GraphBot: Botnet Detection via Dynamic Graph Representation Learning

1st Siyu Cheng
School of Information Science and Technology
University of Science and Technology of China
Hefei, China
syccc@mail.ustc.edu.cn

2nd Jingxian Shuai
Institute of Advanced Technology
University of Science and Technology of China
Hefei, China
vexertron_shuai@mail.ustc.edu.cn

3rd Fengrui Xiao
School of Information Science and Technology
University of Science and Technology of China
Hefei, China
franxfr@mail.ustc.edu.cn

4th Dan Yang
Department of Mechanical and Automation Engineering
The Chinese University of Hong Kong
Hong Kong, China
1836417042@qq.com

5th Yuanyi Ma
School of Information Science and Technology
University of Science and Technology of China
Hefei, China
yym672@mail.ustc.edu.cn

6th Shuangwu Chen
School of Information Science and Technology
University of Science and Technology of China
Institute of Artificial intelligence
Hefei Comprehensive National Science Center
Hefei, China
chensw@ustc.edu.cn

2024 10th International Conference on Computer and Communications (ICCC) | 979-8-3315-0707-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICCC62609.2024.10942170

***Abstract*—Botnets, networks of compromised devices controlled remotely by attackers, are particularly threatening due to their ability to conduct various cyber-attacks, such as Distributed Denial of Service (DDoS) attacks, data theft, and spamming. Existing botnet detection methods often rely on static or manually designed features, which are increasingly ineffective against modern botnets that utilize sophisticated encryption and evasion techniques. In this paper, we propose GraphBot, a novel botnet detection method that leverages dynamic graph representation learning to overcome these challenges. GraphBot introduces the concept of superflows, which are collections of network flows exhibiting similar behavior. It constructs a superflow correlation graph with timestamps to model temporal correlations and continuously updates node representations. This dynamic learning framework effectively captures the complex and evolving patterns in network traffic. We evaluate GraphBot on two public datasets, and the experimental results demonstrate that GraphBot outperforms state-of-the-art botnet detection methods, achieving high precision and recall while maintaining a low false positive rate. These highlight GraphBot's robustness, reliability, and practical applicability in real-world network security scenarios.**

***Index Terms*—Botenet detection, Dynamic graph representation learning, Network flow behaviors.**

I. Introduction

Botnets are networks consisting of multiple hosts infected by malicious software and controlled by hackers [1]. Infected hosts can be used to perform various malicious activities, such as Distributed Denial of Service (DDoS) attacks, data theft, spamming, and etc. [2]. Compared to 2022, the first half of 2023 saw a 400% increase in IoT malware attacks [3], primarily driven by popular IoT botnets such as Mirai and Gafgyt. In the past few years, Mirai and its various variants have exploited unprotected IoT devices to conduct large-scale DDoS attacks [4], making it one of the most severe cybersecurity threats.

With the development of technologies such as the Internet of Things (IoT) and cloud services, botnets have evolved to feature covert communications and intelligent control, making the detection of botnets increasingly difficult. Botnet detection requires effective monitoring of complex network traffic. However, the employment of numerous evasion techniques significantly hampers effective detection. Specifically, during the communication stages of bots, attackers often deploy encryption protocols, like TLS, to conceal malicious data. By 2021, encryption was employed in more than 46% of botnet communications [5]. Such widespread use of encryption substantially affects the efficacy of botnet detection that relies on deep packet inspection (DPI).

