR}(H) = \max\{\mathrm{RCN}(\Gamma_1), \dots, \mathrm{RCN}(\Gamma_n)\}$/* Compute $\mathrm{RCR}(H)$ */

17 if $(\mathrm{RCR}(H) < \theta_{RCR})$ /* P2P bot identification */

18 Output("This is a benign P2P host.")

19 else

20 Output("This is a P2P bot.")

21 end

Moreover, unlike most previous approaches which assume the coexistence of multiply bots inside a monitored network as a prerequisite, our identification mechanism relies solely on the network behavior of single bot. Furthermore, even if host H is infected with a P2P bot and run another benign P2P application simultaneously, the value of $\mathrm{RCR}(H)$ will not be impacted. It is because that respective AF will be extracted from the traffic generated by both P2P bot and benign P2P application without interference, and at least one AF will be generated by the P2P bot which will yield a great value of $\mathrm{RCN}(\Gamma)$ due to its network behavior characteristic.

To sum up, Table 1 describes the algorithm of PeerDigger. The input of the algorithm is the alive traffic (packets stream) generated by host H. At the end of every time window, it outputs a conclusion whether host H is a P2P host or not, and if yes, whether it is a P2P bot or a benign P2P host. From line 01 to line 11 is the P2P host detection phase. More specifically, line 01 translates the real-time packets stream into flows stream, line 02-03 describe the flow filter module, and line 04-11 implement detection module. The P2P bot identification phase is depicted in the rest part of the algorithm (line 12-21).

TABLE II. SUMMARIES OF TRAFFIC DATASETS

Category	Application	Duration	Hosts	Packets	Flows
Non-P2P dataset	Web, email, online games...	24 hours	35	465.24M	618.25K
Benign P2P dataset	BitTorrent	24 hours	2	416.52M	495.57K
	eMule	24 hours	2	403.83M	99.57K
	CNTV	24 hours	2	5.48M	90.12K
P2P botnet dataset	Storm	24 hours	13	54.91M	3578.55K
	Waledac	24 hours	3	12.92M	1296.86K

IV. EXPERIMENTAL RESULTS AND ANALYSIS

A. Dataset Collection

To evaluate the performance of PeerDigger, we constructed three datasets: a dataset of non-P2P traffic, a dataset of P2P traffic generated by a variety of popular P2P applications and a dataset of traffic from two famous P2P botnets.

Dataset of non-P2P traffic. To collect the dataset of non-P2P traffic, we first monitored the traffic of a subnet in our campus network and captured all packets crossing the gateway router for a whole day. Then, in order to discard possible P2P traffic, we used Snort [18] with a large set of publicly available P2P detection rules based on deep packet inspection (DPI). Overall, we observed 35 active hosts which did not run any P2P application in the subnet. This dataset contains a large number of general traffic from a variety of applications, including web-browsing, email, online games, instant message, etc. This variety of traffic serves as a good example of daily use of campus networks.

Dataset of P2P traffic. We collected the P2P traffic dataset in a fully controlled environment. In order to increase the diversity of P2P hosts in our evaluation, we chose three of the most popular P2P applications, namely, BitTorrent, eMule and CNTV, which contain both P2P file-sharing systems and P2P-TV platform. We built an experimental subnet in our campus consisting of 6 hosts. We separately used two hosts to run one of the aforementioned P2P applications for an entire day. By means of AutoIt scripts [34], these hosts automatically simulated various user behaviors. For example, at random time intervals, BitTorrent and eMule randomly chose files to download or upload, CNTV randomly selected channels or videos to play. All traffic generated by them was captured and collected into the dataset. Using traffic dataset from completely controlled network has the advantage of providing a reliable ground truth, since there is no doubt on the application generating the traffic.

Dataset of P2P botnet traffic. We also obtained datasets of P2P botnet traffic from third parties. Specifically, there include a 24-hour trace of Storm which contains traffic from 13 bots, and a 24-hour trace of Waledac which contains 3 bots. Table 2 summarizes these traffic datasets and reports their brief information.

