A Two-phase Multi-class Botnet Labeling Approach for Real-world Traffic

Ta-Chun Lo
Department of Electrical Engineering
National Cheng Kung University
Tainan, Taiwan
N28091108@gs.ncku.edu.tw

Shan-Hong Yang
Institute of Computer and Communication Engineering
National Cheng Kung University
Tainan, Taiwan
Q36101325@gs.ncku.edu.tw

Jyh-Biau Chang
Department of Electronic Engineering
Lunghwa University of Science and Technology
Taoyuan, Taiwan
andrew@gm.lhu.edu.tw

Chung-Ho Chen
Institute of Computer and Communication Engineering
National Cheng Kung University
Tainan, Taiwan
chchen@mail.ncku.edu.tw

Ce-Kuen Shieh
Department of Electrical Engineering
National Cheng Kung University
Tainan, Taiwan
shieh@ee.ncku.edu.tw

Abstract—**Within the realm of cybersecurity, botnets represent an increasingly formidable threat, characterized by diverse types exhibiting distinct behavioral patterns and characteristics. This study addresses the imperative need for real-time botnet activity detection by introducing a multi-class labeling system tailored for real-world network traffic. Employing clustering algorithms and a semi-supervised learning framework, this system efficiently labels benign traffic and performs multi-class labeling for various botnet traffic categories. Hierarchical Density-based Spatial Clustering of Applications with Noise (HDBSCAN) is harnessed for clustering both synthetic and real-world datasets, significantly enhancing labeling coverage. The remaining traffic is designated as "unknown" and subjected to identification through a semi-supervised learning approach. A comparative analysis underscores the superiority of HDBSCAN over Density-based Spatial Clustering of Applications with Noise (DBSCAN), successfully clustering an additional 11% of data. Remarkably, our system exhibits substantial advancements in data labeling when juxtaposed with prior research efforts. This research introduces an effective solution for botnet labeling in the context of network security, thereby enhancing the capacity for detecting and mitigating malicious botnet activities.**

Keywords—Botnet Classification, Clustering algorithm, Data labeling, Real-world traffic labeling, Self-learning

I. INTRODUCTION

In the field of cybersecurity, botnets pose an escalating threat as they house malicious network traffic generated by compromised hosts, manipulated by hackers or malware. These botnets serve as platforms for large-scale malicious operations. Various botnet types exhibit unique behavioral patterns. Waledac prompts infected hosts to distribute malicious emails with harmful links or attachments, leading to further infections. On the other hand, TrickBot operates discreetly, clandestinely harvesting login credentials during website access. Given the substantial behavioral diversity among botnets, it's imperative to categorize and distinguish them within real-world traffic effectively.

In previous studies, researchers have utilized clustering algorithms to label real-world traffic. For instance, in [1], clustering algorithms were employed to label the traffic associated with attack behaviors. However, the classification was limited to benign and malicious categories, overlooking the variations within malicious behaviors and lacking consideration for multi-class cases. Although some studies attempted multi-class analysis of real-world network traffic using clustering algorithms, these methods focused only on specific types of botnet infections, and the labeled data accounted for only a small portion of the testing dataset. For example, in the study by [2], DBSCAN clustering algorithm was used to perform multi-class analysis of IoT-based botnets on synthetic datasets and real-world network traffic, but it achieved accurate labeling for only 10% of the data, with most of the traffic being labeled as malicious without explicit class labels. In our team's related research [3], we utilized DBSCAN clustering on both real-world traffic and synthetic datasets to label P2P botnet traffic, achieving accurate labeling for 30% of the botnet traffic. The performance of clustering algorithms plays a crucial role in determining the amount of accurately labeled data in multi-class botnet classification. In previous studies [2, 3], the DBSCAN clustering algorithm used could result in clusters containing multiple types of botnet infections, leading the researchers to know that the traffic within these clusters is malicious but unsure which category of botnet it belongs to, resulting in labeling them as malicious without specifying the exact class.

This study aims to provide a multi-class labeling system for real-work traffic, capable of labeling various type of botnet traffic as well as benign traffic. We propose a two-phase botnet labeling system that combines clustering algorithms and semi-supervised learning, yielding good results for multi-class labeling in real-world cases. In the first phase, we employ a more suitable clustering algorithm HDBSCAN to cluster the synthetic datasets and real-world data, identifying traffic similar to botnet activity in the real-world traffic and improving the labeling coverage. The remaining real-world traffic is labeled as unknown and further identified in the second phase. In the second phase, we utilize semi-supervised learning methods to train a multi-class model, reducing the amount of unknown data.