Botnet detection methods are mainly divided into three categories: signature-based detection, artificial intelligence-based detection, and complex network-based detection. Signature-based techniques [6]–[8] identify botnets by comparing network traffic against predefined patterns of known bots. These methods include the use of regular expressions, blacklists, and N-gram models. These techniques are highly effective against known botnet families but struggle with new or unknown botnets due to tactics like code obfuscation used by attackers. On the other hand, artificial intelligence-based approaches detect malicious activities of botnets by employing algorithms, such as Naive Bayes [9], Support Vector Machines [10], Random Forest [11], and neural networks, to assess key network attributes. These attributes include the number of packets,

packet sizes, the timing between data packets, and behaviors like repeated connections to the same servers. However, the dynamic nature of botnet attributes complicates manual feature selection, and encryption and traffic interference tactics further challenge these methods, affecting detection accuracy. An alternative strategy employs the construction of correlation graphs from botnet communications, which are then analyzed for community detection [12], [13]. However, while this method enhances detection capabilities, it still confronts challenges related to the selection of metrics and thresholds. Botnets can adaptively adjust to specific metrics to evade detection.

In this paper, we introduce a superflow-based dynamic graph representation learning method for efficient botnet detection . Due to the similarities in attack targets and methods among some botnets, we propose the concept of 'superflows', which are collections of network flows that exhibit identical behaviors. We create a timestamped superflow-host correlation graph. We propose a dynamic graph representation learning framework that models the continuous changes in network traffic behavior by learning dynamic node representations for hosts and superflows. Specifically, when network traffic arrives, the associations between hosts and superflows are first encoded into message vectors. These message vectors are then integrated with the representations of the relevant hosts and superflows to update their dynamic node representations. Next, we dynamically cluster hosts with similar network traffic behavior into superflow clusters and update the representations of the superflow clusters. Finally, we introduce a detection module to identify network traffic that exhibits botnet behavior. Our main contributions are threefold which are summarized as follows:

- We introduce a dynamic graph representation learning framework and the concept of superflows to address the problem of botnet detection. Innovatively, we consider the relationships between hosts and superflows as a temporal correlation graph, and we develop dynamic node representations to effectively capture the interconnected features of large-scale network traffic.
- We also propose a cluster-based dynamic information diffusion mechanism to aggregate superflow features. This mechanism dynamically clusters hosts with similar network traffic behavior into superflow clusters and learns their respective superflow representations. This is the first detection method to use continuous-time dynamic graphs specifically for botnet detection.
- We validated the effectiveness of the proposed method for detecting botnet traffic on public datasets. Additionally, our method excels at extracting features of network host clusters and analyzing the behavior of similar hosts.

II. Background

In this section, we introduce the concept of 'superflows' and present the temporal interaction graph used for dynamically identifying botnet networks, framing the botnet detection issue as a dynamic cluster network detection problem.

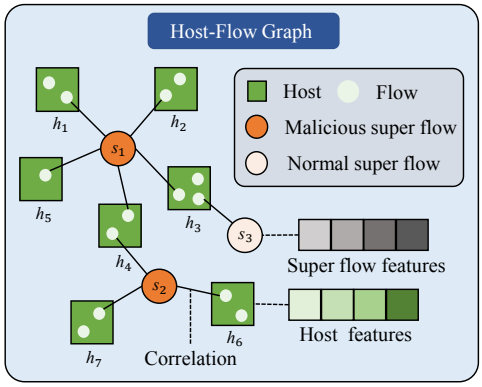

Fig. 1. Host-superflow temporal graph. Host-superflow temporal graph focuses on the temporal correlations between network flows generated by hosts within a given time window and the superflows.

A. Superflow

Typically, a network flow is characterized by a set of five-tuples $[ip_{src}, ip_{dst}, port_{src}, port_{dst}, proto]$, which ip_{src} is the source IP address, ip_{dst} is the destination IP address, $port_{src}$ is the source port, $port_{dst}$ is the destination port, and $proto$ is the protocol type.

Based on research [5], [14], it has been found that botnets within the same family tend to exhibit similarities in their attack targets and methods, as they typically share similar malware and utilize the same software frameworks. Due to the cost benefits associated with code reuse, these botnets frequently employ default, fixed parameters, particularly in terms of message length and timing intervals [15]. Consequently, the behavior of hosts within the same botnet family is highly similar, resulting in similar and relatively fixed traffic patterns. Building on this idea, we propose the concept of super flow, which represents a collection of network flows that exhibit similar behaviors and are directed towards the same destination address. We categorize network flows into different super flows using a set of four-tuples $[ip_{dst}, ab_{up}, ab_{down}, proto]$, where abp_{up} and abp_{down} are the average bytes per packet of the upward and downward traffic of the network flow.