In order to produce an experimental dataset which contains all the three types of traffic mentioned above, we merged the three individual datasets into a single one via a process as follows. Firstly, we separately overlaid the traffic of 6 benign P2P hosts to the traffic of 6 randomly selected hosts from non-P2P traffic dataset. Then, we randomly selected 3 P2P bots (2 Storm bots and a Waledac bot), and separately overlaid their traffic to the traffic of a benign P2P host of each type (after merging with non-P2P host). This is to simulate the scenario

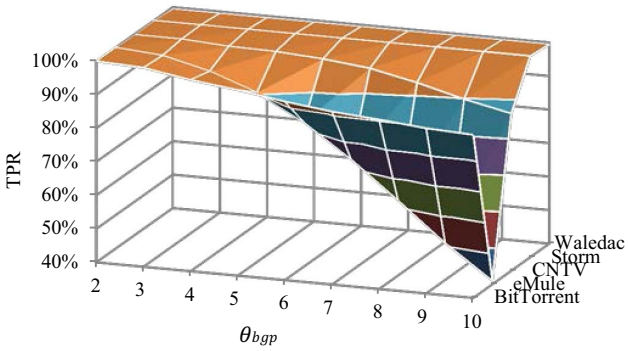

Figure 2. TPR of P2P host detection for different value of θ_{bgp}.

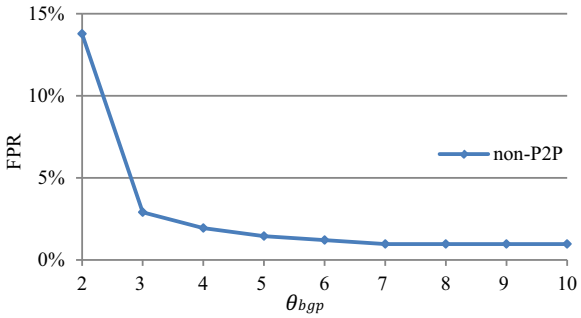

Figure 3. FPR of P2P host detection for different value of θ_{bgp}.

that a benign P2P host is compromised by a P2P bot simultaneously. Finally, the traffic of remaining 11 Storm bots and 2 Waledac bots was overlaid to the traffic of other 13 randomly selected hosts from non-P2P traffic dataset respectively. In this way, our experimental dataset is composed of 35 hosts including 1) 16 non-P2P hosts, 2) 3 benign P2P hosts of each type, 3) 11 Storm bots and 2 Waledac bots, 4) 3 dual-hosts that run a P2P bot along with a benign P2P application at the same time.

We empirically set the length of time window to 3 minutes and applied PeerDigger on the aforementioned experimental dataset to evaluate its performance. The evaluation of different time window length is discussed in Section Ⅳ-D.

B. Evaluation of P2P Host Detection

The value of threshold θ_{bgp} is crucial for the detection of P2P host. Some irrelevant flows may be considered as AF, if θ_{bgp} is too small. Inversely, if θ_{bgp} is set too big, some groups of aggregation flows may be discarded. To achieve a high true positive rate (TPR) while keep the false positive rate (FPR) in a low level, we should set the value of θ_{bgp} carefully. In order to find out the optimal value of threshold θ_{bgp}, we separately set θ_{bgp} from 2 to 10, and test the P2P host detection performance of PeerDigger. The results in terms of TPR and FPR are separately presented in Fig. 2 and Fig. 3. In this phase, all P2P hosts within the experimental dataset (either benign or malicious) belong to the positive class, while all non-P2P hosts compose the negative class. As can be seen, smaller value of

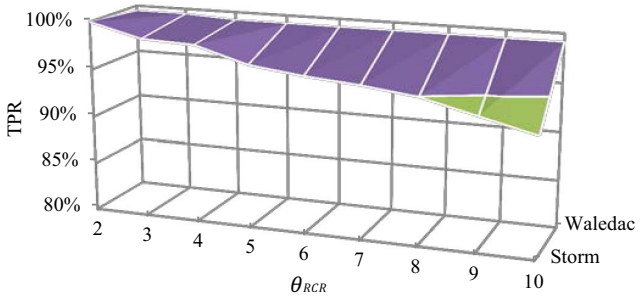

Figure 4. TPR of P2P bot identification for different value of θ_{RCR}.

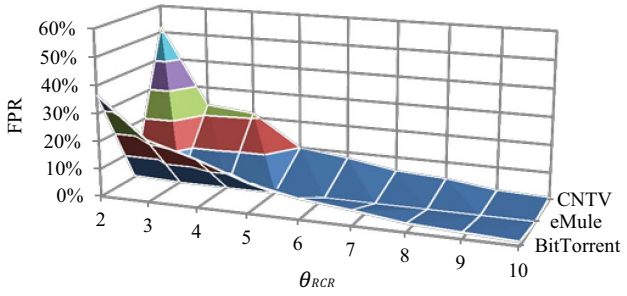

Figure 5. FPR of P2P bot identification for different value of θ_{RCR}.

