A HYBRID UNSUPERVISED AND DECISION TREE APPROACH FOR BOT DETECTION IN NETWORKS

MADHURI MAMILLAPALLI[1], DR. A. RAMASWAMI REDDY[2]

PG SCHOLAR[1], PRINCIPAL & DIRECTOR[2]

MALLA REDDY ENGINEERING COLLEGE, HYDERABAD

ABSTRACT Botnet-assisted attacks remain a persistent and damaging threat to organizations, leading to billions of dollars in financial loss and long-term reputational damage. Botnets consist of compromised devices (bots) controlled remotely by a botmaster through command-and-control (C2) channels. These bots are frequently used for malicious activities such as Distributed Denial-of-Service (DDoS) attacks, largescale spamming, and identity theft. While machine learning (ML)-based detection using network flow features has been widely explored, these approaches often suffer from high computational overhead and fail to capture the full scope of communication patterns between hosts. To address these challenges, this research introduces a hybrid bot detection framework that integrates unsupervised learning, graph-based feature extraction, and supervised classification. Initially, K-means clustering is applied to partition the dataset into clusters representing benign and bot-related traffic, based on request frequency. Clusters with high request volumes are labeled as potential bot activity.Next, a graph is constructed where each IP address is treated as a node, and communication links form the edges. Edge weights are computed using metrics such as betweenness centrality, alpha centrality, and directional edge weights. From this, features including in-degree, out-degree, clustering coefficient, and centrality measures are extracted. These graph-based features are normalized and used to train a Decision Tree Classifier. The resulting model can accurately classify future network traffic as benign or malicious, offering a more scalable and effective solution to bot detection. This integrated approach leverages structural network insights and behavioral patterns to improve detection accuracy and reduce false positives in dynamic network environments.

Keywords: Bot detection, Graph-based analysis, Decision tree classifier, Unsupervised learning, Network security.

1.INTRODUCTION

Undoubtedly, organizations are constantly under security threats, which not only costs billions of dollars in damage and recovery, it often also detrimentally affects 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. Approximately 500 million computers are infected each year, translating into 18 victims per second.” The most infamous attack, called Rustock, infected 1 million machines, sending up to 30 billion spam emails a day [1]. Hence, it is imperative to defend against these botnet-assisted attacks. A botnet is a collection of bots, agents in compromised hosts, controlled by botmasters via command and control (C2) 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 identify theft. While bots thrive for different sinister purposes, they exhibit a similar behavioral pattern when studied up-close. The intrusion kill-chain [2] dictates the general phases a malicious agent goes through in-order to reach and infest its target. Anomaly-based methods are widely used in both detection [3], [4]. They first establish a baseline of normal behavior for the protected system and model a decision engine. The decision engine determines and alerts any divergence or statistical deviations from the norm as a threat. Machine learning (ML) [3] is an ideal technique to automatically capture the normal behavior of a system. The use of ML has boosted the scalability and accuracy of anomaly-based IDSs [4]. The most widely employed learning paradigms in ML include supervised and unsupervised. Supervised learning uses labeled training datasets to create models. It is employed to learn and identify patterns in the known training data. However, labeling is non-trivial and typically require domain experts to manually label the datasets [3]. This can be cumbersome and prone to error, even for small datasets. On the other hand, unsupervised learning uses unlabelled training datasets to create models that can discriminate between patterns in the data