We compared the performance of two clustering algorithms, HDBSCAN and DBSCAN, and demonstrated that HDBSCAN outperforms DBSCAN in clustering by accurately clustering an additional 11% of the data. Compared to our team's previous research, our system increases the ratio of labeled data from 30% to 77% for real-world traffic.

II. BACKGROUND AND RELATED WORKS

A. HDBSCAN & DBSCAN

DBSCAN assumes that the data distribution has the same density throughout, which is not applicable in real-world network conditions. Therefore, we need to adjust the clustering results according to user expectations. Figure 1 simply illustrates the difference between the two algorithms.

DBSCAN clusters instances into a group with predefined distance value, while HDBSCAN considers the density of data in their vicinity. If the density is low, it expands the distance between data, while it has minimal effect when high density.

After careful evaluation, we opted for the HDBSCAN clustering algorithm to handle our data. HDBSCAN stands out as a robust clustering algorithm tailored for diverse data characteristics, encompassing arbitrary shapes, variable sizes and densities, and noisy datasets. It adeptly detects and analyzes a spectrum of clustering structures, even those that are non-convex and non-linear, all while effectively managing data noise. An open-source performance assessment[27] comparing HDBSCAN to other widely used clustering algorithms can be found in the HDBSCAN section of scikit-learn-contrib[28]. Figure 2 visually illustrates the outcomes of this comparison, highlighting HDBSCAN's exceptional performance.

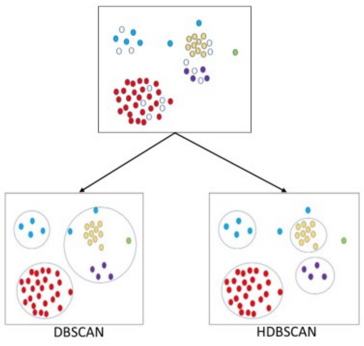

Fig. 1. Data distribution

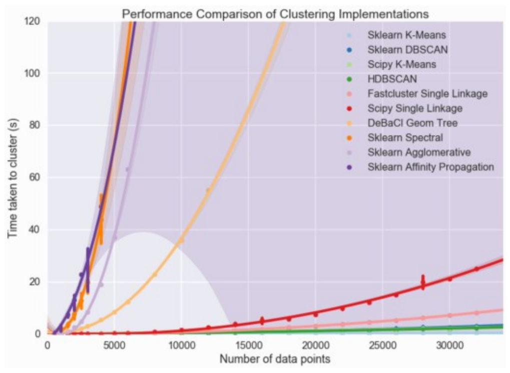

Fig. 2. Performance of clustering algorithms [28]

B. Self-training

Both DBSCAN and HDBSCAN have instances of real-world traffic that remain un-clustered. Furthermore, a portion of real-world traffic lacks a definite classification due to the presence of multiple synthetic data categories within clusters. We employ a self-learning approach to augment the amount of data we can label. Self-training [7] is a semi-supervised learning approach used to tackle problems with a small amount of labeled data and a large amount of unlabeled data. The goal of this method is to train a model using known labeled data and then use that model to predict and label the unlabeled data. This can be particularly useful in situations where obtaining a large amount of accurate labeled data is difficult or costly. Through augmented, thereby enhancing the model's performance.

Self-training typically comprises these steps: First, an initial model is trained with labeled data. This model is then applied to unlabeled data, predicting their labels and thus generating predicted labels. These predicted labels are subsequently incorporated into the labeled dataset, expanding it. This augmented dataset is used to train an updated model. This iterative process continues until the desired performance or label quantity is attained. Self-training's advantage lies in its capacity to harness the valuable information within unlabeled data, effectively increasing the labeled dataset's coverage. This proves especially valuable in scenarios where obtaining ample, accurate labeled data is challenging or costly. Self-training enables us to maximize the utility of the existing limited labeled data, enhancing the model's generalization and performance.

C. Related works

Numerous studies have concentrated on the detection of botnets within specific domains, often employing synthetic datasets for training and testing. While these studies have achieved favorable results in terms of accuracy and correctness, the applicability of these achievements to real-world traffic labeling warrants investigation. Central to this issue is a pivotal question: whether the scenarios simulated by synthetic datasets can comprehensively encompass the intricate and ever-changing network traffic characteristics of the real world. For instance, within these research endeavors, we observe certain publications [31-35] opting to employ the synthetic dataset of CTU-13 as a benchmark for testing. Although these methods have yielded satisfactory outcomes on synthetic datasets, it does not imply their seamless adaptation to the complex diversity of real-world network traffic. The design of synthetic dataset might not fully capture the myriad possible scenarios present in the real world, which often comes with unforeseeable variations and challenges.

