PeerClean: Unveiling Peer-to-Peer Botnets through Dynamic Group Behavior Analysis

Qiben Yan Yao Zheng Tingting Jiang Wenjing Lou Y. Thomas Hou
Virginia Polytechnic Institute and State University, Blacksburg, VA, USA

***Abstract*—Advanced botnets adopt a peer-to-peer (P2P) infrastructure for more resilient command and control (C&C). Traditional detection techniques become less effective in identifying bots that communicate via a P2P structure. In this paper, we present PeerClean, a novel system that detects P2P botnets in real time using only high-level features extracted from C&C network flow traffic. PeerClean reliably distinguishes P2P bot-infected hosts from legitimate P2P hosts by jointly considering flow-level traffic statistics and network connection patterns. Instead of working on individual connections or hosts, PeerClean clusters hosts with similar flow traffic statistics into groups. It then extracts the collective and dynamic connection patterns of each group by leveraging a novel dynamic group behavior analysis. Comparing with the individual host-level connection patterns, the collective group patterns are more robust and differentiable. Multi-class classification models are then used to identify different types of bots based on the established patterns. To increase the detection probability, we further propose to train the model with average group behavior, but to explore the extreme group behavior for the detection. We evaluate PeerClean on real-world flow records from a campus network. Our evaluation shows that PeerClean is able to achieve high detection rates with few false positives.**

I. Introduction

Botnet has become a major threat to the health of modern networks. Through large-scale compromise of end hosts, botmasters can commit organized cyber-crimes, such as launching distributed denial-of-service (DDoS) attacks, sending spams, performing click frauds, or stealing sensitive information.

The C&C channel is one of the most essential components of a botnet, through which a botmaster manages a bot army of compromised end hosts. One common type of C&C infrastructure relies on a central C&C server, which has recently drawn a great deal of attention from security researchers and law enforcement forces. From the attacker's point of view, such centralized architecture suffers from a *single point of failure* problem, because if the C&C server is identified and taken down, the botmaster will lose control over the whole botnet. As a response, sophisticated botnet developers attempt to build more advanced and resilient P2P C&C infrastructures. P2P C&C allows the bots to exchange C&C messages via their connected peers in a P2P manner. Therefore, despite of numerous takedown attempts, P2P botnets kept reviving. Some notable examples of active P2P botnets include Sality, ZeroAccess, and Kelihos, which have survived in the wild for a long time and will likely continue to be alive in the near future.

To date, a few solutions have been proposed to detect P2P botnets [1]–[4]. Host-level malware detection techniques such as traditional signature-based approaches and more recently behavior-based approaches [5] have been designed. However, these approaches are not only vulnerable to advanced malware obfuscation or polymorphism, but they also require host-side installation. So they are unattractive to the network administrators who aim to crack down a network-wide botnet. Alternatively, network-level techniques have been proposed to correlate the traffic patterns of suspicious bots [2], [3], [6]–[9] or collect network communication graphs to identify P2P bots [1], [10]. Some of them apply *deep packet inspection* (DPI), which is not only computationally expensive, but is also evadable through encryption. Other approaches are based upon network flow traffic analysis. For instance, Yen et al. [2] described an algorithm to differentiate P2P file sharing applications with P2P bots based on network traffic features such as traffic volume, peer churn rate, and interstitial time distribution. Recently, Zhang et al. [3] developed a botnet detection system to extract statistical fingerprints for every host, and identify the bots based on a set of traffic features such as communication persistency, fingerprint similarity, and shared contacts' number. However, the traffic features used in these approaches are not robust enough to identify bots in a dynamic network, as observed from our experiments. On the other hand, a communication graph-based approach [1] seems more reliable, but it can only identify structural P2P subgraphs regardless of whether the subgraphs contain bots. Also, it requires a list of *honeypot* hosts to bootstrap its detection algorithm, limiting its practicality.

In this paper, we jointly consider two sets of features: flow traffic statistics and network connection behaviors, to detect the presence of P2P bots within a monitored network, such as a campus network or an ISP network. We introduce *PeerClean* to utilize the best of these feature sets via a novel combination of clustering and classification. PeerClean identifies P2P bots by detecting their C&C communication patterns that characterize the botnets, regardless of how they perform malicious activities. In order to accomplish this goal, we extract the flow traffic statistics from the network flow data, based on which we group the hosts with similar traffic patterns into the same cluster. PeerClean leverages the fact that bots from the same botnet use the same C&C communication protocol which produces similar traffic patterns [3], [7], and the same type of bots thereby are highly likely to be included in the same cluster.

However, flow traffic statistical features are not robust and reliable enough to distinguish specific bots from benign hosts due to the dynamics exhibited in the Internet traffic. Therefore, PeerClean further incorporates more robust network connection patterns for a more accurate bot identification. Rather than exploring the connection persistency of every individual bot [2] or the number of overlapping peers of every bot pair [3], we propose a *dynamic group behavior analysis* (DGBA)

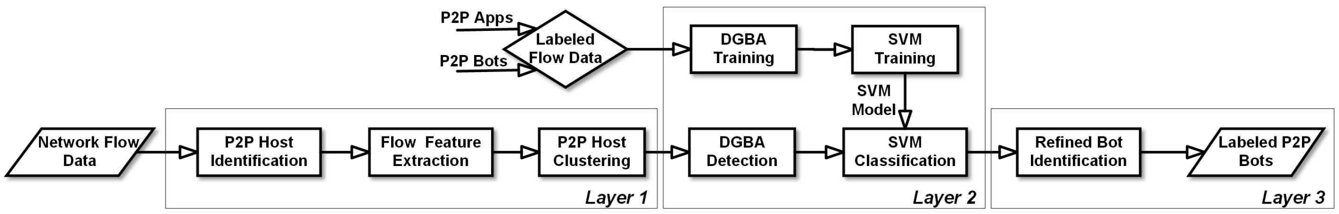


Fig. 1: PeerClean system flow

method to investigate the group-level connection behaviors inside botnets. We apply DGBA to every host cluster so as to extract the aggregated connection features. PeerClean then trains a *support vector machine* (SVM) classifier using the group-level training features, and labels each cluster using the SVM classifier subsequently. To improve the detection performance, we train the classifier using average group behavior, but explore the extreme group behavior for the detection. After detecting botnets, PeerClean is able to specify the botnet types. Furthermore, PeerClean is tailored to support real-time bot detection, and to enable a quick response to the bot infections.