2.LITERATURE SURVEY 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 capture the topological structure of the communication graph, which can expose additional aspects of malicious hosts. In addition, flowlevel models can incur a high computational overhead, and can also be evaded by tweaking behavioral characteristics e.g., by changing packet structure [5]. Graph-based features, derived from flow-level information, which reflect the true structure of communications, interactions, and behavior 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 major contributions of this paper are as follows: We propose BotChase, an anomaly-based bot detection system that is protocol agnostic, robust to zeroday attacks, and suitable for large datasets. We employ a two-phased ML approach that leverages both supervised and unsupervised learning. The first phase filters presumable benign hosts. This is followed by the second phase on the pruned hosts, to achieve bot detection with high precision. We propose feature normalization (F-Norm) on top of graph-based features in BotChase and evaluate various ML techniques. Our graph-based features, inspired from the literature and derived from network flows, undergo F-Norm to overcome severe topological effects. These effects can skew bot behavior in different networks, exacerbating ML prediction. Furthermore, these features allow to combine data from different networks and promote spatial stability [6] in the ML models. We compare the performance of our graph-based features with flow-based features from BotMiner [7] and BClus [8] in a prototype implementation of BotChase. Furthermore, we compare BotChase with BotGM [9] and the end-to-end system proposed for BClus. We evaluate the BotChase prototype system in an online setting that recurrently trains and tests the ML models with new data. We also leverage the Hoeffding Adaptive Tree (HAT) [10] classifier for incremental learning. This is crucial to account for changes in network traffic and host behavior. Botnet detection has been an active area of research that has generated a substantial body of work. Common botnet detection approaches are passive. They assume successful intrusions and focus on identifying infected hosts (bots) or detecting C2 communications, by analyzing system logs and network data, using signature- or anomaly-based techniques. Signature-based techniques have commonly been used to detect pre-computed hashes of existing malware in hosts and/or network traffic. They are also used to isolate IRC-based bots by detecting bot-like IRC nicknames [12], and to identify C2-related DNS requests by detecting C2-like domain names [13]. Metadata such as regular expressions based on packet content and target IP occurrence tuples [14] is an example of what could be employed in a signature and pattern detection algorithm. More generally, signature-based techniques have been employed to identify C2 by comparison with known C2 communication patterns extracted from observed C2 traffic, and infected hosts by comparison with static profiles and behaviours of known bots [15]. However, they can be easily subverted by unknown or modified threats, such as zero-day attacks and polymorphism [15], [16]. This undermines their suitability to detect sophisticated modern botnets.

3.PROPOSED METHODOLOGY The CTU-13 is a dataset of botnet traffic that was captured in the CTU University, Czech Republic, in 2011. The goal of the dataset was to have a large capture of real botnet traffic mixed with normal traffic and background traffic. The CTU-13 dataset consists in thirteen captures (called scenarios) of different botnet samples. On each scenario we executed a specific malware, which used several protocols and performed different actions. Each scenario was captured in a pcap file that contains all the packets of the three types of traffic. These pcap files were processed to obtain other type of information, such as NetFlows, WebLogs, etc.

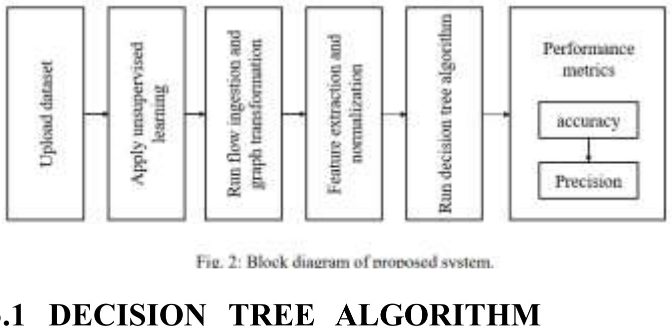

Upload dataset
Apply unsupervised learning
Run flow ingestion and graph transformation
Feature extraction and normalization
Run decision tree algorithm
Performance metrics
accuracy
Precision

Fig. 2: Block diagram of proposed system.

3.1 DECISION TREE ALGORITHM Decision Tree is a supervised learning technique widely used for classification problems, although it can also be applied to regression tasks. It functions as a tree-structured classifier where internal nodes represent the features of a dataset, branches denote decision rules, and each leaf node signifies an outcome. A decision tree comprises two types of nodes: decision nodes, which make decisions and have multiple branches, and leaf nodes, which represent the final output and contain no further branches. The decision-making process is based on the features of the dataset and is structured to explore all possible solutions to a problem under given conditions. The model starts from a root node and expands into branches, forming a tree-like structure. Typically, the tree is constructed using the CART (Classification and Regression Tree) algorithm, where each split is based on answers to feature-specific questions. This process continues until the tree reaches its leaf nodes, which deliver the final classification or prediction. Decision Tree learning is known for its simplicity, interpretability, and efficiency. It approximates discrete-valued functions, including boolean functions, and expresses learned functions as decision trees or if-then-else rules. With a highly expressive hypothesis space, the method accommodates disjunctions of conjunctions based on attribute constraints. During classification, an instance traverses the tree from the root node through internal nodes—each representing an attribute test—until it reaches a leaf node that provides the final label. The approach is robust to noisy data and capable of handling both numerical and categorical features. It requires minimal data preparation, supports multi-output problems, and operates with a cost that scales logarithmically with the training data size, making it suitable for a wide range of real-world applications.