B. Host-Superflow Temporal Graph

We categorize all network flows $F = \{f_1, f_2, \ldots, f_m\}$ into superflows $S = \{s_1, s_2, \ldots, s_n\}$ using a four-tuple $[ip_{dst}, ab_{up}, ab_{down}, proto]$. We have designed a heterogeneous graph $G = (H, S, E)$ to represent the associations between host network flows and superflow nodes, as shown in Fig. 1. Let H denote the set of hosts, with each host uniquely identified by its IP address. Let S represent the set of superflows, each initialized by the characteristics of the first flow that connects to it, comprising statistical information extracted from the flow. We define $E = \{e | e = (h_i, s_j, t_{ij}) \mid t^- \leq t_{ij} \leq t\}$, where $h_i \in H$ and $s_j \in S$ represent all connections between hosts and super flows. If a flow f from host h_i is associated with a super flow s_j at time t_{ij} within a

specific time window $\Delta t = t - t^-$, an edge $e = (h_i, s_j, t_{ij})$ connecting these two nodes is added to E.

We regard the associations between a host's network flows and superflows as a temporal graph, specifically focusing on the temporal correlations between network flows generated by hosts within a given time window and the super flows. In this host-superflow temporal graph, new hosts and super flows emerge over time and disappear as the time window shifts. This dynamic nature more closely aligns with the practical scenarios of detecting botnet activities.

III. GraphBot

In this section, we introduce a efficient botnet detection method and system architecture based on dynamic graph representation learning. By leveraging the similarity in network behavior among botnet clusters, we utilize the associations between botnet hosts and superflows to propagate spatio-temporal information. Initially, we construct a heterogeneous graph to associate all relationships between hosts and superflows, and maintain dynamic representations of hosts and superflows. As new traffic emerges, the association information between hosts and superflows is encoded into association vectors, which are then fused with the representations of the relevant hosts and superflows to update the dynamic node representation. Next, we use ynamic graph representation learning to capture the evolving representations of cluster over time. The representations and associations of hosts and superflows gradually change as new traffic emerges, and the cluster representations change accordingly. Finally, we design a botnet detection module that classifies superflows to identify botnet behavior patterns, subsequently marking the corresponding hosts as botnet nodes. The system framework is shown in Fig. 2, and it consists of four modules: dynamic traffic processing module, dynamic node representation module, dynamic cluster representation module, and botnet detection module.

A. Dynamic Node Representation

To integrate of arriving traffic with node memory, we use a vectorized representation for new network flow associations. Timestamps offer a wealth of temporal information into network behavior patterns. Because network flows that occur at different times can indicate distinct patterns of network activity. When a new network flow (h_i, s_j, t_{ij}) association arrives, we calculate the time interval between the current network flow and the last one within the same superflow, defined as $\Delta t_j = t_{ij} - t_j^-$, where t_{ij} is the timestamp of the current flow and t_j^- is the timestamp of the previous flow. These timestamps are encoded by $T(\Delta t_j)$, capturing the temporal features derived from the time intervals. Following the encoding method outlined in [16], we encode the time interval Δt as follows:

$$T(t_{ij}) = [\cos(w_1 \Delta t_j + b_1), \ldots, \cos(w_n \Delta t_j + b_n)], \quad (1)$$

where $w_1, ..., w_n$ and $b_1, ..., b_n$ are trainable parameters.

