Multi-Attributed Heterogeneous Graph Convolutional Network for Bot Detection

Jun Zhao[a,b], Xudong Liu[a,b,*], Qiben Yan[c], Bo Li[a,b,*], Minglai Shao[a,b] and Hao Peng[a,b]

[a] School of Computer Science and Engineering, Beihang University, Beijing, China
[b] Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing, China
[c] Computer Science and Engineering, Michigan State University, East Lansing, Michigan, USA

ARTICLE INFO

Keywords:
Botnet detection
Bot behavioral model
Multi-attributed graph
GCN

ABSTRACT

Bot detection is a fundamental and crucial task for tracing and mitigating cyber threats in the Internet. This paper aims to address two major limitations of current bot detection systems. First, existing flow-based bot detection approaches ignore structural information of botnets, which lead to false detection. Second, they cannot identify the interactive behavioral patterns among heterogeneous botnet objects. In this paper, we propose a novel bot detection framework, namely Bot-AHGCN, which models fine-grained network flow objects (e.g., IP, response) as a multi-attributed heterogeneous graph and transforms bot detection problem into a semi-supervised node classification task on the graph. Particularly, we first build a multi-attributed heterogeneous information network (AHIN) to model the interdependent relationships among botnet objects. Second, we present a weight-learning based node embedding method, which learns the interactive behavioral patterns among bots and integrates them into weighted similarity graphs. Finally, we perform graph convolution on the learned similarity graphs to characterize more comprehensive and discriminative features of bots, and feed them into a forward neural network to identify bots. The overall experimental results on two real-world datasets confirm that Bot-AHGCN outperforms the existing state-of-the-art approaches, and presents better interpretability by introducing meaningful meta-paths and meta-graphs.

1. Introduction

Botnet attack has been regarded as one of the most serious threats against multiple industries such as finance, education, government, medical care, critical infrastructure, Internet of Thing (IoT), etc [3]. Recently, with the explosive growth of IoT platforms, an increasing number of IoT devices (e.g., camera, sensor) without protection software are going online. The large volume of low-security IoT devices have attracted hackers to use them as weapons [20]. For example, Mirai, one of the most notorious botnets, infected more than 30 million IoT devices in one day and crippled Krebs with 650 Gbps attack volume [6].

A botnet consists of a large number of compromised devices, in which each compromised device is a bot and controlled by a botmaster. Perpetrators only need to supervise a small number of botmasters to distributively manipulate the bots via command and control (*C&C*) channels. Different from traditional viruses and worms, bots can receive commands from botmasters remotely to launch a distributed cyber crime [40]. Recently, botnet attack has been causing catastrophes against cybersecurity, such as: spreading malware and virus, launching distributed denial-of-service (DDoS) attacks, sending spamming emails and advertisements, phishing, and click frauds [31]. Therefore, it is critical and urgent to develop an effective tool to detect botnets.

During the last decade, a large volume of bot detection approaches based on diverse technologies have been proposed. Generally, the majority of existing botnet detection approaches focus on particular botnet command and control (*C&C*) protocols (e.g., *HTTP*, *IRC*) or network structures (e.g., centralized or P2P). Correspondingly, existing botnets detection methods can be roughly divided into two types: flow-based and graph-based [29]. Particularly, flow-based bot detection methods [4, 26, 27, 32, 13, 39, 7, 5, 22, 2] rely on statistics or machine learning techniques to analyze botnet characteristics from each individual traffic flow, which generally focus on source IP, destination IP, port, protocol, packet size, and session duration, etc. These characteristics can be regarded as a fingerprint of each

Email:{zhaojun,liuxd,shaoml,penghao,libo}@act.buaa.edu.cn qyan@msu.edu
*Corresponding author
ORCID(s): 0000-0000-1224-2019 (J. Zhao)

flow to discriminate whether a host carries malicious traffic [4]. However, the major limitations of flow-based approaches are two-fold. First, they excessively rely on computing the statistical features from each individual network flow while ignoring the topological structure among bots, inevitably resulting in losing the important interdependent relationships among bots. Second, they compare the features learned from each isolated network flow to identify bots instead of handling the global interactive behavioral patterns, leading to an unsatisfactory performance on unseen or well-disguised botnet flows.

To address the limitations of flow-based approaches, another stream of research focused on identifying graph-based features for bot detection [9, 35, 10, 18, 24, 16]. These studies leverage various graph theories to detect bots, which mainly consider the spatial relationships in a group of network flows, by analyzing the network topology, mining similar community structures, or discovering specific subgraphs [10]. Generally, these methods are more effective than flow-based approaches, since they can tackle the topological relationships of bots and are capable of characterizing more discriminative behavioral features among bots [35].

Nevertheless, the majority of graph-based approaches expose two serious deficiencies [18]. First, similar to flow-based approaches, the graph-based methods mostly overemphasize particular rules such as similar community structure or specific subgraph, which means that predetermined rules need to be established before discriminating bots in a graph. In other words, such methods may become feeble and obsolete if attackers frequently adapt botnet structures for evading detection [10]. Second, the majority of graph-based methods build a homogeneous graph which only contains hosts (i.e., IP address) [10, 9]. These graphs cannot model complex interdependent relationships among heterogeneous network objects, such as source IP, destination IP, port, protocol, request, and response, etc. As a result, these graph-based approaches cannot identify the underlying interactive behavioral patterns among bots.

In summary, existing bot detection methods suffer from two major challenges. First, the features and rules based on flows and graphs are often too rigid for unseen network flows or adaptive topological structures. Second, they cannot handle the potential interactive relationships among fine-grained network flow objects, resulting in the inability to model the interactive patterns among bots.