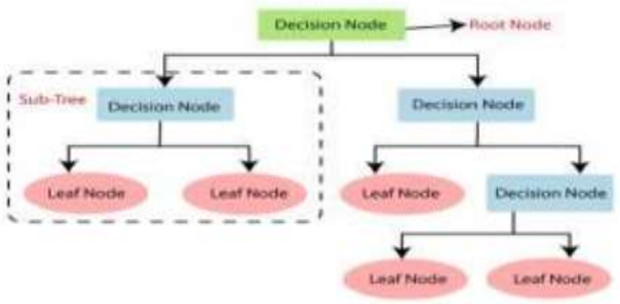
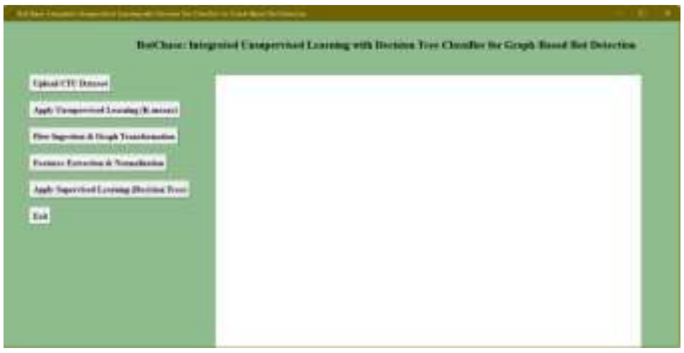


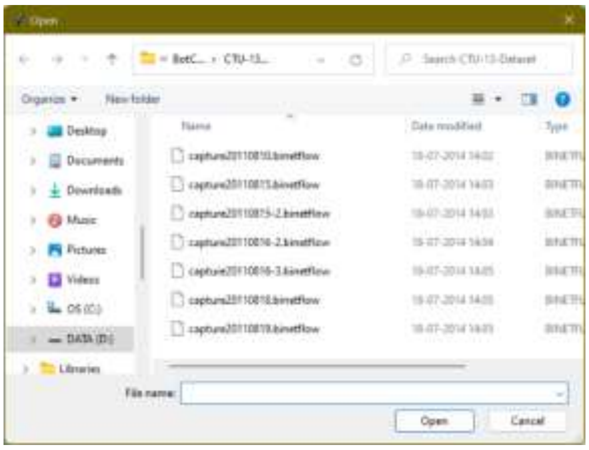
Fig. 3: Decision tree algorithm.

RESULTS

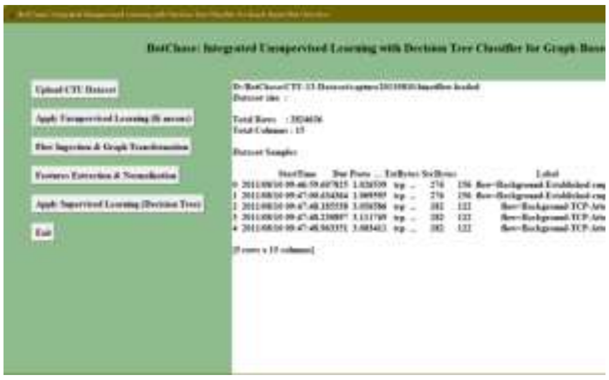
To run project double click on 'run.bat' file to get below screen


BotChase: Integrated Unsupervised Learning with Decision Tree Classifier for Graph Based Bot Detection
Upload CTU Dataset
Apply Unsupervised Learning (K-means)
Flow Ingestion & Graph Transformation
Features Extraction & Normalization
Apply Supervised Learning (Decision Tree)
Exit