After obtaining the time encodings, we process the network flow associations to generate information representations

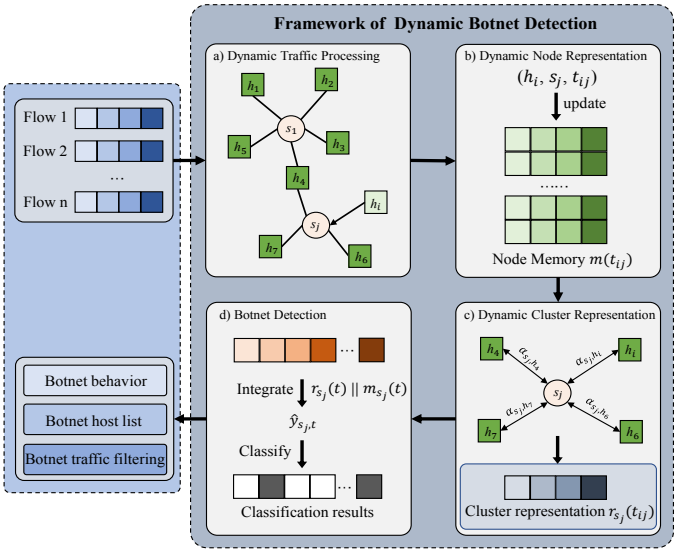

Fig. 2. Dynamic representation learning architecture for botnet detection using superflows, including dynamic traffic processing module, dynamic node representation module, dynamic cluster representation module, and botnet detection module.

$I_{h_i}(t_{ij})$, $I_{s_j}(t_{ij})$ for host h_i and superflow s_j, respectively. These representations effectively capture the temporal correlation features of both the hosts and the superflows. The formula for message representation $I_{s_j}(t_{ij})$ is as follows:

$$I_{s_j}(t_{ij}) = \sigma\left(W_n\left[m_{s_j}(t_{ij}^-) \| m_{h_i}(t_{ij}^-) \| T(t_{ij})\right] + b_n\right), \quad (2)$$

where $||$ signifies the operation of concatenation, $m_{s_j}(t_{ij}^-)$ and $m_{h_i}(t_{ij}^-)$ represent the memory state of node s_j and h_i prior to t_{ij}, and W_n and b_n are trainable parameters. $I_{h_i}(t_{ij})$ can be calculated in the same way.

Memory is used to retain long-term patterns of traffic behavior. After obtaining the information representations, we use a GRU [17] to integrate the memory of the relevant nodes with the information representations to update the node memory representations. The dynamic memory of the host and superflow are initialized as a zero vector and learns from the global host-superflow associative behavior. The formula for this update is as follows:

$$m_{s_j}(t_{ij}) = \text{GRU}\left(m_{s_j}(t_{ij}^-), I_{s_j}(t_{ij})\right). \quad (3)$$

The memory of host h_i can be updated using the same procedure.

B. Dynamic Cluster Representation

Using deep learning models for dynamic cluster detection on continuous-time data presents a challenging task. In this section, we have developed a continuous-time network flow clustering algorithm aimed at learning the cluster information of dynamic network behaviors. As new network traffic arrives, the cluster structure is incrementally updated to reflect the incoming data. This algorithm not only continuously updates the representations of clusters but also helps to prevent cluster

drift and extensive numerical matrix computations. We tend to cluster nodes based on similar traffic features and network behaviors.

To address the problem of botnet detection, we model the network as a host-superflow association graph and form superflow clusters based on host behaviors. Each cluster is represented by a superflow that embodies the cluster's network flow behavior, serving as the cluster center. Hosts with similar behaviors and superflows with similar network flow behaviors are assigned to these clusters. Additionally, as new network flows arrive, we continuously update the cluster structures. Below, c_{s_j} denotes the cluster centered around s_j.