Some studies have focused on detecting real-world network traffic, successfully identifying a portion but leaving the rest unlabelled as potential threats. For example, Z. Liu et al. [1] developed a method to predict botnet attacks in real-world traffic based on Command and Control (C2C) activities. They used ISP flow data to calculate C2C indicators and trained a Long Short-Term Memory (LSTM) model, achieving a 0.767 prediction accuracy for binary classification. However, this approach did not address the multi-class nature of modern botnet attacks, which is crucial given their increasing diversity.

Trajanovski et al. [2] proposed an automated behavior-based clustering method for Internet of Things (IoT) botnets, aiming to identify botnets with new functionalities. The method captures the behavior of IoT botnet samples in a simulated environment, represents it as behavioral profiles, and vectorizes them. Then, DBSCAN algorithm is applied to cluster these vectors, enabling automated clustering analysis. They constructed a test dataset using IoT botnet samples propagated from the Internet and evaluated the method. The results showed that the method can accurately identify IoT botnets with new functionalities, but it could only identify 10% of the data, while the remaining was classified as malicious traffic.

In our team's previous research [3], Chen, Wei-Yu et al. employed BotCluster[4] to perform clustering on both real-world data and synthetic datasets. Chen transformed the real-world data into sessions and used BotCluster in conjunction with the synthetic datasets, after removing certain whitelisted data. After clustering, if a cluster contained only one category of synthetic data, we labeled all the real-world data in that cluster as the corresponding category. For other cases, we labeled the real-world data as unknown malicious data. We also introduced a specific category called the "malicious group" to represent traffic that was considered botnet malware in previous studies but lacked clear classification.

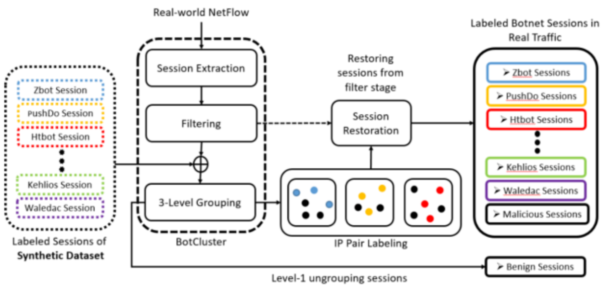

Fig. 3. [3] labeling method

III. METHODOLOGY

A. *Workflow*

Figure 4 outlines our Two-phase Data Labeling approach, which aims to label benign and multi-class botnet traffic. We recognize that directly applying a model trained on synthetic data to real-world traffic may lead to misjudgments due to domain differences. To address this, we cluster real-world and synthetic data to enable the model to learn real-world traffic features. This clustering also enhances our labeled dataset for effective self-training.

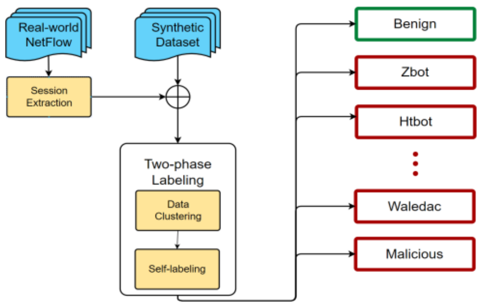

Fig. 4. System workflow

Additionally, referring to [3], we improve the representation of IP address communication by using "Session Extraction" on both real-world and synthetic datasets, transforming them into sessions. Our Two-phase Data Labeling system comprises two phases: Data Clustering for labeling real-world traffic resembling synthetic data, and Self-Labeling for enhancing labels on un-clustered real-world traffic and traffic within clusters.

B. *Phase 1 Data Clustering*

In the first phase, we use HDBSCAN to cluster the synthetic dataset and real-world traffic, resulting in five scenarios. The first combines real-world traffic with multiple categories of synthetic data, the second contains synthetic data only, possibly encompassing one or multiple categories. The third scenario consists solely of real-world traffic, and the fourth merges real-world traffic with synthetic data of the same category. The fifth scenario includes un-clustered noise data. We label real-world traffic within the fourth scenario using synthetic data categories. In Figure 6, the left side showcases clustering results, where colored dots represent various botnet types and white dots denote real-world sessions. We then label real-world traffic within the fourth scenario, as depicted on the right side of Figure 6. In the second phase, we integrate data from the first, third, and fifth scenarios into the unlabeled dataset, while the remaining data goes into the labeled dataset. A higher proportion of the fourth scenario enhances our ability to label genuine network traffic.