Specifically, this paper makes the following contributions:

- We propose a novel botnet detection framework, PeerClean, using the high-level features extracted from network flow data based on the flow-level traffic statistics and dynamic network connection patterns. Our method explores the best of these different features with a novel combination of unsupervised (clustering) and supervised (classification) machine learning methods.
- We design a dynamic group behavior analysis method to automatically extract the collective connection features from P2P host clusters. We show through experiments that the extracted group features are robust, reliable, and effective in identifying various types of P2P botnets.
- We develop a prototype system, and evaluate the system using network traces from various real-world botnet families, as well as background traces from a large campus network. We demonstrate through experiments that PeerClean can identify different types of bots with up to 95.8% accuracy, and negligible false positive rate.

II. Overview of PeerClean

Our primary goal is to design a detection system for the network administrators to identify P2P bots in a monitored network. Toward that goal, we present our data-driven detection framework, PeerClean, which exploits network flow data captured at the edge of the network.

Figure 1 shows the system flow of PeerClean. The upper part of the figure describes the training process, with inputs from two labeled data sets: one is a subset of monitored traffic data that is from the labeled legitimate P2P hosts, and the other one contains the data from the labeled P2P bots (we discuss the acquisition of training data in §IV-A). For each type of legitimate P2P hosts and P2P bots, PeerClean then performs DGBA training to extract a collection of group-level connection features aggregated from all the hosts of this specific type, and trains a SVM classification model using the extracted group-level features. The bottom part of the figure presents the detection process with input of monitored traffic data. After identifying the P2P hosts in the network, PeerClean carries out P2P host clustering using the statistical features of their traffic flows, and applies DGBA detection to every cluster of interest with the goal of detecting clusters containing bots. Finally, the refined bot identification picks out the bots from the clusters for further processing. PeerClean can be regarded as a three-layer system, with the first-layer modules processing data on a per-host basis, the second-layer modules processing data on a per-cluster/host group basis, while the third-layer modules further handling the identified bot clusters.

Input Data: The input data set consists of a training data set and a testing data set of NetFlow flow record format. Each flow record holds a number of attributes, such as: starting time, flow duration, source and destination IP address, source and destination ports, the number of bytes and packets transferred, TCP flags, etc. The testing data is the real-time NetFlow traffic traces captured at the gateway routers of a campus (or enterprise, ISP) network, while the training data set is constructed by combining the traffic from identified P2P bots and legitimate P2P hosts.

P2P Host Identification: The high-speed networks generate a huge amount of NetFlow data, which would potentially overwhelm the processing capability of our detection system. Thus, the first step of PeerClean is to reduce the traffic volume by filtering out the hosts that are unlikely to be related to P2P communications. Our approach is based on the observation that the hosts engaging in P2P communications exhibit high failed connection rates mainly caused by the high peer churn rate [11]. Therefore, we compute the percentage of failed connections inside each time epoch (e.g. 1 hour). The hosts with failed connection rate higher than an empirical threshold are selected as *candidate* P2P hosts [2]. This selection process allows us to retain hosts engaging in P2P communications, while eliminating a vast majority of non-P2P hosts.

Detection Period: Since bot memberships are dynamically changing with some bots being cleaned up and others being newly infected, we propose to perform bot detection periodically. PeerClean supports various lengths of detection period, as long as bots generate enough network flows with representable flow and connection features during that period. In this paper, we select one hour as the detection period in response to agile bot infections. Specifically, PeerClean produces one SVM model for each hour of the day. Then, by examining each hour of testing traces collected from the edge routers, PeerClean identifies specific types of bots existing in the network within that hour. In this manner, PeerClean enables real time bot detection which supports a fast response to the bot infections (i.e. one hour response time in this paper).

III. System Design

PeerClean systematically integrates two categories of features including flow statistical features and network connection features. The effectiveness of PeerClean largely hinges upon the discriminative ability of the selected features to set apart various P2P bots and legitimate P2P hosts. In this section, we

Feature	Descriptions
Bytes-per-flow pattern	The distribution of the number of bytes per flow sent from (received by) a host
Packets-per-flow pattern	The distribution of the number of packets per flow sent from (received by) a host

TABLE I: Flow size statistical features

discuss the rationale behind the feature selection, and look into the strengths and weaknesses of the selected features. Meanwhile, two machine learning techniques performing clustering and classification are described, which are used to gather, identify, and subsequently label the P2P bots.

A. Flow Statistical Features

The performance of host clustering relies on a set of carefully selected network flow features. A common criticism of early attempts using machine learning methods over network flow data is that the selected features were often not *robust*, resulting in an overfit model to some specific features of the training set, such as a particular port or IP address used by a bot. Dedicated bots can simply adapt their used ports and IP addresses to impair the flow analysis. To overcome such overfitting issue, we select flow features that are both robust and distinctive among the botnets, including *flow size statistical features* and *host access pattern features*. Note that, at this stage, we only extract the flow features of candidate P2P hosts who survived the P2P host identification process.

Flow Size Statistical Features: Flow size statistical features capture the flow size distribution for both outgoing flows and incoming flows at a specific host. Let $F_i^{(ob)} = \{f_j^{(ob)}\}_{j=1..m}$ and $F_i^{(ib)} = \{f_j^{(ib)}\}_{j=1..n}$ denote the series of flows sent from or received by host i inside E. We consider the basic flow size related features such as: *bytes-per-flow* (bpf) feature and *packets-per-flow* (ppf) feature, as shown in Table I. Note that each feature records the distribution of flow sizes among all outgoing (incoming) flows at the corresponding host. In particular, we extract the mean $\mu_{F_i^{(ob)}}$, $\mu_{F_i^{(ib)}}$ and the standard deviation $\sigma_{F_i^{(ob)}}$, $\sigma_{F_i^{(ib)}}$ of bpf and ppf from both the outgoing and incoming flows respectively. This group of features characterizes the regularity of traffic flow size over time for each host.

Host Access Pattern Features: We introduce host access pattern features to capture the flow arrival patterns. Table II lists the adopted features, including flow interarrival pattern, flow density pattern, and diurnal pattern. Assume $T_i^{(ob)}$ ($T_i^{(ib)}$) is a time series of starting time of outgoing (incoming) flows from host i inside E, based on which we can compute a sequence of flow interarrival time $I_i^{(ob)}$ ($I_i^{(ib)}$) by taking the difference of the starting time of two consecutive flows. Flow interarrival feature represents the statistical features of flow interarrival time sequences, including the minimum, maximum, median, and standard deviation.