When new network flow associations arrive, both the graph structure and node representations undergo changes. We consider each involved superflow node as a cluster center, with its neighbors regarded as cluster members. Taking the superflow node s_j as an example, we consider s_j as the center of the cluster c_{s_j}. For s_j, we use $N(s_j)$ to represent the set of cluster members within c_{s_j}, which consists of s_j's neighbors. To measure the degree of association between hosts and the cluster, we employ an attention mechanism to model the membership association strength, denoted as α. We first create a representative query vector for the potential cluster c_{s_j} using the memories of the members in $N(s_j)$, as illustrated below:

$$q_{s_j}(t_{ij}) = \max_{z \in \mathcal{N}(s_j)} \left(m_{s_j}(t_{ij}), m_z(t_{ij}) \right), \tag{4}$$

where $m_z(t_{ij})$ represents the memory associated with each member z of the cluster c_{s_j}. We calculate the association degree α of the members using an attention mechanism as follows:

$$e_{s_j,z}(t_{ij}) = \mathbf{v}^T \sigma \left(W_e q_{s_j}(t_{ij}) \| m_z(t_{ij}) \right), \tag{5}$$

$$\alpha_{s_j,z}(t_{ij}) = \text{softmax} \left(e_{s_j,z}(t_{ij}) \right), \tag{6}$$

where $\mathbf{v}^T$ and $\mathbf{W}_e$ represent parameters that can be learned, while $\alpha_{s_j,z}(t_{ij})$ indicates the association degree of membership of node z within its cluster c_{s_j}.

Based on the calculated membership strengths, we aggregate the memories of members within the cluster $c(s_j)$ to compute the cluster representation $r_{c(s_j)}(t_{ij})$, which is given by the following formula:

$$r_{c_{s_j}}(t_{ij}) = \sum_{z \in \mathcal{N}(s_j)} \alpha_{s_j,z}(t_{ij}) m_z(t_{ij}). \tag{7}$$

Hosts connected to the same superflow cluster exhibit similar network behaviors and are more likely to be associated with similar superflow clusters. After obtaining the cluster structure and capturing the cluster representation, we disseminate messages across the entire cluster to propagate the clustering effects of network behavior. Message representations have been generated for nodes directly connected, and next, we recursively propagate this new information to other cluster members. Thus, when a new network flow (h_i, s_j, t_{ij}) arrives, we calculate its message representation based on the membership strength of other cluster members, as determined by the following formula:

$$I_{s_j,z}(t_{ij}) = \alpha_{s_j,z}(t_{ij}) I_{s_j}(t_{ij}), \tag{8}$$

where z is the member of the cluster c_{s_j}. After calculating the cluster-based message representation $I_{s_j,z}(t_{ij})$, we can update the memory m_z of each node within cluster c_{s_j} using equation (3). This process allows the influence of new network flows to be propagated and diffused throughout the cluster.

C. Botnet Detection Module

In this section, we detect superflows within the time interval from t to $t+\Delta t$ to identify botnets. We integrate the superflow node representation and cluster representation obtained above, with the calculation formula as follows:

$$\hat{y}_{s_j,t} = W_y \left[r_{c_{s_j}}(t) \| m_{s_j}(t) \right] + b_y, \tag{9}$$

where W_y and b_y are trainable parameters.

The task of botnet detection is treated as a classification problem. We utilize the cross-entropy function as the loss function, which is expressed as follows:

$$L = -\frac{1}{N} \sum_{j=1}^{N} \sum_{c=1}^{C} y_{s_j,c,t} \log(\hat{y}_{s_j,c,t}), \tag{10}$$

Where N is the number of training samples. When $c = 2$, it corresponds to a binary classification problem, where y_i and $\hat{y}_i$ represent the actual label and the model's predicted probability that the i-th sample, within the time window from t to $t+\Delta t$, belongs to botnet behavior, respectively. These predictions are typically the outputs processed by a sigmoid function, ensuring the values range between 0 and 1. Here, $y_i = 1$ indicates that the sample is part of a botnet behavior, and $y_i = 0$ indicates it is not. For multi-class problems, the principle is similar but can distinguish which botnet family the network behavior of the superflow belongs to.

IV. Experiments

In this section, we conducted experimental evaluations of our proposed botnet detection system on two public datasets. First, we assessed our method's ability to distinguish between normal network behavior and botnet behavior. Subsequently, we evaluated our method's capability to classify different botnet families.