In order to overcome these challenges, we present a novel bot detection framework, namely Bot-AHGCN, which models fine-grained network flow objects into a heterogeneous graph and transforms bot detection problem into a semi-supervised node classification task on the graph. Bot-AHGCN not only models more fine-grained network objects and handle more meaningful semantic relationships, but is capable of effectively learning the interactive behavioral patterns among bots for various network structure and scenarios. Moreover, Bot-AHGCN brings better interpretability by introducing the real-world semantic relationships (*see Figure 2 and Figure 4*) in network communications. The main contributions of this paper are summarized as follows:

- ***AHIN of fine-grained network flow objects***. To the best of our knowledge, we are the first to leverage Attributed Heterogeneous Information Network (AHIN) to model the interactive behavioral patterns among bots. Different from existing graph-based detection methods, AHIN is capable of modeling meaningful semantic relationships that reflect interactions among fine-grained network flow objects, such as source IPs, destination IPs, protocols, ports, requests, responses.

- ***Weight-learning based similarity embedding***. We propose a novel weight-learning based similarity embedding approach to measure the similarity between any two hosts using meta-paths and meta-graphs. The proposed similarity embedding can evaluate the importance of different meta-paths and meta-graphs to precisely characterize hosts, and integrate them into the weighted homogeneous graph.

- ***Bot-AHGCN***. We transform bot detection into a semi-supervised node classification task on the *AHIN*, and present Bot-AHGCN, a more robust and effective bot detection approach based on multi-attributed heterogeneous graph convolutional networks. Bot-AHGCN can reliably characterize the interactive behavioral relationships and attributed features of bots, which provide discriminative features even with adaptive botnet topologies.

The rest of this paper is organized as follows: In Section 2, we introduce the preliminary of this work. In Section 3, we introduce our approach to detecting bots using multi-attributed heterogeneous graph convolutional network, including AHIN construction, similarity graph embedding, heterogeneous graph convolution, and detection model training. In Section 4, we verify the effectiveness and efficiency of Bot-AHGCN on two real-world datasets. We evaluate the stability and scalability of Bot-AHGCN in Section 5, and the related work is reviewed in Section 6. Finally, a conclusion is presented in Section 7.

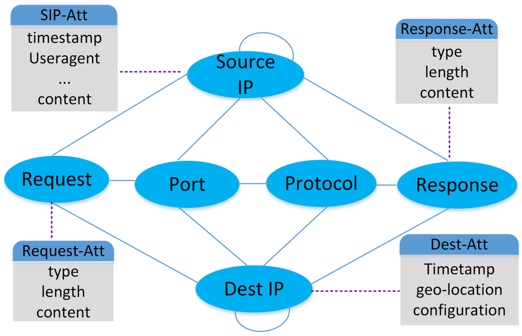


Figure 1: Schema of AHIN

2. Preliminary

In this section, we present important definitions used in our work, such as attributed heterogeneous information networks of network flows (AHIN), network schema, and meta-path.

Definition 1. ***(Attributed Heterogeneous Information Networks of Network Flows (AHIN))****. AHIN is a graph $G = (V, E, A)$, where V and E are the collection of nodes and links in G respectively, and each link describes a semantic relationship between two nodes v_i and v_j. $A = \cup_{i=1}^{m} A_i$ is a set of attributes of node V_i. Given a set of node types $T = \{t_1, t_2, ..., t_n\}$, let V_i be the set of objects of type t_i, and A_i be the set of attributes for object V_i. A specific node v_j that belongs to type t_i is associated with its corresponding attribute set $f_j = (f_{j_1}, f_{j_2}, ..., f_{j_{|A_i|}})$.*

As illustrated in Figure 1, each network flow can be represented as a six-tuple including fine-grained network objects $T = (IP_{src}, IP_{des}, Port, Protocol, Request, Response)$, and their interdependent relationships are defined as relationship ***R1~R10*** introduced in Section 4. Meanwhile, unlike conventional HIN, AHIN integrates attribute information of objects. Taking source IP as an example, it contains the session timestamp, user-agent, session contents, and package length, etc. AHIN can simultaneously handle the attribute information of bots and interactive behavioral patterns among bots, boosting the robustness and accuracy of bot detection.

In order to better understand the object types and relationship types in AHIN, it is necessary to provide the schema-level description of the network. Network schema [33] describes possible associations between different entities in a global perspective, which is formalized as Definition 2.

Definition 2. ***(Network Schema)****. The network schema [33] of AHIN denoted as $H_S = (T, R)$ is a meta template for $G = (V, E, A)$ with the object type mapping $\varphi : V \rightarrow T$ and the link type mapping $\Phi : E \rightarrow R$, which is a directed graph defined over object types T, with edges as relationships from R.*

The network schema specifies type constraints on the sets of objects and links between objects, which makes AHIN structured and guides a walker to explore semantics relations that meet specific rules in the network. For a link/relationship type R (defined in Section 3.2) connecting object type S to object type T, *i.e.*, $S \xrightarrow{R} T$, S and T are the source object type and target object type of link type R, which can be denoted as $R.S$ and $R.T$, respectively. The inverse relation R^{-1} holds naturally for $T \xrightarrow{R} S$.

Definition 3. ***(Meta-path)****. A meta-path [33] P is a path defined on a network schema $S = (N, R)$, and is denoted in the form of $N_1 \xrightarrow{R_1} N_2 \xrightarrow{R_2} \cdots \xrightarrow{R_i} N_{i+1}$, which defines a composite relation $R = R_1 \diamond R_2 \diamond \cdots \diamond R_{i+1}$, where $\diamond$ denotes the composition operator on relations.*

A meta-path can be considered as a schema instance that satisfies a particular network schema, which depicts a template of relationships between entities. For example, the relationship "two source IP (IP_{src}) send the same request (R)" can be described by a symmetrical meta-path $IP_{src} \xrightarrow{send} request \xrightarrow{send^{-}} IP_{src}$. For bot detection, we focus on 10 types of symmetrical meta-paths which start and end with source IPs (IP_{src}), as shown in Figure 2.