Different from all the aforementioned features which are extracted inside each detection period E, the last two types of features, flow density pattern and diurnal pattern, are determined anew every day. In this work, we select three hours as a time unit with one full day divided into 8 equal time

Feature	Descriptions
Flow interarrival pattern	The distribution of incoming (outgoing) flow interarrival time at a host
Flow density pattern	The fraction of time units with more than x flows at a host
Diurnal pattern	The percentage of flow numbers in the peak (dip) period of the day at a host

TABLE II: Host access pattern features

units. We denote the number of flows to/from a certain host in each time unit during a day as $N_j, j = 1, 2, \cdots, 8$. Flow density pattern records the fraction of time units with equal or more than x flows per day, i.e., $\frac{\sum_{j=1}^{8} \sigma(N_j \geq x)}{8}$, where $\sigma()$ is a step function yielding one when $N_j \geq x$ holds, and zero otherwise. In our prototype, x is empirically set as 1000. In addition, to assess whether the flow arrival demonstrates a diurnal pattern, we define two ratios, the number of flows in the peak period and the number of flows in the dip period respectively over the total number of flows of the day [1] , as the diurnal pattern features, i.e., $\frac{N_P}{\sum_{j=1}^{8} N_j}$ and $\frac{N_D}{\sum_{j=1}^{8} N_j}$. These two types of features are inserted as additional features for the last hour of the day, further elevating the detection probability.

B. P2P Host Clustering

The rationale behind host clustering comes from the following observation: bots that belong to the same botnet run the same P2P communication protocol and share the same C&C messages. *Affinity Propagation* (AP) is a recently proposed partition-based clustering method by Frey and Dueck [12]. Compared with K-means, one of the most popular clustering methods, the performance of AP does not rely on an initial selection of *exemplars*[2] or cluster centers. Rather than specifying the number of clusters, AP can automatically determine the number of clusters solely based on the data.

The similarity $s(i, k)$ of AP indicates how well data point x_k is suited to be the exemplar of data point x_i. With the goal of minimizing the squared error, we use negative squared error (*Euclidean distance*) as the similarity measure, i.e., $s(i, k) = -\|x_i - x_k\|^2$. Since unsupervised learning is a notoriously difficult task, it seems impossible to obtain a perfect clustering result. As a result, in addition to several clearly separated *bot clusters* (i.e. clusters of bots) and *benign clusters* (i.e. clusters of benign hosts), we expect some clusters to include both benign hosts and bots, as shown in Section IV-B, which we call *mixed clusters*. For ease of presentation, bot clusters and mixed clusters are collectively called *bot-included clusters*. In the following section, we will show how we use supervised learning to identify and further examine bot-included clusters, as well as the method to pinpoint bots inside them.

C. Dynamic Group Behavior Analysis

In this section, we introduce DGBA with the objective of identifying bot-included clusters. DGBA is based on our

[1]The peak time is expressed as $P = \arg\max_j N_j$ with flow amount $N_P = \max_j N_j$, and the dip time as $D = \arg\min_j N_j$ with flow amount $N_D = \min_j N_j$, $j = 1, \cdots, 8$.

[2]Exemplar represents for the cluster center that best accounts for the data in the cluster [12].

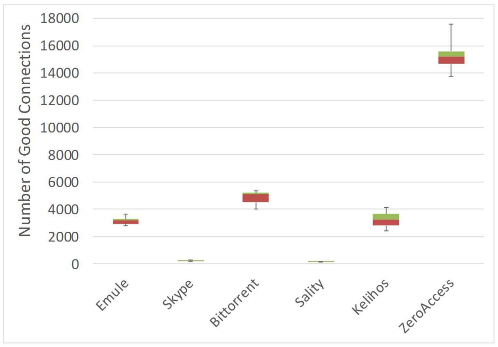
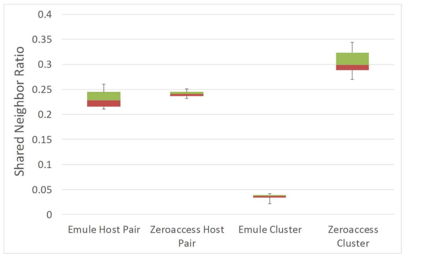
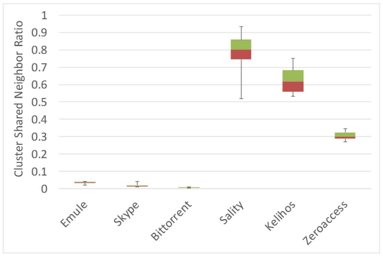


Fig. 2: Cluster connectivity feature



(a)



(b)

Fig. 3: (a): The shared neighbor ratio of one Emule host pair compares with that of one Zeroaccess host pair. (b): Group shared neighbor ratio.

intuition that the bot-included clusters have cluster-level aggregated characteristics that are distinguishable from benign clusters. Whereas the connection activity of a single bot is highly dynamic and hard to distinguish, we believe the group connection behavior will help us identify bots' communications.

DGBA training and DGBA detection are two modules that extract features from the training set and testing set respectively. The purpose of DGBA training is to extract the representative group behavior from a collection of labeled P2P hosts to build SVM classifiers, whereas DGBA detection searches for the abnormal behaviors from every unlabeled cluster to catch P2P bots. Thus, we propose to use different statistics of the collected host-level features from a group to represent group-level training and detection features, respectively. Specifically, the training features capture the average group behavior, while the detection features capture the extreme group behavior (i.e. the maximum or the minimum). Note that all the features below are extracted from the collection of traces inside each detection period if not otherwise stated.

1) Cluster Connectivity Feature: Cluster connectivity feature captures the aggregated connectivity of the peers inside each cluster. A connection between two hosts can be good or bad. We define a *good connection* as a successfully established connection between two hosts, and a *bad connection* as a failed connection. We consider a TCP connection as good if it completes a SYN, SYN/ACK, ACK handshake, and a UDP connection as good if there is at least one UDP "request" packet and a consequent UDP "response" packet. We denote the good connection set of host h_i as $\mathcal{C}_i$ which includes all the good connections of host h_i.

Training feature: The cluster connectivity feature for DGBA training is defined as the average number of good connections among all the P2P hosts of each type, i.e., $\sum_{i=1}^{M} |\mathcal{C}_i|/M$, assuming M hosts of one specific type exist in the training set.