θ_{bgp} results in higher TPR, but also leads to higher FPR. Conversely, higher θ_{bgp} results in both lower TPR and FPR. For $T = 3$ minutes, the best results are obtained when $\theta_{bgp} = 4$, with an average TPR of 98.2% and a FPR of 1.93%.

C. Evaluation of P2P Bot Identification

As θ_{bgp} in the first phase, θ_{RCR} is a determining factor in P2P bot identification. To identify P2P bots from detected P2P hosts, we set the value of threshold θ_{RCR} carefully. We repeated the same experiment for different values of θ_{RCR} ranging from 2 to 10. The P2P bot identification TPR and FPR of different value of θ_{RCR} are presented in Fig. 4 and Fig. 5. In this phase, the positive class consists of 11 Storm bots, 2 Waledac bots, and 3 dual-hosts that run both P2P bot and other benign P2P application at the same time, while the negative class consists of 3 hosts that only run a benign P2P application alone. As can be seen from the curves, PeerDigger achieves an extremely high TPR when θ_{RCR} is small. However, the FPR of BitTorrent and CNTV is also non-ignorable at the same area. It is probably because that the threshold value of θ_{RCR} is not large enough to separate P2P bots from benign P2P hosts. When $\theta_{RCR} = 8$, we obtain an average TPR of 97.05% and an acceptable average FPR of 2.32%. It is worth noting that all the 3 dual-hosts, which were compromised by a P2P bot and run a benign P2P application simultaneously, are correctly identified as P2P bots. The high TPR strongly demonstrates that the P2P bot identification performance of PeerDigger is completely unaffected by the benign P2P application running on the infected host.

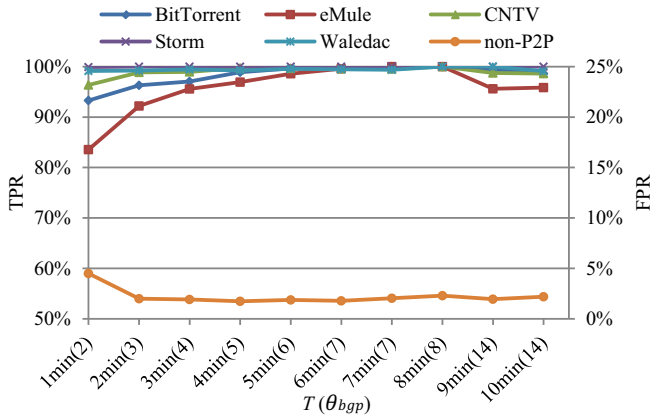

Figure 6. TPR and FPR of P2P host detection for different length of time window T.

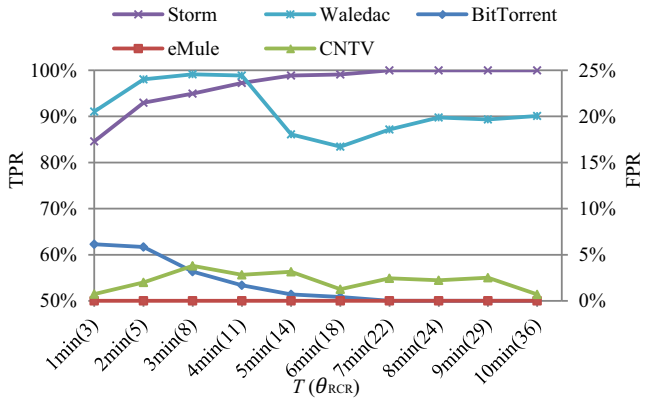

Figure 7. TPR and FPR of P2P bot identification for different length of time window T.

D. Adjusting of Time Window T

The length of time window T is a tradeoff between efficiency and accuracy. From the perspective of efficiency, we would like to set T as small as possible to satisfy real-time processing. However, too small time window usually leads to low TPR and high FPR, since not enough evidence is collected during the time window. In order to find out the optimal length of time window T, we separately set the value of T from 1 minute to 10 minutes, and repeatedly test the performance of PeerDigger using the experimental dataset. For every value of time window T, we carefully selected the threshold value of θ_{bgp} and θ_{RCR} to make sure that PeerDigger achieved a relative high TPR and an acceptable FPR, as we did in Section Ⅳ-B and Ⅳ-C.