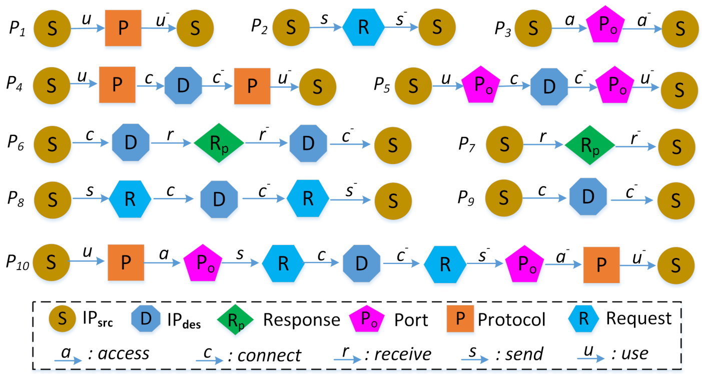


Figure 2: Meta-paths starting and ending with source IPs.

3. Bot Detection Using Multi-Attributed Heterogeneous Graph Convolutional Network

In this section, we first present the problem, and then describe our proposed framework, which consists of four major components: AHIN construction, weight-learning based host similarity embedding (i.e., node embedding), graph convolutional operation, and bot detection.

3.1. Problem Definition

Here, we formulate the problem of bot detection based on AHGCN as follows:

Definition 4. ***(Bot Detection Based on AHGCN).*** *Given an AHIN $G = \{V, E, A\}$, the meta-path set $M_P = (P_1, \cdots, P_i)$, and the meta-graph collection $M_G = (M_1, \cdots, M_j)$. The task of bot detection based on AHGCN is to : (i) measure the similarity between any two hosts (each individual source IP in G is treated as a host) based on meta-path M_P and meta-graph M_G to generate homogeneous weighted graphs between hosts A_M and A_G accordingly; (ii) construct the feature matrix of hosts X by mapping attribute information of hosts into latent vector space; (iii) perform graph convolution $GCN(A_M, X)$ and $GCN(A_G, X)$ to characterize more discriminative bot features respectively, (iv) feed the embedded features into a forward network to train a model for detecting bots.*

Where, V, E, and A represent the set of nodes, links, and attributes in heterogeneous graph G, respectively. X is the adjacency matrix of host's attributes, A_M and A_G are the weighted adjacency matrices between hosts based on meta-path and meta-graph, and $GCN(\circ)$ is graph convolutional operation. P_i and M_j represent a specific meta-path instance and meta-graph instance, respectively.

Next, we will introduce our proposed framework, which consists of AHIN construction, weight-learning based host similarity embedding, and graph convolutional operation for bot detection.

3.2. AHIN Construction

We first build the AHIN of network flows, which is capable of representing more fine-grained network flow objects. As we explore the behavioral interaction among bots, the following relationships are considered in our work.

- ***R1***: To denote a connection from a source IP to a specific destination IP, we build ***Source IP-Destination IP*** matrix M, for each element $M_{i,j} \in \{0, 1\}$, $M_{i,j}$=1 indicates source IP i visits the destination IP j.
- ***R2***: A source IP needs to use protocols to send packages. We establish the ***Source IP-Protocol*** matrix R to record the relation between source IP and used protocols, for each element $R_{i,j} \in \{0, 1\}$, $R_{i,j}$=1 indicates source IP i uses protocol j.
- ***R3***: To depict the relation between a source IP and the ports it leverages, we define ***Source IP-Port*** matrix S, for each element $S_{i,j} \in \{0, 1\}$, $S_{i,j}$=1 indicates source IP i utilizes port j.
- ***R4***: We construct a ***Source IP-Request*** matrix C to uncover the interaction between a source IP and requests it sends. For each element $C_{i,j} \in \{0, 1\}$, if $C_{i,j}$=1, there exists a sending relationship between source IP i and request j.

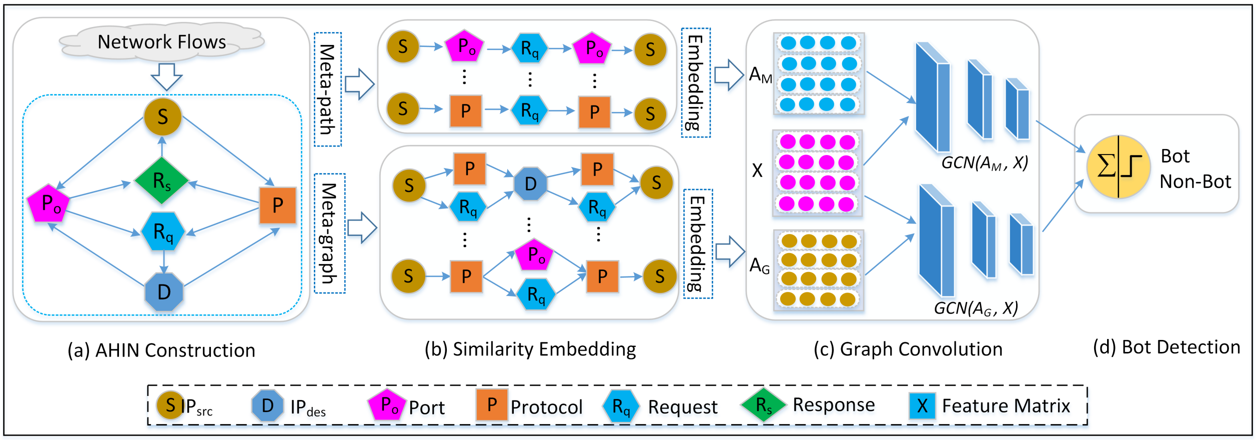


Figure 3: Framework of Bot-AHGCN. Bot-AHGCN consists of four major components: (a) AHIN construction models network flows into a heterogeneous graph to depict the interdependent relationships among fine-grained network objects, and each work flow is modeled as a six-meta-tuple $T = (IP_{src}, IP_{des}, Port, Protocol, Request, Response)$; (b) similarity embedding builds the adjacency metrics A_M and A_G leveraging Eq.1 and Eq. 2 in session 4, which can measure the similarity between any two IP_{src} based on diverse meta-paths and meta-graphs respectively; (c) graph convolution based on the similarity embedding A_M and A_G learns more discriminative interactive behavioral features of bots; (d) the embedded features are fed into a neural network to train an automated model to identify bots.