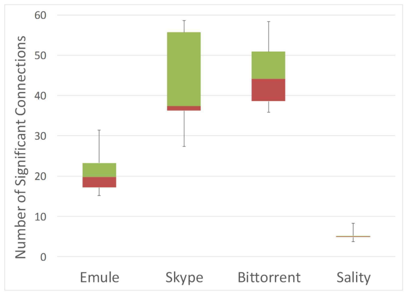
In order to see the discriminating strength of this feature, we run an experiment using 24-hour training data (refer to §IV-A for the data sets used in the experiment) to show the cluster connectivity features of different P2P bots and legitimate hosts running various P2P applications. The box-plot results are shown in Fig. 2, from which we notice different types of P2P hosts indeed exhibit disparate cluster connectivity features. In particular, ZeroAccess bot stands out with a significantly larger amount of good connections. We attribute the difference to several factors including: (1) the botnet network size; (2) the botnet peer discovery mechanisms. For instance, the bots in a populous network with a more aggressive peer discovery mechanism are supposed to have more network connections.

Detection feature: The cluster connectivity feature for DGBA detection is defined as the maximum number of good connections among all the hosts in the unlabeled cluster, i.e., $\max_{i=1}^{M'} |\mathcal{C}_i|$, assuming M' hosts in the cluster. Fig. 2 shows a notable gap between ZeroAccess bots and other types of hosts, thus this detection feature can help detect the ZeroAccess bots.

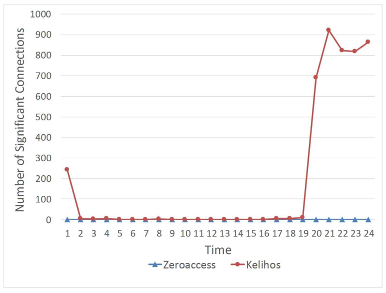
2) Shared Neighbor Feature: The shared neighbor feature captures the amount of shared connections between every pair of hosts in each cluster. The set of shared neighbors of host h_i and h_j can be written as: $\mathcal{C}_i \bigcap \mathcal{C}_j$. We further define *pairwise shared neighbor ratio* of a host pair as the ratio between the number of shared neighbors and the number of total neighbors, i.e., $s_{ij} = \|\mathcal{C}_i \bigcap \mathcal{C}_j\| / \|\mathcal{C}_i \bigcup \mathcal{C}_j\|$ for the host pair (h_i, h_j).

Training feature: Given the above definitions, shared neighbor feature for DGBA training is represented by *group shared neighbor ratio*, simply defined as the average pairwise shared neighbor ratio among all the host pairs of one type, i.e., $\sum_{i,j \in [1,M], i \neq j} \{s_{ij} / \frac{M(M-1)}{2}\}$. Previous work has adopted pairwise shared neighbor ratio s_{ij} [3] to distinguish between bots and benign hosts. However, according to our experiment, pairwise shared neighbor ratio seems ineffective in identifying certain pairs of P2P bots. In Fig. 3(a), we compare the pairwise shared neighbor ratio of an emule host pair (who download the same file) with that of a ZeroAccess bot pair. We find it almost impossible to make a distinction between these two pairs, which brings false positives or false negatives. In contrast, group shared neighbor ratio clearly differentiates ZeroAccess bots from the emule hosts with a large gap between them, via feature aggregation from multiple hosts.

In addition, Fig. 3(b) shows that different types of P2P bots and P2P hosts exhibit distinguishable shared neighbor features, where we observe that P2P bots have much higher group shared neighbor ratios compared with legitimate P2P hosts. The reason is obvious - the bots from the same botnet *search for the same commands* published by the botmaster [3], which makes their contacted peers more likely to be shared by other companions. Furthermore, although P2P botnets have a decentralized C&C architecture, botmasters still strive to make their P2P network robust against peer churns and provide end-to-end communication with a minimum delay. This inherent C&C objective translates into a convergence of contacted peers by a group of bots to ensure the reliable delivery of C&C messages. On the other hand, different legitimate P2P hosts generally search for different contents from their peers, which



(a)



(b)

Fig. 4: (a): Significant connection feature. (b): Significant connection feature of Kelihos and ZeroAccess bots.

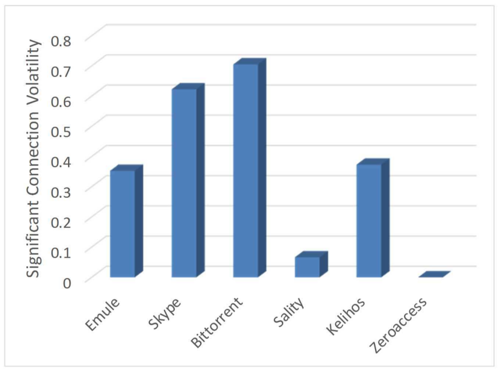


Fig. 5: Significant connection volatility

yields a more dispersed peer list.

Detection feature: Correspondingly, the shared neighbor feature for DGBA detection is defined as the maximal pairwise shared neighbor ratio among all the host pairs in each cluster, i.e., $\max_{i,j\in[1,M'],i\neq j} s_{ij}$. The shared neighbor feature of every bot-included cluster is again a distinguishing feature of the bots, since bots have significantly higher shared neighbor ratios than benign hosts, which will help discover the presence of bots in the cluster.

3) Significant Connection Feature: The significant connection (SC) feature captures the amount of *hot links* in the network, i.e., the connections that contribute significantly larger amounts of network flows compared with the other connections. The SCs extracted from the Internet traffic data have been used to diagnose the network operation and quickly identify the anomalous events [13]. Similarly, we try to identify SCs of bot groups for better understanding the bots' behaviors and accurately identifying bots' presence.

Training feature: We define the SC feature for DGBA training as the average number of SCs for all the hosts of one type. Fig. 4 shows the SC features of Sality bots and three other types of P2P hosts. Compared with Sality bots, these legitimate P2P hosts produce a larger number of SCs.

This distinctive observation may be attributed to the following fact: the SCs in a botnet indicate the existence of some active bots that are critical to the P2P botnet infrastructure. These active bots may be well connected with a high bandwidth connection, or may be close to the botmaster. Few number of distinctive connections helps the bots remain stealthy under the radar of numerous intrusion detection systems. In contrast, benign P2P hosts yield a much higher number of SCs due to their unorganized nature.