A. Experiment Settings

We utilized a host configured with Ubuntu 16.04.7 LTS 64-bit operating system, an Intel(R) Xeon(R) Gold 5220 CPU @ 2.20GHz, 400GB of RAM, and an Nvidia A100 40G GPU as our experimental platform. The system was developed in Python, with learning models constructed using the PyTorch library and a learning rate set to 0.001. We use Tshark to process network traffic data. The botnet detection module outputs the classification results for all superflows, with the

TABLE I
PERFORMANCE OF DIFFERENT METHODS.

Dataset	Method	Botnet Detection			
		Precision	Recall	F1	FPR
ISCX	RF	0.8626	0.8540	0.8549	0.8015
	CNN-AE	0.9894	0.9884	0.9889	0.0100
	FS-Net	0.7826	0.7791	0.7808	0.2041
	GraphBot	**0.9948**	**0.9992**	**0.9970**	**0.0015**
CTU-13	RF	0.9117	0.9096	0.9090	0.9266
	CNN-AE	0.9688	0.9686	0.9687	0.0314
	FS-Net	0.9133	0.9158	0.9145	0.0860
	GraphBot	**0.9992**	**0.9952**	**0.9972**	**0.0003**

system's detection window set to 5s. If a superflow is identified as malicious, the associated network flows will be sanitized, and its host will be placed on a botnet host list.

We conducted experiments on two real-world datasets ISCX-2014 [18] and CTU-13 [19]. The ISCX-2014 dataset is divided into training and testing datasets, including 7 and 16 types of botnets, respectively. The CTU-13 dataset contains different botnet samples across 13 scenarios. We use normal traffic from the ISCX-2014 dataset and real-world captured normal traffic as background traffic. Over the course of three months, we captured real-world network flows using three hosts and saved them in pcap format.

B. Baseline and Metric

We compared our approach against three baseline methods:

- **Random Forest [11]:** This method uses handcrafted features and the machine learning method Random Forest to detect botnets. Random Forest is a widely used machine learning algorithm that improves model accuracy and robustness by constructing multiple decision trees and combining their predictions.
- **CNN-AE [20]:** This method detects botnets using one-dimensional convolutional neural networks and autoencoders, utilizing the higher reconstruction error observed in abnormal traffic compared to normal traffic.
- **FS-Net [21]:** This method captures the temporal information within each network flow's sequence. FS-Net is an end-to-end deep learning model employing a multi-layer encoder-decoder structure to extract potential sequential features from flows. It learns representative features from raw flows and then classifies them.

We set the hyperparameters of the baselines to the values adopted by their respective authors or default values. Specifically, for FS-Net, we set the hidden state dimension to 128, the number of layers to 2, and the length embedding dimension to 16. For Random Forest, we applied the default settings in scikit-learn.

For botnet detection, we use precision, recall, F1-score, and false positive rate (FPR) to evaluate the performance of the botnet detection model.

C. Performance of Botnet Detection

To verify the effectiveness of GraphBot, we compared it against several baseline methods, including RF, CNN-AE, and FS-Net, using the ISCX-2014 and CTU-13 datasets. The results are summarized in Table I, leading to the following observations.

Firstly, traditional feature-based approaches, such as RF, demonstrated relatively poor performance. This suggests that manually designed features struggle to capture the intricate patterns of network traffic that indicate botnet activities. The lower accuracy and higher false positive rates associated with RF highlight its limitations in detecting botnets effectively. FS-Net, despite incorporating sequence features, did not perform well across most metrics, suggesting the sequence features of network flows might not adequately capture the necessary characteristics for botnet detection. Secondly, while CNN-AE performed better than RF, it still did not match the performance of GraphBot. CNN-AE's higher precision and recall compared to RF illustrate its improved ability to detect botnet activities. However, its false positive rate remains relatively high, indicating it still misclassifies a notable amount of normal traffic as botnet activities.

