# Bot mitigation: Using A graph-based machine learning approach.

Gagan Ganapathy
gaganganapathyas@gmail.com

Dr. Satish Kumar Singh
sk.singh@iiita.ac.in

***Abstract*—Bot mitigation using machine learning (ML), with network flow-level features, has been extensively studied in the literature. However, existing flow-based approaches typically incur a high computational overhead and do not completely capture the network communication patterns, which can expose additional aspects of malicious hosts. Recently, bot detection systems which leverage communication graph analysis using ML have gained attention to overcome these limitations. A graph-based approach is rather intuitive, as graphs are true representations of network communications. In this paper, we propose a two-phased, graph-based bot detection system which leverages both unsupervised and supervised ML. The first phase prunes presumable benign hosts, while the second phase achieves bot detection with high precision. Our system detects multiple types of bots and accommodates different network topologies and is suitable for large-scale data.**

***Index Terms*—Bot-detection, machine learning, graph-based approach**

## I. Introduction

Organizations are constantly under security threats, which not only cost billions of dollars in damage and recovery, but also detrimentally affect their reputation. A botnet-assisted attack is a widely known threat to these organizations. According to the U.S. Federal Bureau of Investigation, "Botnets caused over $9 billion in losses to U.S. victims and over $110 billion globally." The most infamous attack, Rustock, infected 1 million machines, sending up to 30 billion spam emails a day [13]. More recently, WannaCry resulted in data breach from over 230,000 computers in 150 countries [10].

A botnet is a collection of bots, agents in compromised hosts, controlled by botmasters via command and control channels. A malevolent adversary controls the bots through botmaster, which could be distributed across several agents that reside within or outside the network. Hence, bots can be used for tasks ranging from distributed denial-of-service (DDoS), to massive-scale spamming, to fraud and identity theft. The intrusion kill-chain [6] dictates the general phases a malicious agent goes through in-order to reach and infest its target.

Detection of bots can be largely achieved via intrusion detection systems (IDSs), which can be broadly classified into signature-based and anomaly-based [6]. Signature-based methods use pre-computed hashes of existing malware binaries. . They scale well and efficiently detect known threats. However, they require frequent database updates and can be subverted by unknown or modified attacks, such as zero-day attacks and polymorphism [17], [9]. This undermines their suitability for bot detection. Anomaly-based methods overcome these limitations. Machine learning (ML) is an ideal technique to automatically capture the normal behavior of a system. Its use has boosted the scalability and accuracy of IDSs [14], [7].

An important step prior to learning, or training a ML model, is feature extraction. These features act as discriminators for learning and inference, reduce data dimensionality, and increase the accuracy of ML models. The most commonly employed features in bot detection are flow-based (e.g., source and destination IPs, protocol, number of packets sent and/or received, etc.). However, these features do not completely capture the communication patterns that can expose additional aspects of malicious hosts. In addition, flow-level models can incur a high computational overhead, and can also be evaded by tweaking behavioral characteristics e.g., by changing packet structure [4].

Graph-based features, derived from flow-level information to reflect the true behaviour of hosts, are an alternate that overcome these limitations. We show that incorporating graph-based features into ML yields robustness against complex communication patterns and unknown attacks. Moreover, it allows for cross-network ML model training and inference.

The rest of the paper is organized as follows. In Section II, we present a background on bot detection and highlight limitations of the state-of-the-art. Our system design is delineated in Section III. We discuss the results of our experiments in Section IV. In Section V, we conclude with a summary of our contributions and instigate future research directions.

## II. Related Work

Bot detection has been an active area of research and has generated a substantial body of work. Most existing bot detection techniques employ methods for detecting C2 channels based on the statistical features of packets and flows [11], [5]. Solutions like [11], [2] are focused on specific communication protocols, such as IRC, providing narrow-scoped solutions. On the other hand, Botminer [8] is a protocol-independent solution, which assumes that bots within the same botnet are characterized by similar malicious activities and communication patterns. This assumption is an over simplification, since botnets often randomize topologies [17] and communication patterns as we observe in newer malware, such as Mirai [12]. Therefore, it is evident that a nonprotocol-specific, less evadable detection method is desired.