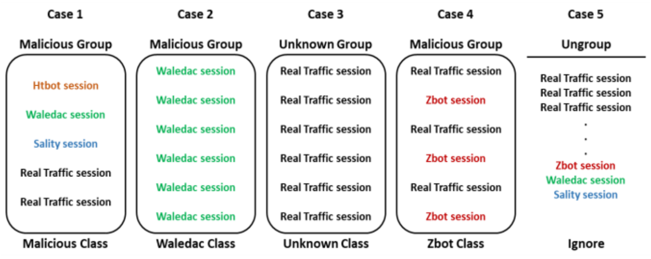

Fig. 5. Cluster situation

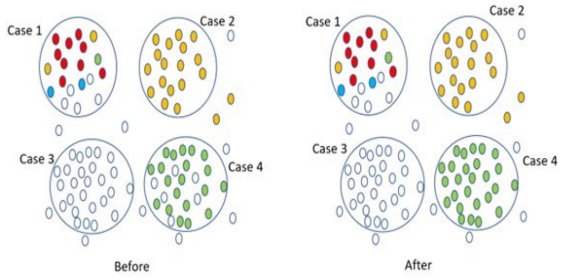

Fig. 6. Phase 1 labeled example

Figure 7 illustrates the procedural flow of the data clustering phase. Following the clustering of both real-world traffic and synthetic data, our initial step entails the determination of data labeling status. If data has already been labeled, this designation signifies its synthetic origin; conversely, unlabeled data is associated with real-world traffic. In cases involving synthetic data, we incorporate it into the labeled dataset. When dealing with real-world traffic, our initial assessment involves verifying whether all synthetic data instances within a given cluster correspond to the same category, thereby constituting Case 4. In such instances, we assign the label of the category as the synthetic data within that cluster to the real-world traffic. Subsequently, this labeled real-world traffic is included in the labeled dataset. Conversely, if synthetic data within a cluster spans multiple categories or no synthetic data is present, the real-world traffic remains unlabeled within this phase.

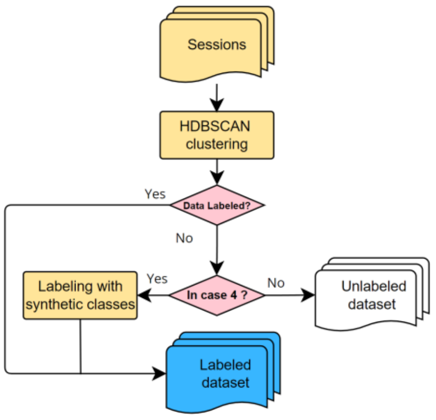

Fig. 7. Phase 1 Data Clustering workflow

C. *Phase 2 Self-labeling*

Figure 8 provides a detailed depiction of the self-labeling process. In this phase, we employ the previously labeled dataset to train a model. Subsequently, we utilize this model to predict the unlabeled dataset. If the model exhibits confident predictions for the unlabeled data, indicating a prediction probability surpassing 90% for a certain category, it signifies a high level of confidence in the classification. We include this data in the high-confidence dataset and assign it the predicted label. Alternatively, if the model's prediction lacks confidence, we return the data to the unlabeled dataset.

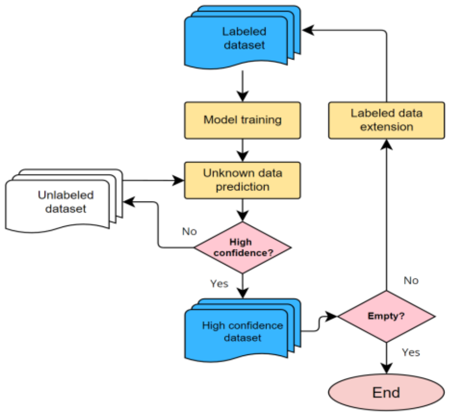

Fig. 8. Phase 2 Self-labeling workflow

After evaluating all the data, we verify whether the high-confidence dataset is empty. If it contains data, we amalgamate its contents into the labeled dataset, retrain the model, and then repeat the steps. If the high-confidence dataset remains empty, the self-labeling process concludes. For data that cannot be labeled in either phase, we uniformly classify them as an unknown category.