- ***R5***: To describe the relation of a source IP and received responses, we build ***Source IP-Response*** matrix M. For each element $M_{i,j} \in \{0, 1\}$, $M_{i,j}$=1 means that source IP i received response j.
- ***R6***: To portray whether a protocol utilizes a port, we build ***Protocol-Port*** adjacency matrix P. For each element $P_{i,j} \in \{0, 1\}$, $P_{i,j}$=1 indicates that protocol i utilizes port j to send packages.

As demonstrated in Figure 1, for the destination IP in AHIN, we can construct the semantic relationships between destination IP and protocol, port, request, and response similar with ***R2~R5*** respectively, denoted as ***R7~R10***. The ten types of relationships can fully tackle the interactive behavioral patterns among fine-grained objects in constructed AHIN, based on which we focus on the 10 symmetrical meta-paths where start node and end node are source IPs, as shown in Figure 2. Different from traditional heterogeneous information networks, our AHIN involves the attribute information of nodes, which can assist in conveying more richer and meaningful semantic information for improving the performance of characterizing bots. On the one hand, AHIN can model the characteristics of bots from their attribute information. On the other hand, it can tackle the interactive relationship among bots, and learn their behavioral patterns from a global perspective. Therefore, it offers better performance and interpretability for bot detection by introducing the real-world meaningful meta-paths and meta-graphs.

3.3. Weight-learning Based Similarity Embedding

For the task of bot detection, we aim to identify malicious bots from all hosts (*each individual source IP is treated as a host*) in the constructed AHIN by analyzing their similarity in terms of attributes and behavioral patterns. In order to characterize the similarity of bots we propose a weight-learning based similarity embedding method, which can measure the similarity of any two hosts based on meta-path and meta-graph, respectively. Intuitively, objects are more strongly connected by the significant meta-paths, they tend to be more similar [33]. Similarly, in our task, there is a higher probability that they are both malicious bots or legitimate hosts if they hold large amount of similar meta-path instances. Formally, we provide *Definition 5* to model the similarity of hosts based on meta-path instances.

Definition 5. ***meta-path based host similarity embedding.*** *Given a set of symmetric meta-path set $P = [P_m]_{m=1}^{M'}$, the similarity between any two hosts h_i and h_j is defined as:*

$$S_M(h_i, h_j) = \sum_{m}^{M'} w_m \frac{2\times \mid \{h_{i\to j} \in P_m\} \mid}{\mid \{h_{i\to i} \in P_m\} \mid + \mid \{h_{j\to j} \in P_m\} \mid}, \tag{1}$$

where $h_{i\to j} \in h_m$ is a path instance between host h_i and h_j following meta-path P_m, $h_{i\to i} \in P_m$ is the instance between host instance h_i and h_i, and $h_{j\to j} \in P_m$ is the instance between host instance h_j and h_j, $|\{h_{i\to j} \in P_m\}| = \mathbf{M}_{P_m}(i,j)$, $|\{h_{i\to i} \in P_m\}| = \mathbf{M}_{P_m}(i,i)$, $|\{h_{j\to j} \in P_m\}| = \mathbf{M}_{P_m}(j,j)$. $w = [w_1, \dots, w_m, \dots, w_{M'}]$ to denote the meta-path weights, w_m is the weight of meta-paths P_m. M' is the number of meta-paths.

$S_M(h_i, h_j)$ has two components: (1) the semantic overlap in the numerator, which describes the number of meta-paths between host instances h_i and h_j; (2) and the semantic broadness in the denominator, which depicts the number of total meta-paths between themselves. A larger number of meta-paths between host instance h_i and h_j indicates a higher similarity between them. Different from Pathsim [33], our proposed similarity embedding method introduces a weight vector w_m, which is a trainable coefficient vector to learn the importance of different meta-paths for characterizing bots.

Obviously, it is costly to calculate the similarity between any two hosts in the AHIN since it usually requires to randomly walk a larger number of nodes in the graph. Fortunately, in our work, it is unnecessary to walk the entire graph as we prescribe a limit based on the predefined meta-paths. Moreover, for bot detection, we only concern with the symmetrical meta-paths where both the start and end with source IPs. To calculate the similarity between any two hosts under different meta-path instances, we need to compute all the commuting matrices [33] related to them following the meta-paths. Given a meta-path set $P = \sum_m^{M'} \{A_1, A_2, \cdots, A_{l+1}\}$, the meta-path based commuting matrix can be defined as $C_P = U_{A_1A_2} \circ U_{A_2A_3} \cdots \circ U_{AA_{l+1}}$, where $C_P(i,j)$ represents the probability of object $i \in A_1$ reaching object $j \in A_{l+1}$ under the path P, and $\circ$ is a connection operation. These symmetric meta-paths not only efficiently reduce the complexity of walking, but also ensures that the commuting matrix can be easily decomposed, which greatly hoists computing performance. In addition, due to the consideration of the symmetric meta-paths in AHIN, we leverage pairwise random-walk [33] to accelerate calculations.

Definition 6. ***Pairwise random-walk***. *Given a symmetric meta-path P that can be divided into two shorter paths owning the same length $P = (P_1, P_2)$, $w(x,y)$ is the pairwise random-walk probability that starts from nodes x and y terminates at the same connected node: $s(x,y) = \sum_{p_1p_2\in(P_1,P_2)} Prob(p_1)Prob(p_2^{-1}))$, where $Prob(p_1)$ and $Prob(p_2^{-1})$ are random walk probabilities under the two meta-path instances.*

With Eq. (1) and pairwise random-walk, we can obtain the similarity embedding between any two hosts h_i and h_j under a meta-path set $P = \sum_m^{M'} \{P_m\}$, and eventually can learn a homogeneous weighted ***host-host*** similarity graph (i.e., adjacent matrix of source IPs) from the AHIN, denoted as $A_M \in \mathbb{R}^{N\times N}$, where N is the number of hosts (source IPs) in AHIN.