Interestingly, the traffic flows from ZeroAccess and Kelihos bots reveal unique SC patterns as shown in Fig. 4(b). ZeroAccess bots simply have none SCs, while Kelihos bots suddenly generate a large amount of SCs from a "hot" period between $7pm$ to $1am$. This period perhaps can be interpreted as a peak period of C&C message exchanging, with so many suddenly emerging hot links. The study of this abrupt phenomenon and the exact origin of SCs of botnets are out of scope of this paper, but may become research problems on their own rights.

Detection feature: Among all the hosts in the cluster, SC feature for DGBA detection is defined as the minimal number of SCs, or the maximal number if it exceeds an empirical threshold α. Thus in most cases, the SC feature of bot-included cluster will be dominated by the bots with less SCs. However, the number of SCs of Kelihos bots skyrockets during the "hot" period, which far exceeds that of the normal hosts. Hence, by setting an appropriate threshold α (e.g. 200), the SC feature of the bot-included cluster is again a distinguishing feature of bots.

4) Temporal Feature: Lastly, the temporal feature captures the dynamic evolvement of SC sets. Instead of performing feature extraction per one-hour detection period, temporal features are computed at the end of each day to combat noise and disturbance, which are represented by *significant connection volatility*, measuring whether the cluster has the same set of SCs over time. We assume the number of distinct SCs for host h_i over the day is U_i, and the number of SCs during k-th hour is S_{ik}, $i = \{1, \ldots, M\}$, $k = \{1, \ldots, 24\}$. SC volatility of host i is defined as: $\Phi_i = \frac{U_i}{\sum_{k=1}^{24} S_{ik}}$. Obviously, if the SC sets of the 24 hours are all different, we have $\Phi_i = 1$. On the contrary, when the same set of SCs appears every hour, we have $\Phi_i = 1/24$. In general, the less volatile the set of SCs is, the closer Φ_i is toward zero.

Training feature: The temporal feature for DGBA training is represented by the average SC volatility of all the hosts of the same type, expressed as: $\frac{1}{M}\sum_{i=1}^{M}\Phi_i$. Fig. 5 shows different temporal features for various P2P bots and legitimate P2P hosts. We notice that Sality and Zeroaccess bots have small volatility features, while emule hosts and Kelihos bots have a moderate value of SC volatility. SC volatility is related to a number of factors, such as the number of SCs, the size of P2P networks and how dynamic the network connections are.

Detection feature: The temporal feature for DGBA detection captures the minimal SC volatility of all the hosts in the cluster, i.e., $\min\Phi_i$. Therefore, the temporal feature of bot-included cluster will be determined by the bots, whose SC sets appear less volatile. In the end, a low value of temporal feature immediately reveal the presence of bots.

D. Training and Classification

Data Preprocessing: Data preprocessing tries to cope with the issue that the collective features extracted from the network flow data have different data ranges. To make sure every feature in the feature sets is given equal importance, we perform feature-wise normalization to shift and re-scale each feature value so that they lie within the range of $[0, 1]$.

Multi-class Classification: Support vector machine (SVM) is adopted as our main classification method due to its robustness, efficiency, and excellent non-linear classification performance. In particular, we use multi-class SVM classification to assign each cluster one label corresponding to a specific type of botnet or a non-bot host. We denote the multiple labels as $\{B_1, B_2, \ldots, B_k\}$, assuming $k-1$ classes of botnets with the last class representing non-bot label. The basic component of the SVM method is a binary classification mechanism, which classifies an unlabeled cluster based on the distance of its feature to the decision hyperplane with norm vector $\mathbf{w}$ and constant b:

$$f(x) = \mathbf{w}^T\mathbf{x} + b = \sum_{\forall i} y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b, \tag{1}$$

where $\mathbf{x}_i$ is the feature vector of host i from the training set, $y_i \in \{-1, 1\}$ denotes the label of the training data, and the parameters α_i determines whether the host i is a support vector ($\alpha_i > 0$) or not ($\alpha_i = 0$). The feature vector $\mathbf{x}_i$ is transformed into a higher dimensional space by a non-linear kernel function $K(\mathbf{x}_i, \mathbf{x})$.

The two-class SVM determines $\mathbf{w}$ and b by searching for the optimal hyperplane to separate the feature space into two parts. This is also termed as a maximum margin approach, since the objective is to maximize the distance between training data and decision hyperplane. The multi-class SVM model is built by combining multiple two-class SVM models. For a K-class SVM model ($K > 2$), we use "*one-versus-one*" approach [14], in which $K(K-1)/2$ classifiers are trained on all possible pairs of classes, and then a voting strategy is used to classify the clusters to the corresponding classes with the highest number of votes. The clusters labeled as specific types of botnets become bot-included clusters, demanding a further inspection.

E. Refined Bot Identification

After labeling bot-included clusters, the final step is to extract bots from the cluster based on their individual connection features. Utilizing the experimental results of training features, we devise a *feature test* to separate bots from benign hosts who happen to be in the same cluster. The feature test (see pseudo code in Algorithm 1) exploits the differences of various connection features between bots and benign hosts. A number of threshold values are defined to empirically set apart bots and benign hosts (e.g. $\lambda_1 = 8000$, $\lambda_2 = 0.2$, $\lambda_3 = 10$, $\lambda_4 = 200$, $\lambda_5 = 0.2$). As long as one type of features satisfies the statement, the host is identified as bot.

Algorithm 1 Feature Test

```
1: for each bot-included cluster do
2:     for each host in the cluster do
3:         host ← benign host label
4:         if number of connections > $\lambda_1$ or
           shared neighbor ratio with any peer > $\lambda_2$ or
           number of significant connections < $\lambda_3$ or > $\lambda_4$ or
           significant connection volatility < $\lambda_5$ then
5:             host ← bot label
6:         end if
7:     end for
8: end for
```

F. Evasion mechanisms and Limitations

PeerClean detects botnets without relying on deep packet inspection, which already raises the bar for botnet authors. In the following, we discuss the potential evasion mechanisms that botnet authors might use to circumvent PeerClean.

The bots may disrupt the clustering mechanism by not following the same transmission protocol. However, that will increase the complexity of bot implementations and will also affect the efficiency of C&C message exchange. Evading DGBA is even harder to achieve. The possible attempts to evade the DGBA detection include lowering the connection number, lowering the shared neighbor ratio, raising the significant connection number, and raising the significant connection volatility. The change of one or more connection features will greatly affect the P2P network operation and may compromise the stealthiness of individual bots. The collective features enlarge the gaps between the bots and benign hosts. To make the collective features indistinguishable from those of benign hosts will require substantial work on designing a complex botnet.