Furthermore, [19], [5] have exploited ML-driven anomaly

detection with traffic-based statistical features, for detecting known and unknown attacks with low error rates. However, such techniques require that all flows are compared against each other to detect C2 traffic, which incurs a high computational overhead. Graph-based approaches, where graphs are extracted from network flows and host-to-host communication patterns, overcome these limitations [4]. BotGM [18] uses a statistical technique, the inter-quartile method, for outlier detection. Their results exhibit moderate accuracy with low FPs based on different windowing parameters. However, it generates multiple graphs from a selection of network flows. For every pair of unique IPs, a graph is constructed, such that every node in the graph represents a unique 2-tuple of source and destination ports, with edges signifying the time sequence of communication. This entails high overhead and will not scale for large datasets.

Graph-based approaches using ML for bot detection are intuitive and show promising results. In this paper, we propose an anomaly-, graph-based bot detection system, which is protocol agnostic i.e., it detects bots regardless of the protocol. We employ graph-based features in a two-phased ML approach, which is robust to zero-day attacks, spatially stable, and suitable for large datasets.

## III. Our Approach

Our bot detection system consists of 3 major components, as depicted in Fig. 1. These components pertain to data preparation and feature extraction, model training, and inference. [3]

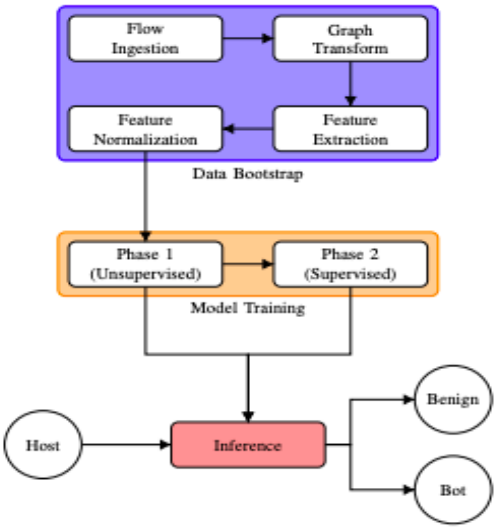



Fig. 1. Components of the bot detection system

### A. Flow Ingestion

The input to the system are bidirectional network flows. These flows are transformed into a set T that contains 4- tuple flows $t_i = \{sip_i, srcpkts_i, dip_i, dstpkts_i\}$.

Where $sip_i$ is the source IP address that uniquely identifies a source host, $srcpkts_i$ quantifies the number of data packets sent by $sip_i$ to $dip_i$, the destination host IP address.

### B. Graph Transform

The system creates a Graph *G(V,E)* where V is set of nodes and E the set of directed edges $e_{i,j}$ from node $v_i$ to $v_j$ with edge weights $|e_{i,j}| = srcpkts_x$ and $|e_{j,i}| = dstpkts_x$. Moreover, if there exists a reverse tuple such that, $dip_y = v_i$ and $sip_y = v_j$ then in this case $|e_{i,j}| = srcpkts_x + dstpkts_y$ and $|e_{j,i}| = srcpkts_y + dstpkts_x$

### C. Feature Extraction

The system creates the required graph-based feature set for the ML model. Features are intrinsic to the success of the model that should genuinely represent and discriminate host behavior, especially bot behavior. We study the following set of commonly used graph-based features. [1]

- **In-Degree (ID) and Out-Degree (OD)** The in-degree, $f_{i,0}$, and out-degree $f_{i,1}$ of a node $v_i \epsilon V$ are the number of its ingress and egress edges, respectively

$$\mathcal{F}(e_{i,j}) = \begin{cases} 1, & \text{if } e_{i,j} \in E \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$f_{i,0} = \sum_{v_j \in V,\ v_i \neq v_j} \mathcal{F}(e_{j,i}) \qquad \forall v_i \in V \tag{6}$$

$$f_{i,1} = \sum_{v_j \in V,\ v_i \neq v_j} \mathcal{F}(e_{i,j}) \qquad \forall v_i \in V \tag{7}$$