The results of P2P host detection and P2P bot identification of different time window length are separately presented in Fig. 6 and Fig. 7. The optimal value of $\theta_{bgp}/\theta_{RCR}$ for each time window length is shown in brackets behind corresponding T. The curves in the top of both figures present TPR of corresponding applications and the curves at the bottom present

FPR. In P2P host detection phase (Fig. 6), with the value of T increases, the TPR curves keep on ascending and the FPR curve always maintains in a low level. In Fig. 7, the average TPR of P2P bot identification reaches peak value when $T = 4$ minutes, and the average FPR is negligible at this point. After comprehensive analyzing, we consider 4 minutes as a suitable length of time window for our system, and accordingly, $\theta_{bgp} = 5$, and $\theta_{RCR} = 11$. In this configuration of parameter settings, PeerDigger is able to obtain an average TPR of 98.96% with a 1.75% FPR in P2P host detection phase, and an average TPR of 98.07% and an average FPR of 1.5% in P2P bot identification phase.

V. CONCLUSIONS

In this paper, we presented PeerDigger, a novel real-time system capable of detecting stealthy P2P bots within a monitored network through traffic analysis. PeerDigger first detects all hosts that engage in P2P communications based on aggregation flows, and then identifies P2P bots among the detected P2P hosts by analyzing their network behavior patterns. The novelty of our system lies in following aspects: First, it is able to detect stealthy P2P bots even through their malicious activities are non-observable, with no access to packets payload content. Second, PeerDigger is capable of real-time detection, because of its simple yet effective mechanism and small detection time windows. Finally, PeerDigger has excellent flexibility compare with other contemporary works. The evaluation results demonstrated that PeerDigger is able to detect P2P hosts with an average TPR of 98.96% and an average FPR of 1.75%, and further identity P2P bots with an average TPR of 98.07% and an average FPR of 1.5% within 4 minutes.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China under Grant No. 61170286 and NO. 61202486.

REFERENCES

[1] J. Grizzard, V. Sharma, C. Nunnery, and B. Kang, "Peer-to-Peer Botnets: Overview and Case Study," Proc. the first conference on first workshop on Hot Topics in Understanding Botnets, 2007, pp. 1-1.

[2] S. Stover, D. Ditrich, J. Hernandez, and S. Dietrich, "Analysis of the Storm and Nugache Trojans: P2P is Here," USENIX, vol. 32, no. 6, 2007, pp. 18-27.

[3] 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," Proc. LEET 08, vol.8, no. 1, 2008, pp. 1-9.

[4] Trusteer, "No Silver Bullet: 8 Ways Malware Defeats Strong Security Controls," 2012. (IBM white paper)

[5] Y. Zhao, Y. Xie, F. Yu, Q. Ke, and Y. Yu, "Botgraph: Large Scale Spamming Botnet Detection," Proc. USENIX NSDI, vol. 9, 2009, pp. 321-334.

[6] P. O'Kane, S. Sezer, and K. McLaughlin, "Obfuscation: The Hidden Malware," IEEE Security & Privacy, vol. 9, no. 5, 2011, pp. 41-47, doi: 10.1109/MSP.2011.98.

[7] L. Borup, "Peer-to-Peer Botnets: A Case Study on Waledac," Master's thesis, Technical University of Denmark, 2009.

[8] G. Sinclair, C. Nunnery, and B. B. Kang, "The Waledac Protocol: The How and Why," Proc. International Conference on Malicious and Unwanted Software (MALWARE 09), IEEE Press, 2009, pp. 69-77, doi: 10.1109/MALWARE.2009.5403015.

[9] B. Stock, J. Goebel, M. Engelberth, F. C. Freiling, and T. Holz, "Walowdac – Analysis of a Peer-to-Peer Botnet," Proc. European Conference on Computer Network Defense (EC2ND), IEEE Press, 2009, pp. 13-20, doi: 10.1109/EC2ND.2009.10.

[10] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. M. Youssef, M. Debbabi, and L. Wang, "On the Analysis of the Zeus Botnet Crimeware Toolkit," Proc. Eighth Annual International Conference on Privacy Security and Trust (PST 10), IEEE Press, 2010, pp. 31-38, doi: 10.1109/PST.2010.5593240.