In above screen click on 'Upload CTU Dataset' button and upload dataset

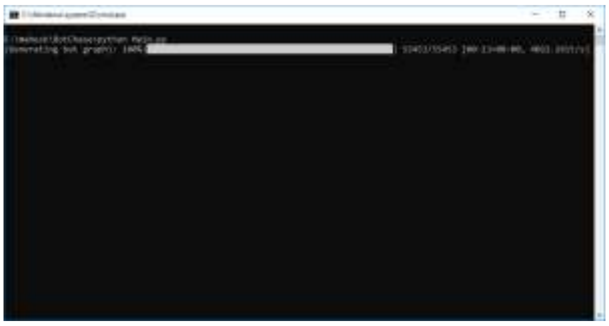

Open
BotC... › CTU-13...
Search CTU-13-Dataset
Organize
New folder
Desktop
Documents
Downloads
Music
Pictures
Videos
OS (C:)
DATA (D:)
Libraries
Name
Date modified
Type
capture20110810.binetflow
capture20110811.binetflow
capture20110815-2.binetflow
capture20110816-2.binetflow
capture20110816-3.binetflow
capture20110818.binetflow
capture20110819.binetflow
File name:
Open
Cancel

In above screen uploading first capture file and now click on 'Open' button to upload and to get below screen


BotChase: Integrated Unsupervised Learning with Decision Tree Classifier for Graph Base
Upload CTU Dataset
Apply Unsupervised Learning (K-means)
Flow Ingestion & Graph Transformation
Features Extraction & Normalization
Apply Supervised Learning (Decision Tree)
Exit

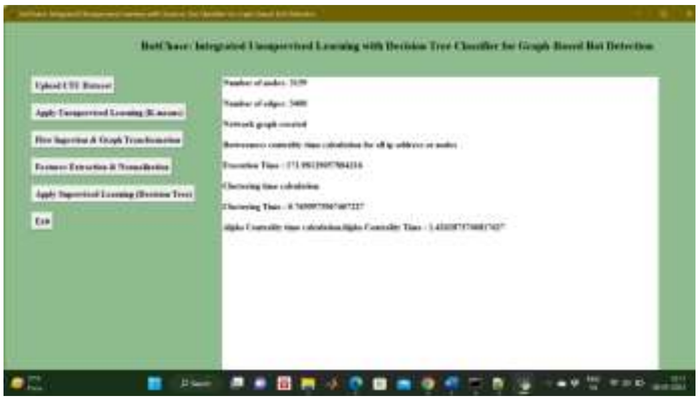
```
D:/BotChase/CTU-13-Dataset/capture20110810.binetflow loaded
Dataset size :

Total Rows    : 2824636
Total Columns : 15

Dataset Sample:

          StartTime    Dur Proto ... TotBytes SrcBytes                       Label
0 2011/08/10 09:46:59.607825 1.026539  tcp ...   276   156 flow=Background-Established-cmp
1 2011/08/10 09:47:00.434004 1.009595  tcp ...   276   156 flow=Background-Established-cmp
2 2011/08/10 09:47:48.185538 3.056586  tcp ...   182   122      flow=Background-TCP-Atte
3 2011/08/10 09:47:48.230897 3.111769  tcp ...   182   122      flow=Background-TCP-Atte
4 2011/08/10 09:47:48.963371 3.083411  tcp ...   182   122      flow=Background-TCP-Atte

[5 rows x 15 columns]
```

In above screen dataset contains total 2824636 records and each record contain 15 columns and below it I am displaying some dataset records. Now click on 'Apply Supervised Learning (K-means) to separate Bot & Benign Data' button to remove benign records In above screen we can see dataset size before removing benign records and after removing benign records. By removing some benign records we can reduce dataset size. Now click on 'Run Flow Ingestion & Graph Transformation' button to generate graph