In contrast, GraphBot consistently achieved superior performance across all metrics on both datasets. On the ISCX dataset, GraphBot's near-perfect accuracy and F1 score, combined with its exceptionally low false positive rate, underline its robustness and reliability in detecting botnet activities. Similarly, on the CTU-13 dataset, GraphBot maintained the highest precision and recall values, further validating its effectiveness. GraphBot's significant improvement over the baseline methods can be attributed to its ability to dynamically learn and model complex patterns in network traffic. Unlike RF, FS-Net and CNN-AE, which rely on static or manually designed features, GraphBot dynamically learns the relationships and interactions within network traffic, enabling it to accurately identify botnet behaviors. Additionally, GraphBot's low false positive rate indicates its ability to minimize the misclassification of legitimate traffic, which is crucial for real-world applications where reducing false alarms is essential for maintaining network efficiency and security.

In summary, GraphBot significantly outperforms the existing baseline methods due to its advanced modeling capabilities. These capabilities include dynamic learning, which allows GraphBot to effectively capture the dynamic interactions within network traffic and identify subtle and complex botnet patterns. Additionally, GraphBot achieves high precision and recall by accurately detecting true positive botnet activities while minimizing false negatives, ensuring high reliability and robustness in botnet detection. Furthermore, GraphBot maintains a low false positive rate, reducing false alarms and enhancing its practical applicability in real-world network security scenarios.

D. Performance of Botnet Family Detection

We evaluate GraphBot's effectiveness in distinguishing between different botnet families through multi-class classification on botnet data. This approach is crucial for understanding the communication behaviors of various botnets. We excluded

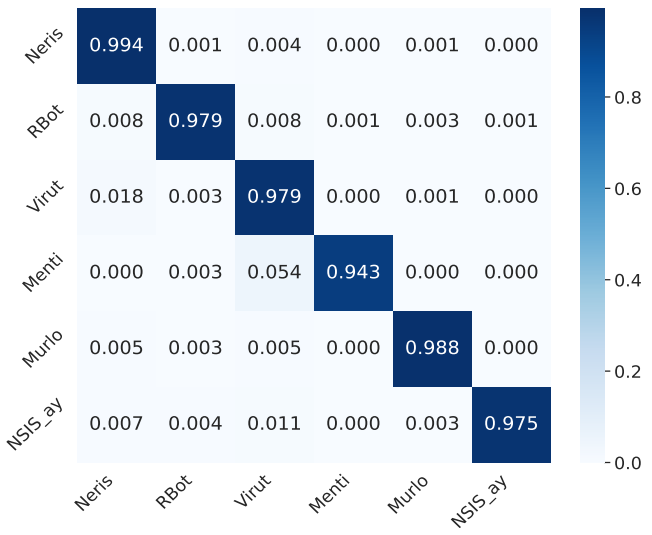

Fig. 3. Botnet family classification for CTU-13 dataset.

botnet family categories with fewer than 100 samples from the dataset to ensure the robustness of our results.

The classification results for the CTU-13 dataset are presented in Fig. 3. GraphBot demonstrated excellent performance in classifying different botnet families in this dataset. The classification accuracy for most botnet families, such as Neris, RBot, and Virut, was very high, with only minimal misclassifications. This further supports the robustness of GraphBot in identifying various types of botnets.

V. Conclusion

In this paper, we propose GraphBot, a dynamic botnet detection system utilizing a graph representation learning algorithm. By introducing the concept of superflows and capturing temporal correlations through a superflow-host correlation graph, GraphBot effectively integrates multiple network features to achieve high detection accuracy. The experimental results on public datasets demonstrate that GraphBot significantly outperforms state-of-the-art botnet detection methods, achieving superior precision and recall while maintaining a low false positive rate. Additionally, GraphBot exhibits strong performance in terms of generalization and robustness, proving its effectiveness across diverse network environments.