[11] S. Shin, Guofei Gu, N. Reddy, and C. P. Lee, "A Large-Scale Empirical Study of Conficker," IEEE transctions on Information Forensics and Security, vol. 7, no. 2, 2012, pp. 676-690, doi: 10.1109/TIFS.2011.2173486.

[12] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "Exposure: Finding Malicious Domains using Passive DNS Analysis," Proc. the 18th Network and Distributed System Security Symposium (NDSS 11), 2011.

[13] "AMaDa Blocklist," http://amada.abuse.ch/blocklist. php.

[14] "Safe browsing API," http://code.google.com/apis/safebrowsing/.

[15] "Phish Archive," http://www.phishtank.com/.

[16] Honeynet Project, "Know Your Enemy: Fast-Flux Servicenetworks," The Honeynet Project and Research Alliance, Tech. Rep., 2008.

[17] K. Rieck, G. Schwenk, T. Limmer, T. Holz, and P. Laskov, "Botzilla: Detecting the Phoning Home of Malicious Software," Proc. Symposium on Applied Computing (SAC 10), ACM Press, 2010, pp. 1978-1984, doi: 10.1145/1774088.1774506.

[18] "Snort Ruleset," http://www.emergingthreats.net/.

[19] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, and E. Kirda, "Automatically Generating Models for Botnet Detection," Proc. European Symp. on Research in Computer Security (ESORICS 09), Springer Berlin Heidelberg, vol. 5789, 2009, pp. 232-249, doi: 10.1007/978-3-642-04444-1_15.

[20] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "BotHunter: Detecting Malware Infection through IDSdriven Dialog Correlation," Proc. USENIX Security, vol. 7, 2007, pp. 1-16.

[21] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection," Proc. USENIX Security, 2008, pp. 139-154.

[22] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov, "Botgrep: Finding P2P Bots with Structured Graph Analysis," Proc. USENIX Security, 2010, pp. 95-110.

[23] J. Francois, S. Wang, R. State, and T. Engel, "BotTrack: Tracking botnets using netFlow and pageRank," in *Proc. the 10th International IFIP TC 6 Conference on Networking – Volume Part I*, pp. 1-14, 2011.

[24] X. Yu, X. Dong, G. Yu, Y. Qin, D. Yue, and Y. Zhao, "Online Botnet Detection Based on Incremental Discrete Fourier Transform," Journal of Networks, vol. 5, no.5, 2010, pp. 568-576.

[25] J. Zhang, R. Perdisci, W. Lee, and X. Luo, "Building a Scalable System for Stealthy P2P-Botnet Detection," IEEE Transactions on Information Forensics and Security, vol. 9, no. 1, 2014, pp. 27-38, doi: 10.1109/TIFS.2013.2290197.

[26] T.-F. Yen and M. K. Reiter, "Are Your Hosts Trading or Plotting? Telling P2P File-Sharing and Bots Apart," Proc. International Conference on Distributed Computing Systems (ICDCS 10), IEEE Press, 2010, pp. 241-252, doi: 10.1109/ICDCS.2010.76.

[27] N. Kheir and C. Wolley, "BotSuer: Suing Stealthy P2P Bots in Network Traffic through Netflow Analysis," Cryptology and Network Security, 2013, pp. 162-178.

[28] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, "Botnet Detection Based on Traffic Behavior Analysis and Flow Intervals," Computers & Security, vol. 39, 2013, pp. 2-16.

[29] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li, "PeerRush: Mining for Unwanted P2P Traffic," Detection of Intrusions and Malware, and Vulnerability Assessment, 2013, pp. 62-82.

[30] P. Narang, V. Khurana, and C. Hota, "Machine-Learning Approaches for P2P Botnet Detection using Signal-Processing Techniques," Proc.

the 8th ACM International Conference on Distributed Event-Based Systems (DEBS 14) , 2014, pp. 338-341.

[31] K. Aberer and M. Hauswirth, “An Overview on Peer-To-Peer Information Systems,” Proc. the 4th Workshop on Distributed Data and Structures, 2002.

[32] Jie He, Yuexiang Yang, Yong Qiao, and Chuan Tang, “Accurate Classification of P2P Traffic by Clustering Flows,” China Communications, vol. 10, no. 11, 2013, pp.42-51.

[33] Daniel Stutzbach and Reza Rejaie, “Understanding Churn in Peer-To-Peer Networks,” Proc. the 6th ACM SIGCOMM Conference on Internet Measurement, 2006, pp. 189-202.

[34] “Autoit script,” http://www.autoitscript.com/.