Meta-paths can be used to depict the individual relationship between objects, yet, it fails to model the high-order semantic relationship. For example, meta-path $host \xrightarrow{use} port \xrightarrow{use^-} host$, and $host \xrightarrow{use} protocol \xrightarrow{use^-} host$ portray two separate relationships: i) two hosts using the same port, and ii) two hosts leveraging the same protocol. However, it cannot directly portray the relationship that *"two hosts use both the same port and the same protocol"*. This calls for an advanced semantic to handle such complex high-order relationship. In order to extract more acute and discriminative behavioral patterns among bots, we furture introduce meta-graph [30] to model high-order interactive relationships.

Definition 7. ***(Meta-graph)***. *A meta-graph S is a directed acyclic graph with a single source node n_s and a single target node n_t, defined on a* AHIN *$G = (V, E, A)$ with schema $T_G = (A, R)$. Formally, a meta-graph is defined as $M_G = (V_S, E_S, A_S, n_s, n_t)$, where $V_S \in V$, $E_S \in E$ constrained by $A_s \in A$ and $R_S \in R$, respectively.*

As demonstrated in Figure. 4, we define seven types of symmetrical meta-graphs (**M1**∼ **M7**) where both source node and target node are source IPs, which models meaningful semantic relationships from the higher-order perspective. Essentially, each meta-path is a specific case of a meta-graph (e.g., the meta-paths $P_1 : host \xrightarrow{use} protocol \xrightarrow{use^-} host$ and $P_2 : host \xrightarrow{send} request \xrightarrow{send^-} host$ in Figure. 2 are particular cases of meta-graph M_1 in Figure 4). Compared to meta-path, meta-graph can simultaneously hold more complex high-order semantic relationships from a group of meta-paths. For example, M_1 depicts that two hosts are related if they both co-use the same protocol and send the same request. To learn more comprehensive behavioral patterns between bots from various meta-graphs, we propose the meta-graphs based similarity embedding. We first introduce the $CouMG$ to record the number of meta-graph instances that allow two nodes to reach.

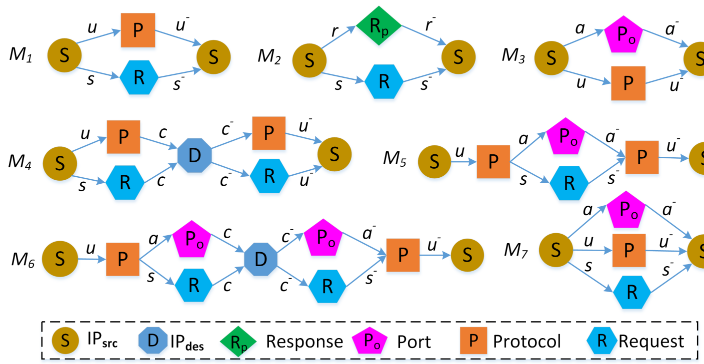


Figure 4: Meta-graphs where both start and end with source IPs.

Definition 8. ***CouMG:*** *Given AHIN $G = \{V, E, A\}$, and meta-graph set $M_G = (M_1, M_2, \cdots, M_{L+1})$, $CouMG$ is a counting function that record the number of meta-graph instances such that $CouMG_M(v_i, v_j) = C_{M_G}\{v_i, v_j\}$ where $C_{M_G} = W_{M_1M_2} \cdot W_{M_2M_3}, \cdots, W_{M_L,M_{L+1}}$, and $W_{M_L,M_{L+1}}$ is the adjacency matrix between type A_k and A_{k+1} under the meta-graph M.*

Definition 9. ***meta-graph based host similarity embedding.*** *Given a meta-graph set $MG = \{G_m\}_{m=1}^{M'}$, similar with Eq. 1, the meta-graph based similarity between any two hosts h_i and h_j can be defined as:*

$$S_G(h_i, h_j) = \sum_{m}^{M'} w_g \frac{2 \times CouMG_M(h_i, h_j)}{CouMG_M(h_i, h_i) + CouMG_M(h_j, h_j)} \tag{2}$$

where $CouMG_M(h_i, h_j)$ is the number of meta-graph M between host instances h_i and h_j, $CouMG_M(h_i, h_i)$ is that between host h_i and h_i, and $CouMG_M(h_j, h_j)$ is that between host h_j and h_j. $w_g = \{w_1, w_2, \cdots, w_{M'}\}$ is the weight vector that learns the importance of meta-graphs for measuring host similarity.

As mentioned earlier, commuting matrix has been used to compute the counting based host similarity embedding for meta-paths. For a given meta-path $P = \{A_1, A_2, \cdots, A_{l+1}\}$, we can build a matrix $W_{A_iA_j}$ as the adjacency matrix between type A_i and A_j. Consequently, the commuting matrix following meta-path P is $C_p = W_{A_1A_2} \cdot W_{A_2A_3}, \cdots, W_{A_lA_{l+1}}$. However, for the meta-graphs, the task becomes more challenging as it introduce joint nodes and links in a meta-graph. For example, for M_5 in Figure. 4, there are two individual meta-paths running through the meta-graph, involving (S, P, P_o, P, S) and (S, P, R, P, S). Here, we propose to "glue" the semantics of (P, P_o, P) and (P, R, P), which holds the higher-order semantic relationship. Obviously, a meta-graph is composed of multiple meta-paths, here, we introduce the Hadamard product [42] to "glue" the semantic relations from multiple meta-paths in a meta-graph. As a general example, Algorithm 1 shows the computational principle of commuting matrix for the meta-graph M_5 in Figure 4. Where, $\circ$ denotes Hadamard product, which can capture the high-order semantic among connected meta-paths. Note that not limited to M_5, all meta-graphs defined in this paper can be computed similar to Algorithm 1.