Since PeerClean identifies the bots based on the traffic flow statistics from every host, it becomes a particular challenge to identify a bot-infected host if it also runs legitimate P2P applications simultaneously and persistently. In this case, the bot traffic might be obscured by the traffic from P2P legitimate applications. Since PeerClean performs detection per hour, the smart bots would have to run P2P legitimate applications all the time to prevent from being detected. We find this either unlikely or costly for the bots to achieve. Most P2P nodes have a fast peer churn rate with short communication sessions [11]. Thus, it is unlikely for a P2P host, on its own course, to run legitimate P2P applications with a P2P bot protocol persistently, which would give away the bots at a certain point of time. On the other hand, the future bots might intentionally run the bot protocol together with legitimate P2P applications. Nevertheless, this will affect the communication efficiency of P2P bots, and might lead to a high peer churn rate or even a complete disruption of C&C communications.

IV. Evaluation

In this section, we evaluate the bot identification performance of PeerClean. We first describe the collected data sets (§IV-A). Then, we show that PeerClean can well separate different types of P2P bots into different clusters, but may falsely include some benign P2P hosts who have bot-like traffic patterns (§IV-B). After generating host clusters, DGBA is carried out to extract group-level connection features from each cluster. By separating the data set into training and testing sets, a multi-class SVM model is trained using the labeled training set. Finally, we evaluate the classification performance and the refined bot identification performance in §IV-C and §IV-D, respectively.

A. Data Collection

We use the traffic trace captured from the edge routers of a large campus network, comprised of two $/16$ subnets. The traffic rate is about 5000 flows per second, and was captured for one whole day. We focus on the TCP and UDP traffic in this traffic trace. However, as the network flow trace does not include traffic payload, we do not have the ground truth whether or not the active hosts are running legitimate P2P applications.

To establish the ground truth data from legitimate P2P hosts, we run three of the most popular P2P applications in our

Trace	**Size**	**Dur**	**Pkts**	**TCP/UDP Flows**	**clients**
Campus	20.7G	24h	21.5G	401,661,350	34743
Bittorrent	6.7G	24h	854M	62,674,080	100
Skype	1.1G	24h	376M	12,615,840	100
Emule	1.6G	24h	406M	18,978,800	100
Sality	40M	24h	10.8M	565,490	6
Kelihos	224M	24h	23.5M	3,249,931	4
ZeroAccess	4.6G	24h	166.9M	69,896,829	4
P2P in campus	487M	24h	608M	7,127,054	783

TABLE III: Traffic summary ('P2P in campus' denotes the traffic flows of the campus network after P2P host identification)

lab machines: Emule, BitTorrent, and Skype, and collect their network flow traces. To make the traffic traces more representative, we interact with the P2P hosts using AutoIt script [15] to randomly select contents to be downloaded/uploaded (for emule and bittorrent application), or randomly generate texts to be transmitted (for skype application) at random time periods. In total, we collected one-day traces from 100 Bittorrent clients, 100 Skype clients and 100 Emule clients.

We also collected the network traces for three recent P2P botnets: Sality, Kelihos and ZeroAccess. These network traces were gathered by purposefully running Sality, Kelihos and ZeroAccess samples in a controlled environment, in which we carefully block spamming, scanning, and Denial of Service attack activities. They contain 24-hour traces for 6 Sality bots, 4 Kelihos bots and 4 ZeroAccess bots. Since the major malicious activities were blocked during the collection of network traces, the collected traces mainly include C&C traffic, e.g., for peer discovery, command exchanging, etc. Note that these traces are collected when the three botnets are fully active. The traffic summary is listed in Table III. The traffic data from 300 legitimate P2P clients and 14 P2P bots constitute our ground truth data set.

To make the evaluation more realistic, we used the traffic flow traces collected from our campus network (a production network) as the background traffic and added the traffic traces collected from 300 legitimate P2P hosts and 12 P2P bots into the campus traffic traces. In order to reduce the traffic volume to be processed by PeerClean, we eliminate flows from well-known and extremely busy servers such as DNS servers, email servers, popular website servers (e.g. google, facebook, youtube, etc.). After that, P2P host identification searches for the hosts with a high percentage of failed connections (with threshold of 5%). In total, we find 1097 hosts involved in P2P applications during the day, including all the 314 P2P hosts serving as ground truths and additional 783 hosts in the campus network as shown in Table III. This result shows that our P2P host identification mechanism is effective, as we do not miss a single one from our ground truth host set.

B. Clustering P2P Hosts

In this section, we evaluate the P2P host clustering performance of the PeerClean system. Based on the extracted flow features in §III-A, we perform AP clustering to group together P2P bots of the same type. During the flow feature extraction, we find that almost all of the traffic flows from Kelihos bots adhere to TCP protocol, while Sality and ZeroAccess bots mostly generate UDP traffic. Hence, we use both the TCP and UDP traffic patterns for host clustering.

The data set contains 24-hour flow traces from 1097 P2P hosts, which is divided into 24 sections with one hour per

Hour	2	4	6	8	10	12	14	16	18	20	22
Cluster Num.	29	25	24	27	30	29	32	25	29	30	28
Sality Cluster Index	28	22	22	26	28	27 28	31	22	22	29	26
ZeroAccess Cluster Index	29	25	24	27	30	29	32	25	29	30	28
BSR_1	1	1	1	1	1	1	1	1	1	1	1
BSR_2	1	1	1	1	1	1	1	1	1	1	1

TABLE IV: Clustering result using UDP traffic (BSR_1, BSR_2 denotes the BSRs of Sality and ZeroAccess bots respectively)

Hour	2	4	6	8	10	12	14	16	18	20	22
Cluster Num.	21	24	25	15	22	25	26	21	24	20	18
Kelihos Cluster Index	13	24	23	14	19	6	5	20	6	14	17
BSR_3	1	1	1	1	1	1	1	1	1	1	1

TABLE V: Clustering result using TCP traffic (BSR_3 denotes the BSR of Kelihos bots)

section. For each data section, we extract the flow statistical features of every host who has 100+ outgoing TCP flows and 100+ incoming TCP flows for the purpose of building representative flow patterns. Then, host clustering is carried out using the AP clustering method based on the extracted flow statistical features. Note that, since the last two features in Table II are refreshed at the end of the day, they will only be used for clustering in the last hour.