  These features play an important role in the behavior of a host. Though, a higher ID for a host makes it a point of interest, often nodes with high ID offer commonly demanded service. Therefore, observing ID alone may not signify malicious activity. For example, a gateway is a central point of network communication, but it is not necessarily a malicious endpoint. Intuitively, bots attempting to infect the network will tend to have higher ID than benign hosts.
- **In-Degree Weights (IDW) and Out-Degree Weights (ODW)** These features augment the ID and OD of the nodes in the graph. The in-degree weight $f_{i,2}$, and the out-degree weight $f_{i,3}$, of a node $v_i \epsilon V$ is the sum of all the weights of its incoming and outgoing edges, respectively.

$$f_{i,2} = \sum_{v_j \in V,\ v_i \neq v_j,\ e_{j,i} \in E} |e_{j,i}| \qquad \forall v_i \in V \tag{8}$$

$$f_{i,3} = \sum_{v_j \in V,\ v_i \neq v_j,\ e_{i,j} \in E} |e_{i,j}| \qquad \forall v_i \in V \tag{9}$$

  For a fine-grained differentiation, it is important to expose features that will eventually bring bots closer to each other, and discriminate bots from hosts. IDW and ODW features add another layer of information, further alienating the malicious hosts from the benign. Similar to ID,

mass-data leeching bots will tend to expose a high IDW in the action phase of the intrusion kill-chain. Similarly, the ODW is the aggregate data packets a node has sent, which can potentially expose bots that mass-send payloads to hosts in a network.

- **Betweenness Centrality (BC)** The betweenness centrality of a node $v_i \epsilon V$, inspired from social network analysis, is a measure of the number of shortest paths that pass through it, such that.

$$f_{i,4} = \sum_{v_j, v_k \in V,\ v_i \neq v_j \neq v_k} \frac{\sigma_{v_j v_k}(v_i)}{\sigma_{v_j v_k}} \qquad \forall v_i \in V. \qquad (10)$$

This method is highly computationally complex and is not scalable to large datasets. However, it can alienate bots early on as they attempt their first connections. This is when the bots exhibit low IDW and ODW. Thus, it would be more favorable for the shortest paths in the network to pass through the host. Likewise, when the IDW and ODW increase, the BC of a node decreases immensely, as it is less favored for being included in shortest paths.

- **Local Clustering Coefficient (LCC)** Local clustering coefficient has a lower computational overhead, and it quantifies the neighborhood connectivity of a node $v_i \epsilon V$, such that

$$f_{i,5} = \frac{\sum_{v_j, v_k \in N_i,\ v_i \neq v_j \neq v_k} \mathcal{F}(e_{j,k})}{|N_i|(|N_i| - 1)} \qquad \forall v_i \in V \qquad (11)$$

### *D. Feature Normalization (F-Norm)*

Topological alterations can severely affect the host's behavior and pattern of communication in the graph. To control the overhead of computing these normalized features, the neighborhood set $N_i$ for $v_i \epsilon V$ V is restricted to a depth D. The mean of j features for $v_i$ across its neighbors $v_k \epsilon N_i$ are computed. Each feature for $v_i$ is then normalized by incorporating neighborhood relativity. Thus, features relative to their neighborhood mean is given as

$$\mu_{i,m} = \frac{\sum_{v_k \in N_i} f_{k,m}}{|N_i|} \qquad \forall v_i \in V,\ 0 \leq m \leq j \qquad (13)$$

$$f_{i,m} = \frac{f_{i,m}}{\mu_{i,m}} \qquad \forall v_i \in V,\ 0 \leq m \leq j. \qquad (14)$$

We have use (D = 1) here due limitations of computational power and time. As aforementioned, normalization attempts to make hosts of the same nature look similar, making the topological alterations less severe. Similarly, in situations where network data is recorded over varying time intervals, IDW and ODW tend to increase substantially with larger time intervals. By normalizing features, the effect of time also diminishes.

## IV. TRAINING

The model is trained to accept graph-based features as input and learn to distinguish between malicious and benign hosts. Two learning phases are involved as explained below.