Algorithm 1 Computing commuting matrix for $\mathbf{C}_{M_5}$
Input: $G = \{V, E, A\}$, meta-graph set $M_G = \{M_1, M_2, \cdots, M_n\}$.
Output: counting commuting matrix;
1: Compute C_{P_1}: $C_{P_1} = W_{PR} \cdot W_{PR}^T$
2: Compute C_{P_2}: $C_{P_2} = W_{PP_O} \cdot W_{PP_O}^T$
3: Compute C_C: $C_C = C_{P_1} \circ C_{P_2}$
4: Compute C_{M_5}: $C_{M_5} = W_{SP} \circ C_C \circ W_{SP}^T$
5: **return** C_{M_5}

By computing the similarity of any two hosts according to the meta-graph MG, we can construct another ***host-host*** similarity graph (i.e., adjacency matrix of hosts) $A_G \in \mathbb{R}^{N\times N}$, where N is the number of hosts. It is worth mentioning that all meta-paths based commuting matrices in our meta-graphs (**M1**~**M7**) have been calculated and stored when constructing the meta-paths based host similarity embedding. Therefore, meta-graph based host similarity embedding only increases the cost of the Hadamard product operation.

3.4. GCN based Bot Detection

Existing botnet detection methods mostly rely on a large number of labeled data for model training. However, the number of bot flows is not substantial. Moreover, a botnet is essentially a graph-based network. In this paper, we model network flows in a multi-attributed heterogeneous graph and transform bot detection into semi-supervised node classification on the graph, which can be trained using a small number of labeled samples. Next, we show how to implement bot detection based on the constructed host similarity graph A_M and A_G, and expound how to train the weight w_m and w_g for meta-paths and meta-graphs, respectively.

As mentioned above, we have established two $N \times N$ homogeneous weighted graph (i.e., adjacent matrix of hosts) A_M and A_G using meta-paths (using Eq. (1)) and meta-graphs (using Eq.2) respectively, which hold the behavioral similarity of any two hosts. Where, N is the number of host instances in A_M and A_G, $A_{M_{ij}} = A_{M_{ji}} = S_M(i,j)$ and $A_{G_{ij}} = A_{G_{ji}} = S_G(i,j)$. Meanwhile, in order to make use of the attributed information of hosts, we train the Word2vec [23] to model the features matrix $X \in \mathbb{R}^{N\times d}$, where N is the number of hosts in A_M and A_G, and d is the dimension of host feature. Naturally, with the learned similarity graphs and feature matrix of hosts, we can leverage the classical GCN [19] to characterize the discriminative features of bots. Conventionally, the layer-wise propagation rule of GCN can be defined as below.

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} A \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}) \tag{3}$$

where, $A = A + I_N$ is the adjacency matrix of hosts with self-connections, I_N is the identify matrix, $D_{ii} = \sum_j A_{ij}$ and $W^{(l)}$ is a l_{th} layer trainable weight matrix. $\sigma(\cdot)$ denotes an activation function, such as $relu$. $H^{(l)} \in \mathbb{R}^{N\times d}$ is the matrix of activation in the l_{th} layer, and the original is $H^{(0)} = X$. We respectively conduct the graph convolution on A_M and A_G to comprehensively model the discriminative characteristics of bots. Particularly, we first implement A_M based GCN.

$$Z_M = f(X, A_M) = \sigma(\hat{A}_M \cdot relu(\hat{A}_M X W_M^{(0)}) W_M^{(1)}) \tag{4}$$

where $W_M^{(0)} \in \mathbb{R}^{d\times H}$ is an input-to-hidden weight matrix for a hidden layer with H feature maps. $W_M^{(1)} \in \mathbb{R}^{H\times F}$ is a hidden-to-output weight matrix. $X \in \mathbb{R}^{N\times d}$, N is the number of hosts and d is the dimension of features. σ is a activation function, such as $sigmoid$. The $\hat{A}_M = \tilde{D}^{-\frac{1}{2}} \tilde{A}_M \tilde{D}^{-\frac{1}{2}}$ can be calculated offline. Similarly, with similarity embedding A_G and feature matrix X, we can obtain

$$Z_G = f(X, A_G) = \sigma(\hat{A}_G \cdot relu(\hat{A}_G X W_G^{(0)}) W_G^{(1)}) \tag{5}$$

where, the meaning of $W_G^{(0)}, W_G^{(1)}, X$ and σ are same as the corresponding parameters in Eq. (4). Notes that $\hat{A}_M = \tilde{D}^{-\frac{1}{2}} \tilde{A}_M \tilde{D}^{-\frac{1}{2}}$ and $\hat{A}_G = \tilde{D}^{-\frac{1}{2}} \tilde{A}_G \tilde{D}^{-\frac{1}{2}}$ can be computed in a pre-processing step. Here, we leverage cross-entropy as loss function to quantify the error of our model for bot detection:

$$Loss(Y_{lf}, Z_{lf}) = -\sum_{l\in Y_l} \sum_{f=1}^{F} Y_{lf} \cdot lnZ_{lf} \tag{6}$$

Where, Y_{lf} is the real label, and Z_{lf} is a corresponding label that our model predicts,

$$Z_{lf} = \alpha Z_M + Z_G \tag{7}$$

Here, α is a trainable coefficient that evaluates the importance of A_M and A_G for boosting the bot detection performance. Under the guidance of the loss function (Eq.6), we conduct stochastic gradient descent to continuously optimize the neural network weights $W_M^{(0)}$, $W_M^{(1)}$, $W_G^{(0)}$, $W_G^{(1)}$, α, w_m and w_g to train an automated bot detection model.

Table 1
Performances comparison of different meta-paths and meta-graphs for bot detection ("#" indicates that the corresponding meta-path or meta-graph is not included in CTU-13 dataset).