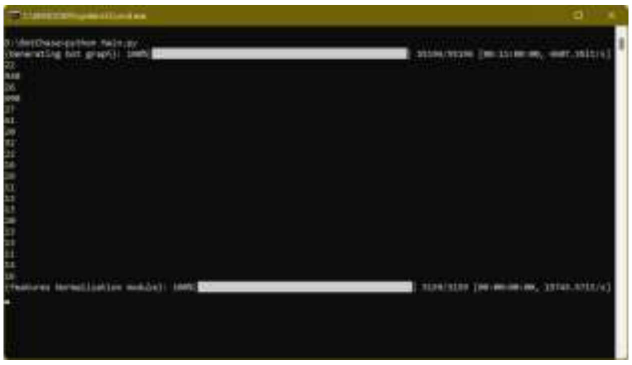
```
[illegible]\BotChase>python Main.py
Generating bot graph): 100%|████████████████████| 55453/55453 [00:13<00:00, 4003.2015it/s]
```

In above screen we can see progress bar which indicates graph-based features extraction and while applying this technique it will open 2 empty windows and you just close those 2 empty windows to get below screen

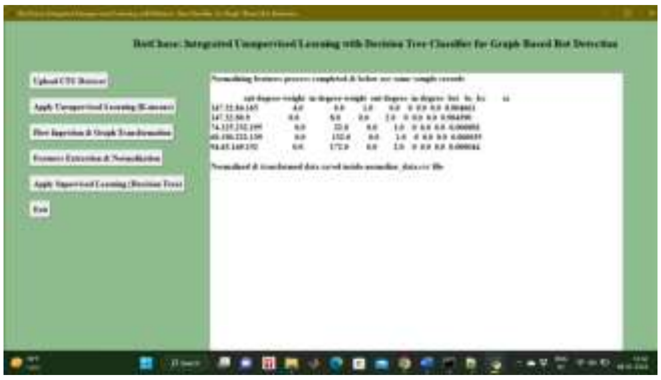

BotChase: Integrated Unsupervised Learning with Decision Tree Classifier for Graph Based Bot Detection
Upload CTU Dataset
Apply Unsupervised Learning (K-means)
Flow Ingestion & Graph Transformation
Features Extraction & Normalization
Apply Supervised Learning (Decision Tree)
Exit

```
Number of nodes: 3139
Number of edges: 3400
Network graph created
Betweenness centrality time calculation for all ip address or nodes
Execution Time : 171.9[illegible]
Clustering time calculation
Clustering Time : 0.7659[illegible]
alpha Centrality time calculation Alpha Centrality Time : 1.43[illegible]
```

In above screen we can see total nodes and edges generated and time taken to calculate between_ness, alpha centrality, and clustering. Now we generate graphs and now click on 'Features Extraction & Normalization' button to extract features and to perform normalization on extracted features.

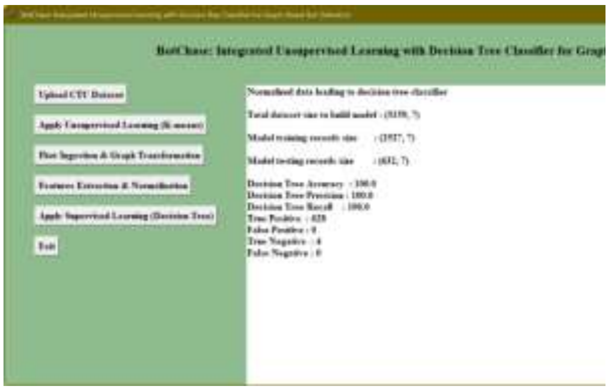


```
D:\BotChase>python Main.py
Generating bot graph): 100%|████████████████████| 35196/35196 [00:11<00:00, [illegible]it/s]
22
[illegible]
26
[illegible]
27
[illegible]
[illegible]
Features Normalization module): 100%|████████████████████| 3139/3139 [00:00<00:00, 15743.5715it/s]
```


BotChase: Integrated Unsupervised Learning with Decision Tree Classifier for Graph Based Bot Detection
Upload CTU Dataset
Apply Unsupervised Learning (K-means)
Flow Ingestion & Graph Transformation
Features Extraction & Normalization
Apply Supervised Learning (Decision Tree)
Exit

```
Normalizing features process completed & below are some sample records

              out-degree-weight  in-degree-weight  out-degree  in-degree  bet  lcc  lcc      ai
147.32.84.165          4.0           0.0        1.0       0.0   0  0.0  0.0  0.004461
147.32.80.9            0.0           0.0        0.0       1.0   0  0.0  0.0  0.984490
74.125.232.195         0.0          22.0        0.0       1.0   0  0.0  0.0  0.000051
65.196.222.139         0.0         152.0        0.0       1.0   0  0.0  0.0  0.000059
94.65.169.252          0.0         172.0        0.0       2.0   0  0.0  0.0  0.000044

Normalized & transformed data saved inside normalize_data.csv file
```