1) Phase 1

The first ML phase performs unsupervised learning (UL) to cluster the hosts. Generally, benign hosts exhibit similar behavior that can be gauged by the graph-based features. These hosts exhibit resembling patterns in data, such as sending (OD/ODW) and receiving (ID/IDW) similar number of packets [15]. Since BC and LCC are directly affected by these traits, their influence can be similar for all benign hosts. This may maximize the size of the benign cluster. This phase not only acts as a first filter for new hosts, but also significantly reduces the training data for the second phase. If a host is clustered into the benign cluster, then it is strictly benign. However, it is important to note that a malicious host can also be incorrectly clustered into a benign cluster, adversely affecting system performance. Although the objective is to maximize the size of the benign cluster, it is essential to jointly minimize the number of bots that are co-located in this cluster. Various UL techniques can be deployed in this phase, including k-Means, Density-Based Spatial Clustering (DBScan) were used.

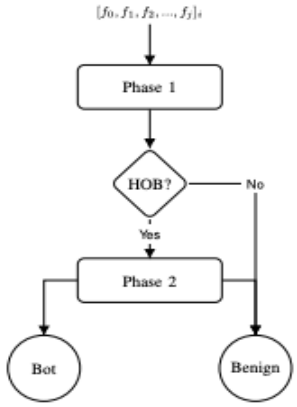
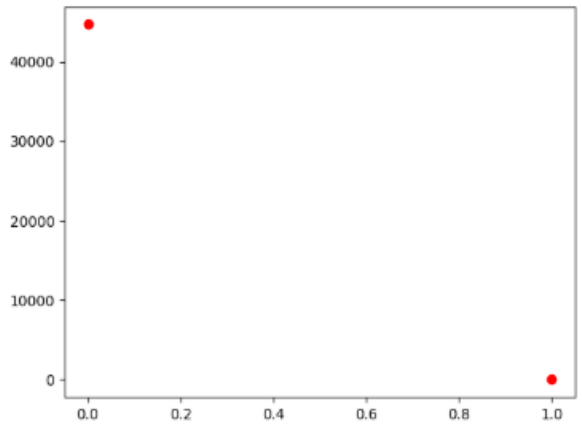


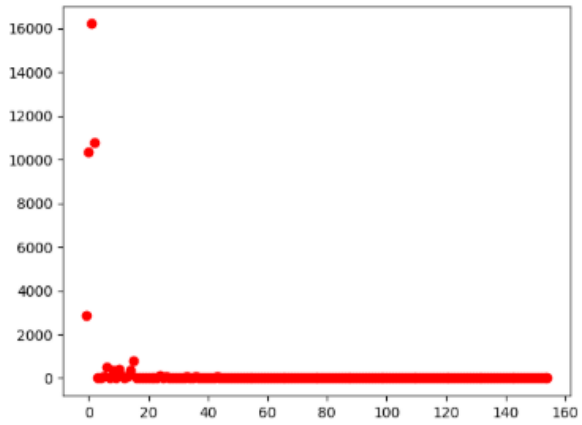

Fig. 4. Flowchart of node classification with $i$ nodes and $j$ features

The following the graphs below visualize the clusters from using the two type of clustering techniques (Kmeans and DBScan):

**Kmeans (n_clusters=2, random_state=0)**




**DBScan (eps=0.4, min_samples=4)**




2) Phase 2

Phase 1 separates the dataset between hosts that are inside and outside the benign cluster. All the hosts, ideally a small number, that reside outside the benign cluster are an input to Phase 2 for further classification. Optimally, all bots should be outside the benign cluster, regardless of whether or not they are co-located in the same cluster. Depending on the number of hosts outside the benign cluster, supervised learning (SL) classifiers in this phase will exhibit varying results. The primary objective in this phase is to maximize recall i.e., how many bots do not go unnoticed. It is proportional to the number of true positives (TPs) and inversely proportional to FNs. Various SL classifiers can be deployed to achieve this objective, such as logistic regression (LR) and the Naive Bayes Classifier.