ID	Honeypot Dataset		CTU-13 Dataset	
	Precision	Micro-F1	Precision	Micro-F1
P_1	0.9214	0.9147	0.9143	0.9238
P_2	0.9443	0.9513	0.9314	0.9452
P_3	0.7142	0.7426	0.6957	0.7258
P_4	0.9521	0.9645	0.9416	0.9596
P_5	0.8236	0.8147	0.8302	0.7976
P_6	0.7512	0.7239	#	#
P_7	0.7116	0.7245	#	#
P_8	0.9714	0.9526	0.9742	0.9641
P_9	0.8913	0.9145	0.9015	0.9164
P_{10}	0.9743	0.9616	0.9821	0.9815
M_1	0.9613	0.9527	0.9586	0.9543
M_2	0.9218	0.9385	#	#
M_3	0.9246	0.9412	0.9174	0.9228
M_4	0.9831	0.9713	0.9870	0.9686
M_5	0.9624	0.9563	0.9617	0.9581
M_6	0.9912	0.9825	0.9876	0.9743
M_7	0.9742	0.9671	0.9635	0.9687

4. Experimental Evaluation

4.1. Datasets and Settings

In this section, the effectiveness and efficiency of Bot-AHGCN are validated on two real-world datasets, involving public botnet dataset CTU-13 [15] and our captured botnet data using honeypot systems.

CTU-13 dataset is a popular public benchmark dataset of botnet traffic that released by the CTU university in 2011, which consists of 78,754 botnet flow entries and 2743,258 normal flow entries from 13 scenarios. In the comparison experiments, we randomly select 50,000 botnet flows and 50,000 normal flows to form the final experimental dataset.

Honeypot dataset. In order to evaluate the robustness of our proposed Bot-AHGCN, it is necessary to conduct Bot-AHGCN on the latest botnet traffics. In this paper, we deploy ten honeypot systems that simulate different protocols and scenarios to capture malicious attack records, which include but not limited to timestamp, source IP, destination IP, source port, destination port, request, response, session duration, etc. From Jun 2017 to Jun. 2019, the honeypot systems have successfully lured 2,738,188 suspicious connections and sniffed out more than 50,000 botnet attacks. Moreover, we randomly select 50,000 legitimate network flows from the campus gateway.

For both of the two datasets, we randomly select 60% of samples as training set, 20% of samples as verification set, and the rest of the samples as our test set. We comprehensively evaluate the performance of Bot-AHGCN for detecting bots on the two datasets. All of the experiments are run on 16 cores Intel(R) Core(TM) i7-6700 CPU @3.40GHz with 64GB RAM and 4× NVIDIA Tesla K80 GPU. The experimental codes developed with python 3.6 are executed on TensorFlow-GPU framework supported by Ubuntu 16.0.4 operating system. We utilize precision, recall, and Micro-F1 to evaluate the performance of bot detection.

4.2. Evaluation of Meta-paths and Meta-graphs

In this paper, we introduce meta-paths and meta-graph to model the interdependent behavioral relationships among fine-grinned flow objects. Here, we evaluate the performance of different meta-paths and meta-graphs on two real-world datasets for bot detection, and the average experimental results are recorded in Table 1.

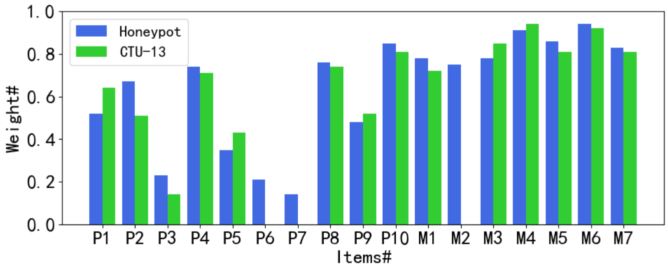


Figure 5: Normalized weight distribution of different meta-paths and meta-graphs.

The results show that different meta-paths and meta-graphs show different performance on bot detection in terms of precision and Micro-F1. Particularly, we can find that several meta-paths perform well on detecting bots, such as P_2, P_4, P_8 and P_{10}, while some meta-paths (e.g., P_3, P_6 and P_7) show poor detection results, which may be attributed to their inability to reflect the characteristics of bots. Generally, meta-graphs are more effective than meta-paths in terms of modeling higher-order semantic relationships since they are capable of combining multiple semantic relations from different meta-paths. In our proposed framework, we simultaneously consider the meta-paths and meta-graphs to model both basic and high-order semantic relationships to characterize bots.

Different meta-paths and meta-graphs have different importance for characterizing the interactive relationships of bots. In our work, we propose wight-learning based similarity embedding method to vectorize the interactive behavioral patterns of bots, which can learn the weights of different meta-paths and meta-graphs to characterize bots. Figure 5 demonstrates the normalized weight distribution of different meta-paths and meta-graphs, from which we can learn that: i) overall, the weights of meta-graphs are greater than that of meta-paths, which is because meta-graphs can model higher-order and more comprehensive semantic relationships than meta-paths; ii) meta-paths and meta-graphs containing requests and protocols own higher weights than those containing other objects (e.g., port, response). iii) as a whole, the learned weighted distribution is positively correlated with the ability of meta-paths and meta-graphs to detect bots. From Table 1 and Figure 5, we can observe that meta-paths (e.g., P_{10}) and meta-graph (e.g., M_6) with better bot detection performances own a larger weight factor.

4.3. Performance Evaluation of Bot-AHGCN

In order to verify the effectiveness of Bot-AHGCN, we evaluate it against six baseline methods: Bot-SVM, Bot-DL [26], Graph-Cluster [9], Graph-ML [10], GCN [19], HAN [37]. For the baseline methods, we implement or utilize the source code published by the authors, and adopt the same parameter set in their work.