In our self-training framework, it is applicable if the output of the neural network classifier is a probability distribution for each category. We utilize a fully connected neural network to process sessions, each comprising 20 features. The model includes 6 hidden layers with 4096 neurons each. We apply standard components such as the Relu activation function [12], He normalization [13], Adam optimizer [14], and employ 3 dropout layers [15] with an initial dropout rate of 0.4

IV. Experiments

A. Environment

We finish our experiment in a personal computer. The CPU is Intel(R) Core (TM) i7-9700 CPU @ 3.00GHz, memory size is 32 GB, the storage space is 8 TB, and the GPU is NVIDIA GeForce RTX 2080 SUPER.

B. Dataset

The dataset can divide into two categories, which are the synthetic dataset, and the real-world traffic dataset.

Synthetic Dataset: Nine distinct botnet types of synthetic datasets were assembled for this study. The Storm and Waledac datasets were obtained from the PeerRush [8] dataset, while the remaining benign and seven botnet types were sourced from the Malware Capture Facility Project [9], generously provided by CTU University of Prague in the Czech Republic. Data collection for these datasets occurred over the period spanning 2013 to 2018. Additionally, we leveraged the widely acknowledged CTU-13 dataset, which encompasses 13 unique scenarios, inclusive of seven botnet traffic types and benign traffic. Detailed information regarding these datasets can be found in Table 1 and Table 2. The relatively balanced dataset from Table 1 served as the basis for contrasting HDBSCAN and DBSCAN methodologies, as well as evaluating our approach against the BotCluster multi-class labeling method. Meanwhile, the dataset in Table 2 was employed for comparative analysis with other research contributions.

Real-world Dataset: Table 3 shows the detail of the real-world NetFlow comes from National Cheng Kung University (NCKU) on the TWAREN (Taiwan Advanced Research and Education Network). We used the data from December 2019 as our real-world traffic data.

TABLE I. MCFP & PeerRush Dataset

Class	From	Year	Sessions	Flows
Benign	MCFP	2016-2017	571,278	1,107,246
Storm	PeerRush	2013	57,448	113,972
Wisdomeyes	MCFP	2016-2017	59,639	122,344
Tinba	MCFP	2015-2016	58,393	118,018
Waledac	PeerRush	2013	61,370	121,737
Amy	MCFP	2016	53,629	126,042
TrickBot	MCFP	2017-2018	59,431	119,703
Emotet	MCFP	2015-2017	53,828	76,967
Papras	MCFP	2017-2018	58,388	116,612
Netsky	MCFP	2015-2016	54,984	104,322

TABLE II. CTU-13 Dataset

ID	Packets	NetFlow	Sessions	Bot
1	71,971,482	2,824,637	2,049,540	Neris
2	71,851,300	1,808,123	1,437,934	Neris
3	167,730,395	4,710,639	3,921,631	Robt
4	62,089,135	1,121,077	1,098,708	Rbot
5	4,481,167	129,833	109,403	Virut
6	38,764,357	558,920	498,137	Menti
7	7,467,139	114,078	103,258	Sogou
8	155,207,799	2,954,231	2,555,343	Murlo
9	115,415,321	2,753,885	2,447,096	Neris
10	90,389,782	1,309,792	1,078,324	Rbot
11	6,337,202	107,252	102,344	Rbot
12	13,212,268	325,472	319,633	NSIS.ay
13	50,888,256	1,925,150	1,403,292	Virut

TABLE III. Real-world Dataset

From	Year	Sessions	Flows
NCKU	2019	500,106	1,072,501

C. Experiment 1 Minimum Cluster Size

In the first experiment, we conducted tests on different minimum cluster size of the HDBSCAN clustering algorithm. Figure 9 illustrates the result. The x-axis represents the values of different min cluster size, while the y-axis represents the ratio of labeled data to the time spent on clustering. Our goal is to achieve more labeled data while minimizing the time spent on clustering. Ultimately, we selected the parameter value of 45 as our HDBSCAN parameter setting.

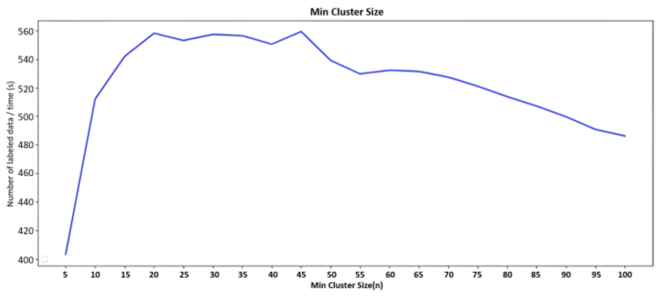

Fig. 9. Minimum cluster size

D. *Experiment 2 HDBSCAN vs DBSCAN*

Figure 10 represents our Experiment 2, where we initially divide the synthetic dataset into three phases: the training dataset, the Imitate real dataset, and the Validation dataset. The training dataset contains labeled data that is known to us, while the Imitate real dataset has the labels removed, simulating our lack of knowledge about the true labels of those data. The testing dataset is used to evaluate the usability of the data labeled by our method.