We evaluate the clustering performance in terms of the ability of producing well separated and compact bot. We propose two performance criterion. We define *cross-clustered bots* as the bots falsely clustered together with other types of bots, and *correctly-clustered bots* as the bots separately clustered. The two performance criterion for evaluating the separation and compactness performances of bot clustering are: (1) *Bot Separation Ratio* (BSR), which is defined for each type of bot as the ratio of the number of correctly-clustered bots over the total number of bots of this type; (2) *Bot Compactness Ratio* (BCR), which is defined as the ratio of the number of correctly-clustered bots over the total number of hosts (whether benign or not) assigned into the same cluster.

Bot Separation Performance: As observed from the per-hour clustering results[3] in Table IV and V, Sality, Kelihos, and ZeroAccess bots are assigned into different clusters with the cluster index shown in the tables, i.e., all three types of bots are well separated from each other, which raises their BSRs to one. Moreover, almost all the bots of one type are grouped into the same cluster, with an exception of Sality bots who are divided into two clusters at the 12-th hour. Nevertheless, none of these clusters contains more than one type of bots, which demonstrates the perfect separation of different types of bots.

Bot Compactness Performance: The perfect bot separation performance indicates that each bot-included cluster only includes one type of bots, although it may also include some benigh P2P hosts, which happen to have a similar traffic pattern during the detection period. BCR quantifies the clustering capability to preclude the inclusion of the benign P2P hosts in the bot-included clusters. If a bot-included cluster contains zero benign hosts, its BCR value would be 1. Accumulating the 24-hour BCR results, we plot BCR box-

[3]Note that we only count the clusters containing more than one node.

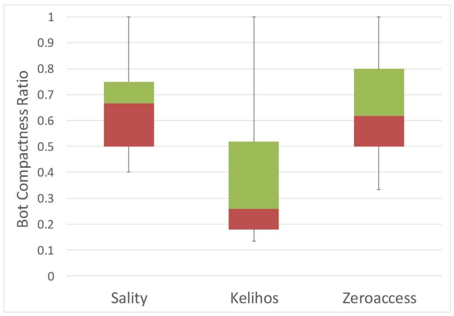


Fig. 6: Box plot of Bot Compactness Ratio

plot performance of three types of bots in Fig. 6. On average, Sality and ZeroAccess bot clusters falsely include 3 benign hosts respectively, while Kelihos bot clusters falsely include 12 benign hosts. Further inspection of the falsely included benign hosts shows that they have traffic profiles that are highly similar to the bots' traffic. This experimental result shows that the proposed clustering mechanism based on network flow features are subject to false positives. In other words, the network flow features alone are not sufficient to discriminate accurately P2P bots from benign P2P hosts.

C. Identifying Bot-included Clusters via Classification

In this section, we evaluate the bot cluster identification method. Since we only have a limited number of labeled bots during every hour, we enlarge the training space by incorporating a half day of labeled bots and benign hosts into the training set. Consequently, the training set contains 36 clusters (3 clusters per hour) of labeled bots (with labels 'Sality', 'Kelihos', 'ZeroAccess') and 36 clusters of labeled legitimate P2P hosts running Bittorrent, Emule and Skype (with labels 'Non-Bot'). We extract the training features from all the 72 labeled clusters to build the SVM classifier. Then, we use the next half day of bots and benign hosts as testing set, which includes a total of 37 bot-included clusters and 545 benign clusters.

After host clustering, the DGBA detection process extracts four different types of group-level connection features from every cluster in the testing set. Then, the SVM model predicts the labels of clusters. Since the classification module relies on four different types of features, we train the classifier on each individual group behavior feature in order to understand their relative importance.

Table VI shows the classification performance using different types of features for training. The classification based on either shared neighbor feature or significant connection feature have high accuracy and recall, but only achieve moderate precision. Looking into the classification results, we find that the classification produces few false negatives but many false positives, i.e., bot-included clusters are unlikely to be regarded as benign cluster with these two features, but many benign clusters are falsely considered as bot-included clusters. On the other hand, cluster connectivity feature seems unable to discriminate bots from benign hosts, as it brings substantial false positives and false negatives. We observe that the correctly classified bots mainly belong to ZeroAccess botnet, which is consistent with our analysis in Section III-C1. Finally, temporal feature is designed to be updated at the end of the day, thus is only used for bot classification in the final hour.

Group Behavior Feature	**Accuracy**	**Precision**	**Recall**
Cluster Connectivity Feature	51.8%	7.9%	34.3%
Shared Neighbor Feature	92.7%	68.8%	91.7%
Significant Connection Feature	91.8%	66.7%	90%
Temporal Feature	71.3%	3.1%	66.7%
All Features	98.8%	94.6%	100%

TABLE VI: Classification accuracy when trained on one type of feature. Shared neighbor feature and significant connection feature present the best classification accuracy. The classifier achieves the best performance when combining all the features. Accuracy=(TP+TN)/all; Precision=TP/(TP+FP); Recall=TP/(TP+FN).

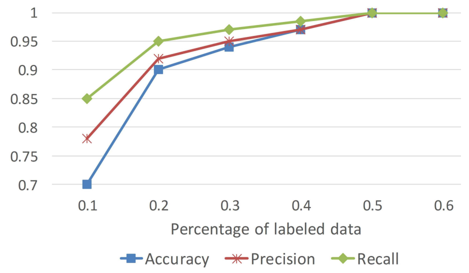


Fig. 7: Classification performance with different percentages of training data

Again, many false positives arise due to the inseparability of bots' features and benign hosts' features. However, the combination of all features provide the best result for detecting bot-included clusters. Overall, we only have two false positives and zero false negatives.

It is worth noting that the training set constitutes 50% of the whole data set in the previous evaluation. Here, we also evaluate the classification performance by varying the percentage of training data, since it is always difficult to collect traffic traces from labeled bots. As shown in Fig. 7, PeerClean can still retain more than 70% classification accuracy when the training sets contain traces from merely 10% of labeled hosts. This suggests that PeerClean is robust against small sets of training data, and may have wide applicability under different network sizes.