## V. Dataset

Our evaluation is based on the CTU-13 [16] dataset. CTU13 comprises of 13 different subset datasets (DS) that include captures from 7 distinct malware, performing port scanning, DDoS, click fraud, spamming, etc. Every subset carries a

TABLE I
CTU-13 Dataset

| DS | Duration | # Flows | Bot | # Bots |
|---|---|---|---|---|
| 1 | 6.15 | 2824637 | Neris | 1 |
| 2 | 4.21 | 1808123 | Neris | 1 |
| 3 | 66.85 | 4710639 | Rbot | 1 |
| 4 | 4.21 | 1121077 | Rbot | 1 |
| 5 | 11.63 | 129833 | Virut | 1 |
| 6 | 2.18 | 558920 | Menti | 1 |
| 7 | 0.38 | 114078 | Sogou | 1 |
| 8 | 19.5 | 2954231 | Murlo | 1 |
| 9 | 5.18 | 2753885 | Neris | 10 |
| 10 | 4.75 | 1309792 | Rbot | 10 |
| 11 | 0.26 | 107252 | Rbot | 3 |
| 12 | 1.21 | 325472 | NSIS.ay | 3 |
| 13 | 16.36 | 1925150 | Virut | 1 |

unique network topology with a certain number of bots that leverage different protocols. Table I summarizes the dataset duration, number of flows and bots, and the type of bot in every subset. CTU-13 labels indicate whether a flow is from/to botnet, background or benign. n. Therefore, known infected hosts are labeled as bots and remaining hosts as benign. We have used k-fold cross validation as our training/evaluation technique. K-fold was done considering 9 datasets (1, 2, 5, 6, 7, 9, 10, 11, 12).Considering all the datasets the ratio to non-bot entries and bot entries in the dataset is largely skewed and unbalanced. The dataset was modified to balance the two classes : Initially percentage of bot entries were just **3.98%** of total entries after modification it was brought to **78.68%** This helps the machine learning algorithms to learn the bot's pattern better and produce better results.

## VI. Results

For evaluating the trained model, we have used the datasets 9 and 10 as listed in TABLE 1 consisting of 20 bots in total (10 Neris and 10 RBot) having in total 4,063,677 flows. First we used **DBScan + Naive Bayes classifier** which took around 7 minutes in total to build the graph, apply normalization and to produce results. Giving an accuracy of **97.40%**. Next, we used the combination **DBScan + Logistic Regression (LR)** on the same testing dataset giving an accuracy **97.47%** and taking an average time of 6 minutes in total. Both which outperforms the results when compared to the case of using only unsupervised learning on the datasets which gave an accuracy of **86.01%**. The average was calculated based on the accuracy mentioned below which was guaged using Kmeans clustering experimented on various cluster sizes and accuracy of using only DBSCAN which was **82.12%**.

| # of clusters | Accuracy (%) |
|---|---|
| 2 | 92.91 |
| 12 | 92.60 |
| 22 | 91.66 |
| 32 | 88.36 |
| 42 | 87.59 |
| 52 | 86.99 |
| 62 | 86.63 |
| 72 | 84.10 |
| 82 | 85.95 |
| 92 | 81.98 |

To get a better inference on the model we shall use k-fold cross validation to evaluate our model. Table below shows the accuracy for Logistic Regression and Naive Bayes using DBSCAN in phase 1 as is tested on the mentioned datasets in the first column.

| Dataset | DBSCAN + LR (%) | DBSCAN + NB (%) |
|---|---|---|
| 1 | 97.48 | 97.47 |
| 2 | 97.49 | 97.48 |
| 5 | 97.48 | 97.46 |
| 6 | 97.47 | 97.42 |
| 7 | 97.49 | 97.47 |
| 9 | 97.43 | 96.78 |
| 10 | 97.40 | 96.66 |
| 11 | 97.47 | 97.45 |
| 12 | 97.41 | 97.38 |

Results using k-fold cross validation

Calculating the average over all the datasets we get **97.46%** using Logistic Regression with DBSCAN and **97.29%** using Naive Bayes with DBSCAN. Therefore, our 2 phase learning technique outperforms the vanilla one phase learning technique by more than **10%**.

## VII. conclusion

The struggle to detect malicious agents in a network has recently converged to ML. High FPs and FNs are detrimental to any intrusion detection system. Network-based approaches exhibit plausible detection rates. When paired with a proper modeling technique, such as graphs, high detection accuracy can be achieved with low FPs. In this paper, we propose a two-phased, graph-based bot detection system that leverages both supervised and unsupervised learning. Looking at the results, we can see DBScan with Naive Bayes and LR produce very similar results giving an accuracy of 97.4% and Kmeans on the other hand not producing poor results compared to when clustered using DBScan.