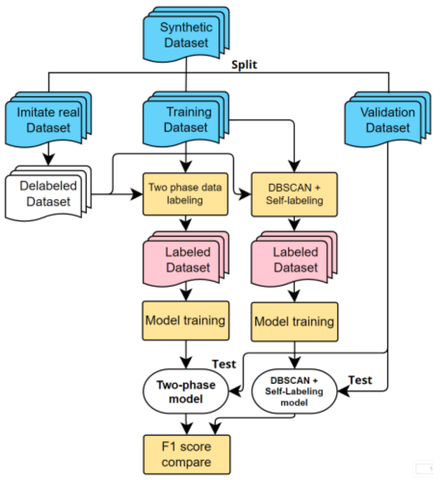

Fig. 10. HDBSCAN vs DBSCAN

We remove the labels from the Imitate real dataset and then conduct Two-phase Data Labeling or DBSCAN + Self-labeling together with the training data to generate Labeled datasets. Once obtain the Labeled datasets, we compare the quantities of Imitated real data that these two methods labeled. Additionally, the Labeled data labeled by both methods will be utilized to train one neural network model. We will employ validation data to compare the differences in F1-score among these models, serving as an indicator to evaluate the usability of our labeled data.

The experimental results (Table 4 and 5) show that HDBSCAN + Self-labeling can label 77% of the Imitate real data, while DBSCAN + Self-labeling can only label 66% of the data. In terms of performance comparison, the two-phase method performs better in each metrics.

TABLE IV. AMOUNT OF LABELING DATA

Model	Labeled imitate data
Two-phase	77%
DBSCAN + Self-labeling	66%

TABLE V. MODELS COMPARISON

Model	Acc	Precision	Recall	F1-score
Two-phase	86%	93%	83%	85%
DBSCAN + Self-labeling	73%	68%	82%	72%

E. *Experiment 3 Two-phase vs previous work*

Figure 11 illustrates the process of Experiment 3. We utilized synthetic datasets and real network traffic, applying both our two-phase data labeling and previous work[3] to label the real network traffic, thus generating a labeled dataset of real traffic. Our evaluation focus lies in comparing how much additional data we can label and the extent of overlap between this data and the labels assigned by our system.

Figure 12 showcases the respective quantities of real network data that they can label. The left circle represents the amount of real network traffic that BotCluster can label, and the right circle represents the quantity labeled in the first phase of the two-phase data labeling. Figure 13 displays the amounts of real network traffic labeled by BotCluster and the complete two-phase data labeling method. In comparison, our method increases the labeled data quantity from 30% to 77% and demonstrates a 25.3% data overlap.

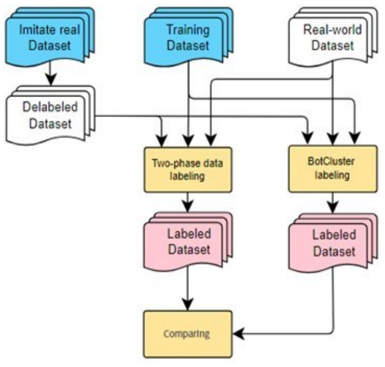

Fig. 11. Two-phase labeling V.S BotCluster multiclass labeling

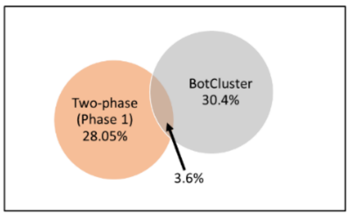

Fig. 12. Amount of labeled real-world data comparison (Phase 1)

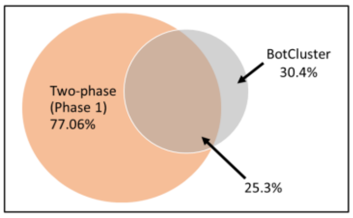

Fig. 13. Amount of labeled real-world data comparison

F. *Experiment 4 Comparison with related work*

Table 6 provides a comprehensive comparison between our methodology and prior research endeavors, as delineated in references [31-35]. To evaluate the efficacy of our approach, we conducted experiments utilizing the synthetic dataset from CTU-13, as outlined in Table 2. The dataset was partitioned into two equal subsets: 50% for training and 50% for testing. Subsequently, we employed a standardized performance matrix to gauge disparities among these methodologies. As discerned in Table 6, our approach exhibits exemplary performance across a spectrum of metrics.