In above screen after normalization, I am displaying few records with out_degree, in degree and weight details. In above screen 'bc' refers to between_ness and 'lcc' refers to clustering and 'ac' refers to alpha centrality. All normalized records are saved inside 'normalize_data.csv' file and you can open and see that file from code folder. Now click on 'Run Decision Tree Algorithm' button to generate training model with decision tree classifier and to calculate metrics such as accuracy, precision etc.


BotChase: Integrated Unsupervised Learning with Decision Tree Classifier for Grap
Upload CTU Dataset
Apply Unsupervised Learning (K-means)
Flow Ingestion & Graph Transformation
Features Extraction & Normalization
Apply Supervised Learning (Decision Tree)
Exit

```
Normalized data leading to decision tree classifier

Total dataset size to build model : (3159, 7)

Model training records size     : (2527, 7)

Model testing records size      : (632, 7)

Decision Tree Accuracy  : 100.0
Decision Tree Precision : 100.0
Decision Tree Recall    : 100.0
True Positive  : 628
False Positive : 0
True Negative  : 4
False Negative : 0
```

In above screen after normalization, we got total records as 3159 with 7 columns (in_degree, out_degree, weight etc.) and application split total records into train size as 2527 and test size as 632. After building train model we apply test records and got accuracy as 100%. Below screen showing normalization progressing steps

CONCLUSION In this research, we proposed BotChase, a system capable of efficiently transforming network flows into an aggregated graph model for enhanced bot detection. The system leverages a two-phase machine learning approach to differentiate bots from benign hosts. The second phase, which utilizes a Decision Tree (DT) classifier, demonstrates high true positive rates and low false positive rates, indicating reliable performance. BotChase is also effective in detecting bots that operate using various protocols and proves robust against unknown attacks, as well as in cross-network machine learning model training and inference. Our results show that flow-based features underperform compared to graph-based features, which provide better representation of communication patterns. BotChase outperforms traditional end-to-end systems that rely solely on flow-based features and performs competitively against existing graph-based systems such as BotGM. In an online setting, BotChase integrates the Hoeffding Adaptive Tree (HAT) algorithm for incremental learning, enabling real-time processing. Although the model takes longer to converge, it ultimately delivers superior classification performance. For implementation, we utilized the CTU-13 dataset, which includes 13 scenarios. Each scenario provides PCAP files (containing raw network traffic) and corresponding capture files (containing extracted data such as source address, destination address, timestamp, and packet size). To construct the graph, we used the capture files instead of the PCAP files. Selected scenarios were downloaded and stored in the 'CTU-13-dataset' folder for analysis.

REFERENCES

[1] J. Caballero, C. Grier, C. Kreibich, and V. Paxson, "Measuring payper-install: the

commoditization of malware distribution," in USENIX Security, 2011, p. 13.

[2] 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," Information Warfare & Security Research, vol. 1, no. 1, p. 80, 2011.

[3] R. Boutaba et al., "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.

[4] G. Creech and J. Hu, "A Semantic Approach to Host-Based Intrusion Detection Systems Using Contiguous and Discontiguous System Call Patterns," IEEE Trans. on Computers, vol. 63, no. 4, pp. 807–819, 2014.

[5] 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.

[6] Y. Jin et al., "A modular machine learning system for flow-level traffic classification in large networks," ACM TKDD, vol. 6, no. 1, p. 4, 2012.

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

[8] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," Computers & Security, vol. 45, pp. 100–123, 2014.

[9] S. Lagraa et al., "BotGM: Unsupervised graph mining to detect botnets in traffic flows," in IEEE CSNet, 2017, pp. 1–8.

[10] A. Bifet and R. Gavalda, "Adaptive learning from evolving data streams," ` in Intl. Sym. on Intelligent Data Analysis, 2009, pp. 249–260.