## References


[1] “”BotGM: Unsupervised Graph Mining to Detect Botnets in Traffic Flows” - Sofiane Lagraa , Jerome Francois, Abdelkader Lahmadi , Marine Miner,, Christian Hammerschmidt and Radu State, 2017”. In: ().

[2] “A. Karasaridis, B. Rexroad, D. A. Hoeflin et al., “Wide-scale botnet detection and characterization.” HotBots, vol. 7, pp. 7–7, 2007.” In: ().

[3] “Abbas Abou Daya, Mohammad A. Salahuddin, Noura Limam, and Raouf Boutaba. ”A Graph-Based Machine Learning Approach for Bot Detection”, 2019”. In: ().

[4] “B. Venkatesh, S. H. Choudhury, S. Nagaraja, and N. Balakrishnan, “Botspot: fast graph based identification of structured p2p bots,” Journal of Computer Virology and Hacking Techniques, vol. 11, no. 4, pp. 247– 261, 2015.” In: ().

[5] “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, pp. 2–16, 2013.” In: ().

[6] “E. M. Hutchins, M. J. Cloppert, and R. M. Amin, “Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains,” Leading Issues in Information Warfare Security Research, vol. 1, no. 1, p. 80, 2011.” In: ().

[7] “G. Creech and J. Hu, “A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns,” IEEE Transactions on Computers, vol. 63, no. 4, pp. 807–819, 2014.” In: ().

[8] “G. Gu, R. Perdisci, J. Zhang, and W. Lee, “Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection,” in Proceedings of USENIX Security, 2008, pp. 139–154.” In: ().

[9] “H. Debar, M. Dacier, and A. Wespi, “Towards a taxonomy of intrusiondetection systems,” Computer Networks, vol. 31, no. 8, pp. 805–822, 1999.” In: ().

[10] “J. M. Ehrenfeld, “WannaCry, Cybersecurity and Health Information Technology: A Time to Act,” Journal of Medical Systems, vol. 41, no. 7, p. 104, 2017.” In: ().

[11] “J. R. Binkley and S. Singh, “An algorithm for anomaly-based botnet detection.” SRUTI, vol. 6, pp. 7–7, 2006.” In: ().

[12] “M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K.

Thomas, and Y. Zhou, “Understanding the Mirai Botnet,” in Proceedings of USENIX Security Symposium, 2017, pp. 1093–1110.” In: ().

[13] “O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell: A neural image caption generator,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 3156–3164.” In: ().

[14] “R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, “A comprehensive survey on machine learning for networking: evolution, applications and research opportunities,” Journal of Internet Services and Applications, vol. 9, no. 1, pp. 1–99, 2018.” In: ().

[15] “S. Chowdhury, M. Khanzadeh, R. Akula, F. Zhang, S. Zhang, H. Medal, M. Marufuzzaman, and L. Bian, “Botnet detection using graph-based feature clustering,” Journal of Big Data, vol. 4, no. 1, p. 14, 2017.” In: ().

[16] “S. Garcia, M. Grill, J. Stiborek, and A. Zunino, “An empirical comparison of botnet detection methods,” Computers Security, vol. 45, pp. 100–123, 2014”. In: ().

[17] “S. Khattak, N. R. Ramay, K. R. Khan, A. A. Syed, and S. A. Khayam, “A taxonomy of botnet behavior, detection, and defense,” IEEE Communications Surveys Tutorials, vol. 16, no. 2, pp. 898–924, 2014.” In: ().

[18] “S. Lagraa, J. Franc¸ois, A. Lahmadi, M. Miner, C. Hammerschmidt, and R. State, “Botgm: Unsupervised graph mining to detect botnets in traffic flows,” in IEEE Cyber Security in Networking Conference (CSNet), 2017, pp. 1–8.” In: ().

[19] “S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, “Detecting P2P botnets through network behavior analysis and machine learning,” in Privacy, Security and Trust (PST), 2011”. In: ().