D. Refined Bot Identification Performance

Bot-included clusters contain a considerable amount of benign hosts as shown in Section IV-B, thus we use refined bot identification to extract the bots inside each bot-included cluster. The feature test in Algorithm 1 is utilized to perform refined bot identification. We run the feature test on all the 39 bot-included clusters identified through SVM classification, including 13 Sality clusters, 12 Kelihos clusters, 12 ZeroAccess clusters and 2 false positives. The bot identification performance is reported in Table VII, which shows the bot number and benign host number in the bot-included clusters. In summary, the refined bot identification method correctly identifies more than 95.8% of bots, and falsely triggers less than 4.8% alarms.

V. Related Work

The increasing popularity of P2P botnets has led to a vast amount of research that attempt to track and remove them. In the literature, the detection mechanisms can be

Bot Type	**Bot Num.**	**Benign Host Num.**	**Correctly Identified**	**Falsely Identified**
Sality	72	36	**69**(95.8%)	**0** (0%)
Kelihos	48	123	**47**(97.9%)	**5** (4.1%)
ZeroAccess	48	42	**48**(100%)	**2** (4.8%)

TABLE VII: Refined bot identification performance (the percentage in the parenthesis denotes the bot detection rate and false alarm rate respectively)

classified into two categories: host-based approaches and network-based approaches. The latter can be subdivided into network traffic-based approaches and communication graph-based approaches. We now review some additional works from the second category that are most related to our work.

Network traffic-based approaches: Some related works utilize attack traffic characteristics to identify hosts with similar abnormal network behaviors, such as spamming, port scanning, sharing the same packet contents [16], or, having common destinations, similar payloads and common host platforms [17]. However, these approaches can be evaded by manipulating attacking strategies.

Several works focused on identifying C&C traffic from the botnets. Bilge et al. [9] proposed to use NetFlow analysis to distinguish botnet C&C servers from benign servers by extracting flow-level features from the data. Wurzinger et al. [18] identify C&C by automatically extracting signatures from bot responses after receiving commands. However, this approach can be circumvented by traffic encryption. Moreover, the above approaches, which use only flow-level statistics, are not robust enough to produce accurate detection results. Instead, PeerClean greatly enhances the detection capability by jointly considering the flow-level traffic statistics and group-level network connection behaviors.

Communication graph-based approaches: In [19], Coskun et al. proposed to identify the local members of P2P bots using mutual contacts graph. However, this method requires to start with a captured seed bot in the network, which may not be available. Jelasity et al. [20] argued that it is difficult to detect P2P bots using traffic dispersion graph (TDG) especially with a limited view of the Internet traffic at a single AS. Most recently, Li et al. [21] proposed to detect P2P community by identifying the densely connected subgraphs. However, this approach only focused on a backbone network which requires a very large communication graph. Also, solely relying on the connection patterns, it may falsely include lots of benign hosts in the discovered P2P botnets.

VI. Conclusion

P2P C&C infrastructure has become a popular choice for the future botnets, which is extremely resilient to even sophisticated takedown measures. The ability to identify botnets inside a network is particularly important to the network administrators. Toward this direction, we present PeerClean, a new network flow-based system to identify and classify botnets with a high accuracy. The main novelty in the design of PeerClean is the use of group-level behavior analysis and the novel strategy of using the average behavior as the training feature but the extreme behavior as the detection feature. Our extensive experimental results show that the group-level connection features are more robust. Together with the proposed SVM training and detection processes, PeerClean is shown to be very effective in detecting several known botnets. Based on the underlying rationales behind these proposed group-level features to capture the characteristics of bots, we believe PeerClean is also able to detect unseen and adaptive botnets, as the group-level features effectively help to distinguish the bots' characteristics from benign P2P hosts' characteristics. An interesting future direction is to apply the group behavior analysis to other types of applications to help identify the network behaviors which would be otherwise unnoticeable. PeerClean could also be tuned to perform anomaly detection to identify unseen bots and the performance of which is yet to be understood.

References

[1] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov, "Botgrep: Finding p2p bots with structured graph analysis," in *Proc. of USENIX Security'10*, 2010.

[2] T.-F. Yen and M. K. Reiter, "Are your hosts trading or plotting? telling p2p file-sharing and bots apart," in *Proc. of ICDCS*, June 2010.

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

[4] Z. Xu, L. Chen, G. Gu, and C. Kruegel, "Peerpress: Utilizing enemies' p2p strength against them," in *Proc. of ACM CCS'12*, October 2012.

[5] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X. Zhou, and X. Wang, "Effective and efficient malware detection at the end host," in *Proc. of USENIX Security'09*, August 2009.

[6] G. Gu, P. Porras, V. Yegneswaran, and M. Fong, "Bothunter: Detecting malware infection through IDS-driven dialog correlation," in *Proc. of USENIX Security'07*, August 2007.

[7] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proc. of USENIX Security'08*, 2008.

[8] J. Zhang, X. Luo, R. Perdisci, G. Gu, W. Lee, and N. Feamster, "Boosting the scalability of botnet detection using adaptive traffic sampling," in *Proc. of AsiaCCS*, March 2011.

[9] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "DISCLOSURE: Detecting botnet command and control servers through large-scale netflow analysis," in *Proc. of ACSAC*, Dec. 2012.

[10] T.-F. Yen and M. K. Reiter, "Revisiting botnet models and their implications for takedown strategies," in *Proceedings of the First international conference on Principles of Security and Trust*, 2012.

[11] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, October 2006.

[12] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[13] K. Xu, Z.-L. Zhang, and S. Bhattacharyya, "Profiling internet backbone traffic: Behavior models and applications," in *Proc. of SIGCOMM*, August 2005.

[14] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[15] "Autoit script," http://www.autoitscript.com/site/autoit/.

[16] G. Gu, J. Zhang, and W. Lee, "Botsniffer: Detecting botnet command and control channels in network traffic," 2008.

[17] T.-F. Yen and M. K. Reiter, "Traffic aggregation for malware detection," in *Proc. of DIMVA '08*, 2008.

[18] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, and E. Kirda, "Automatically generating models for botnet detection," in *Proc. of ESORICS'09*, 2009, pp. 232–249.

[19] B. Coskun, S. Dietrich, and N. Memon, "Friends of an enemy: Identifying local membersof peer-to-peer botnets using mutual contacts," in *Proc. of ACSAC*, 2010.

[20] M. Jelasity and V. Bilicki, "Towards automated detection of peer-to-peer botnets: on the limits of local approaches," in *Proc. of LEET'09*, 2009.

[21] L. Li, S. Mathur, and B. Coskun, "Gangs of the internet: Towards automatic discovery of peer-to-peer communities in the internet," in *Proc. of CNS*, 2013, pp. 167–175.