TABLE VI. CONFUSION MATRIXES COMPARE WITH OTHERS WORK

Method	Accuracy	Precision	Recall	F1-score
[31]	0.966	0.833	0.723	0.774
[32]	0.995	0.993	0.994	**0.994**
[33]	0.979	0.727	0.978	N/A
[34]	0.979	0.727	**1.0**	0.83
[35]	0.962	N/A	0.946	N/A
Two-phase	**0.999**	**0.998**	0.989	**0.994**

V. CONCLUSION

Different types of botnets exhibit distinct behavior patterns and characteristics. To achieve real-time detection, it is necessary to categorize these botnets into subgroups and label the malicious and benign traffic within the real-world traffic. We propose a multi-class two-phase botnet labeling system which combines clustering algorithms and semi-supervised learning for real-world traffic that can label real-world botnets traffic, as well as benign traffic. In the first phase, we utilize HDBSCAN to cluster synthetic datasets and real-world data, thereby labeling traffic like botnet behavior and improving labeling coverage. The remaining real-world traffic is labeled as unknown and subjected to identification in the second phase. In this phase, we employ semi-supervised learning techniques to build a multi-class model and reduce the amount of unknown data.

After comparing the effectiveness of HDBSCAN and DBSCAN, we demonstrated that HDBSCAN performs better in clustering botnet traffic, achieving 11% higher accuracy in clustering data. Furthermore, our system shows significant improvement in data labeling compared with previous work. This research provides an effective solution for labeling botnet in the field of network security, aiding in the detection and prevention of malicious activities in botnets.

REFERENCES

[1] Z. Liu, X. Yun, Y. Zhang and Y. Wang, "CCGA: Clustering and Capturing Group Activities for DGA-Based Botnets Detection," 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), Rotorua, New Zealand, 2019, pp. 136-143, doi: 10.1109/TrustCom/BigDataSE.2019.00027.

[2] Trajanovski, T.; Zhang, N. An Automated Behaviour-Based Clustering of IoT Botnets. Future Internet 2022, 14, 6. https://doi.org/10.3390/fi14010006

[3] Chen, Wei-Yu; Shieh, Ce-Kuen; Chang, Jyh-Biau. A Multi-type Botnet Classifier for Real Traffic Based on BotCluster. https://thesis.lib.ncku.edu.tw/thesis/detail/2328c0d4868bda0532cb46df1530941b/

[4] C.-Y. Wang, C.-L. Ou, Y.-E. Zhang, F.-M. Cho, J.-B. Chang, and C.-K. Shieh, "BotCluster: A Session-based P2P Botnet Clustering System on NetFlow," Computer Networks, Volume 145, 9 November 2018, pp. 175-189.

[5] Ester, M., Kriegel, H. P., Sander, J., and Xu, X.. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the 2nd ACM International Conference on Knowledge Discovery and Data Mining (KDD). 226–231.

[6] Campello, R.J.G.B., Moulavi, D., Sander, J. (2013). Density-Based Clustering Based on Hierarchical Density Estimates. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2013. Lecture Notes in Computer Science(), vol 7819. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-37456-2_14

[7] Rosenberg, Chuck; Hebert, Martial; Schneiderman, Henry (2018). Semi-Supervised Self-Training of Object Detection Models. Carnegie Mellon University. Journal contribution. https://doi.org/10.1184/R1/6560834.v1

[8] Rahbarinia B., Perdisci R., Lanzi A. and Li K., Peerrush "Mining for unwanted p2p traffic", Journal of Information Security and Applications, 2014, pp. 194-208.

[9] Malware Capture Facility Project (2020) – [online] Available at: https://www.stratosphereips.org/datasets-malware.

[10] P. Torres, C. Catania, S. Garcia, C.G. Garino, "An analysis of Recurrent Neural Networks for Botnet detection behavior" 2016 IEEE Biennial Congress of Argentina (ARGENCON), 2016.

[11] N. Koroniotis, N. Moustafa, E. Sitnikova, and J. Slay, "Towards Developing Network Forensic Mechanism for Botnet Activities in the IoT Based on Machine Learning Techniques," in International Conference on Mobile Networks and Management, 2017, pp. 30-44: Springer..

[12] Abien Fred M. Agarap, " Deep Learning using Rectified Linear Units (ReLU)", arXiv:1803.08375, 2018.