VI. Acknowledgment

This work was supported by the Key Science and Technology Project of Anhui (No. 202103a05020006), the National Natural Science Foundation of China (No. 62341113, 62101525, U23A20275) and the China Environment for Network Innovations (CENI) under Grant 2016-000052-73-01-000515.

References

[1] G. Vormayr, T. Zseby, and J. Fabini, "Botnet communication patterns," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2768–2796, 2017.

[2] A. Karim, R. B. Salleh, M. Shiraz, S. A. A. Shah, I. Awan, and N. B. Anuar, "Botnet detection techniques: review, future trends, and issues," *Journal of Zhejiang University SCIENCE C*, vol. 15, pp. 943–983, 2014.

[3] Zscaler ThreatLabz, "Zscaler threatlabz finds a 400% increase in iot and ot malware attacks year-over-year, underscoring need for better zero trust security to protect critical infrastructures," https://www.zscaler.com/press/zscaler-threatlabz-finds-400-increase-iot-and-ot-malware-attacks-year-over-year-underscoring, 2023, accessed: May 21, 2024.

[4] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "Ddos in the iot: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[5] Z. Fu, M. Liu, Y. Qin, J. Zhang, Y. Zou, Q. Yin, Q. Li, and H. Duan, "Encrypted malware traffic detection via graph-based network analysis," in *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*, 2022, pp. 495–509.

[6] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov, "Spamming botnets: signatures and characteristics," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 171–182, 2008.

[7] L. Liu, S. Chen, G. Yan, and Z. Zhang, "Bottracer: Execution-based bot-like malware detection," in *Information Security: 11th International Conference, ISC 2008, Taipei, Taiwan, September 15-18, 2008. Proceedings 11*. Springer, 2008, pp. 97–113.

[8] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, and W. Lee, "Bothunter: Detecting malware infection through ids-driven dialog correlation." in *USENIX Security Symposium*, vol. 7, 2007, pp. 1–16.

[9] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Usilng machine learning technliques to identify botnet traffic," in *Proceedings. 2006 31st IEEE conference on local computer networks*. IEEE, 2006, pp. 967–974.

[10] S. Kondo and N. Sato, "Botnet traffic detection techniques by c&c session classification using svm," in *International Workshop on Security*. Springer, 2007, pp. 91–104.

[11] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: detecting botnet command and control servers through large-scale netflow analysis," in *Proceedings of the 28th Annual Computer Security Applications Conference*, 2012, pp. 129–138.

[12] J. Wang and I. C. Paschalidis, "Botnet detection based on anomaly and community detection," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 2, pp. 392–404, 2016.

[13] D. Zhuang and J. M. Chang, "Enhanced peerhunter: Detecting peer-to-peer botnets through network-flow level community behavior analysis," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 6, pp. 1485–1500, 2018.

[14] A. Oprea, Z. Li, T.-F. Yen, S. H. Chin, and S. Alrwais, "Detection of early-stage enterprise infection by mining large-scale log data," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2015, pp. 45–56.

[15] Cobaltstrike Helpsystem, "Beacon covert c2 payload," https://www.cobaltstrike.com/help-beacon, 2021, accessed: November 20, 2021.

[16] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," *arXiv preprint arXiv:2002.07962*, 2020.

[17] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[18] University of New Brunswick, "Botnet datasets," https://www.unb.ca/cic/datasets/botnet.html, accessed: May 21, 2024.

[19] Stratosphere Laboratory, "Ctu-13 dataset," https://www.stratosphereips.org/datasets-ctu13, accessed: May 21, 2024.

[20] D. Jin, J. Xie, S. Chen, J. Yang, X. Liu, and W. Wang, "Zero-day traffic identification using one-dimension convolutional neural networks and auto encoder machine," in *2020 IFIP Networking Conference (Networking)*. IEEE, 2020, pp. 559–563.

[21] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "Fs-net: A flow sequence network for encrypted traffic classification," in *IEEE INFOCOM 2019-IEEE Conference On Computer Communications*. IEEE, 2019, pp. 1171–1179.