[13] K. He, X. Zhang, S. Ren and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 1026-1034, doi: 10.1109/ICCV.2015.123.

[14] Diederik P. Kingma and Jimmy Ba, "ADAM: A Method for Stochastic Optimization", arXiv:1412.6980, 2014

[15] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

[16] Rahbarinia B., Perdisci R., Lanzi A. and Li K., Peerrush "Mining for unwanted p2p traffic", Journal of Information Security and Applications, 2014, pp. 194-208.

[17] Malware Capture Facility Project (2020) – [online] Available at: https://www.stratosphereips.org/datasets-malware.

[18] A. Kumar, N. Kumar, A. Handa, S.K. Shukla "PeerClear: Peer-to-Peer Bot-net Detection", International Symposium on Cyber Security Cryptography and Machine Learning, 2019, pp.279-296.

[19] P. Gahelot, N. Dayal, "Flow Based Botnet Traffic Detection Using Machine Learning", Proceedings of ICETIT, 2019, pp.418-426.

[20] C.D. McDermott, F. Majdani, A.V. Petrovski, " Botnet Detection in the Internet of Things using Deep Learning Approaches", 2018 International Joint Conference on Neural Networks (IJCNN), 2018

[21] R.H. Hwang, M.C. Peng, V.L. Nguyen, Y.L. Chang, "An LSTM-Based Deep Learning Approach for Classifying Malicious Traffic at the Packet Level" Applied Sciences, 2019.

[22] J. Roosmalen, H. Vranken, M. Eekelen, "Applying Deep Learning on Packet Flows for Botnet Detection" Symposium on Applied Computing, 2018.

[23] I. Letteri, G.D. Penna, G.D. Gasperis, "Botnet Detection in Software Defined Networks by Deep Learning Techniques" International Symposium on Cyberspace Safety and Security, 2018, pp.49-62.

[24] L.F. MAIMO,´ A.L.P. G´ OMEZ, F.J.G. CLEMENTE, M.G. P´ EREZ,´ AND G.M. PEREZ, "A Self-Adaptive Deep Learning-Based System for Anomaly Detection in 5G Networks" IEEE Access, 2018, pp.7700-7712.

[25] A. Pektas¸, T. Acarman, " Botnet detection based on network flow summary and deep learning" Int J Network Mgmt, 2018.

[26] W. Wang, M. Zhu, X. Zeng, X. Ye, Y. Sheng, " Malware Traffic

[27] Benchmarking Performance and Scaling of Python Clustering Algorithms , Sep. 2023, [online] Available: https://hdbscan.readthedocs.io/en/latest/performance_and_scalability.html

[28] scikit-learn-contrib, Sep. 2023, [online] Available: http://contrib.scikit-learn.org/

[29] Classification Using Convolutional Neural Network for Representation Learning" 2017 International Conference on Information Networking (ICOIN), 2017, pp.712-717.

[30] Stratosphere IPS. (2020). CTU-13 Dataset — Stratosphere IPS. [online] Available at: https://www.stratosphereips.org/datasets-ctu13

[31] D. S. Terzi, R. Terzi and S. Sagiroglu, "Big data analytics for network anomaly detection from netflow data," 2017 International Conference on Computer Science and Engineering (UBMK), Antalya, Turkey, 2017, pp. 592-597, doi: 10.1109/UBMK.2017.8093473.

[32] Yu, Y., Long, J., Cai, Z. "Session-Based Network Intrusion Detection Using a Deep Learning Architecture". Modeling Decisions for Artificial Intelligence. MDAI 2017. Lecture Notes in Computer Science, vol 10571. Springer, Cham. https://doi.org/10.1007/978-3-319-67422-3_13

[33] D. P. Hostiadi, T. Ahmad and W. Wibisono, "A New Approach of Botnet Activity Detection Model based on Time Periodic Analysis," 2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM), Surabaya, Indonesia, 2020, pp. 315-320, doi: 10.1109/CENIM51130.2020.9297846.

[34] D. P. Hostiadi and T. Ahmad, "Sliding Time Analysis in Traffic Segmentation for Botnet Activity Detection," 2022 5th International Conference on Computing and Informatics (ICCI), New Cairo, Cairo, Egypt, 2022, pp. 286-291, doi: 10.1109/ICCI54321.2022.9756077.

[35] K. Sinha, A. Viswanathan, and J. Bunn, "Tracking Temporal Evolution of Network Activity for Botnet Detection," arXiv.org, Aug. 09, 2019. https://arxiv.org/abs/1908.03443 (accessed Sep. 21, 2023).