- **Bot-SVM**. Support vector machine (SVM) is an effective model for classification tasks. As a naive flow-based baseline method, we implement Bot-SVM which leverages support vector machine to learn botnet features and identify bots from each individual network flow.
- **Bot-DL** is a state-of-the-art deep learning-based botnet detection approach, which applies deep neural networks to model the characteristics of bots by analyzing individual network flows.
- **Graph-Cluster** is an efficient bot detection approach based on graph features. The method relies on building a topological graph only involving hosts, in which diverse graph-based features are used to cluster malicious bots.
- **Graph-ML** combines graph theory and machine learning (ML) to address the problem of botnet detection, which leverages both supervised and unsupervised machine learning to establish a two-phased, graph-based bot detection system.
- **GCN** is a state-of-the-art method designed for tackling graph-based data, which can directly conduct graph convolutional operation to model features for a specific task. Here, we implement Bot-GCN, a GCN-based botnet detection method taking the topological structure and attribute information of the hosts as input.
- **HAN** is an effective attention-based heterogeneous graph embedding approach, which can evaluate the importance of node-level and path-level features for graph representation.

Table 2
Performances comparison of different methods for bot detection.

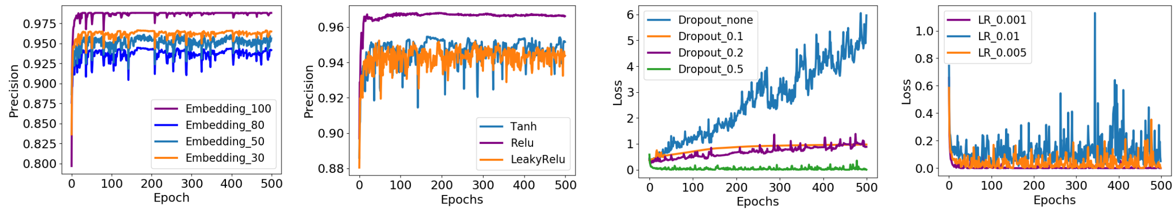
Method	Honeypot Dataset			CTU-13 Dataset		
	Precision	Recall	Micro-F1	Precision	Recall	Micro-F1
Bot-SVM	82.36 ± 0.02	84.21 ± 0.07	83.27 ± 0.12	84.14 ± 0.08	85.32 ± 1.02	84.73 ± 0.14
Bot-DL	93.15 ± 0.68	91.43 ± 0.57	92.28 ± 0.63	94.21 ± 1.04	91.34 ± 0.63	92.75 ± 0.76
Graph-ML	91.04 ± 0.87	89.37 ± 0.65	90.20 ± 0.71	92.31 ± 0.39	87.50 ± 0.56	88.48 ± 0.41
Graph-Cluster	93.21 ± 0.49	92.72 ± 0.52	92.96 ± 0.57	94.17 ± 0.23	92.36 ± 0.47	93.26 ± 0.36
GCN	92.16 ± 0.96	91.45 ± 0.62	91.80 ± 0.79	92.54 ± 0.63	91.85 ± 0.75	92.20 ± 0.69
HAN	93.14 ± 0.67	92.81 ± 1.24	92.97 ± 0.85	93.43 ± 0.74	91.89 ± 0.67	92.65 ± 0.72
GCN-AM	93.68 ± 0.53	**97.71 ± 0.35**	95.65 ± 0.43	92.65 ± 1.04	93.47 ± 0.69	93.06 ± 0.80
GCN-AG	95.21 ± 0.71	95.07 ± 0.38	95.14 ± 0.49	94.36 ± 0.89	92.35 ± 0.54	93.34 ± 0.76
Bot-AHGCN	**98.81 ± 0.24**	97.65 ± 0.41	**98.22 ± 0.36**	**98.24 ± 0.34**	**98.31 ± 0.19**	**98.27 ± 0.21**

- **Bot-AHGCN** is the proposed bot detection framework based on multi-attributed heterogeneous graph convolutional networks, which handles both the multi-attributed information and behavioral interaction of bots. It characterize bots from the perspective of meta-paths and meta-graphs, simultaneously.
- **GCN-AM** is a variant of Bot-AHGCN, which focuses on meta-path based semantic relationships to characterize bots while ignoring meta-graphs.
- **GCN-AG** is another variant of Bot-AHGCN, which only performs graph convolutional operation on A_G using meta-graphs.

Table 2 shows the performance of different methods for bot detection on CTU-13 dataset and our Honeypot datasets. We conduct 10-fold cross-validation for each method on the two datasets, and record their average performance in terms of precision, recall, and Micro-F1 in Table 2. The results show that our proposed Bot-AHGCN model outperforms all baseline methods in terms of the three evaluation metrics. The proposed Bot-AHGCN model achieves 5.5%-16.4% and 4%-14% improvement in terms of precision on Honeypot dataset and CTU-13 dataset, respectively. Our method can reach the precision peak with 99.12% and 99.27% on Honeypot dataset and CTU-13 dataset, respectively.

In fact, the improvement of Bot-AHGCN can be attributed to the following traits. First, comparing with flow-based bot detection methods such as Bot-SVM and Bot-DL, we build an attributed heterogeneous information network (AHIN) to integrate all network flow objects, which can better model the global structural relation of network flows than just considering isolated features extracted from each individual network flow. Actually, as a powerful tool for exploring associations, AHIN is capable of uncovering the concealed bots by analyzing the interactive behavioral relationships among hosts from a global perspective. Notably, the Bot-AHGCN model achieves more than 13% and 5% improvement against Bot-SVM and Bot-DL in terms of Micro-F1 on the two datasets.

Second, the existing graph-based bot detection approaches mostly rely on building a homogeneous graph containing only hosts, which analyze graph-based features (e.g., in-degree, out-degree, specific community structure, subgraph, etc) to detect botnets, such as Graph-ML and Graph-Cluster. However, such methods are overly reliant on matching particular subgraphs or communities in a coarse-grained graph, which is feeble when the attack patterns and network structures of botnets are frequently tampered for evading detection. We build AHIN of network flows, a heterogeneous fine-grained graph involving more types of network objects, such as source IP, destination IP, port, protocol, request, and response. AHIN is equipped to demystify the abnormal behavioral pattern of bots even if their topology is often adaptive. Comparing with homogeneous graph tackling only hosts, our constructed AHIN converges more rich and meaningful semantic relationships conveyed by fine-grained entities, which can model more advanced behavioral interaction to compensate the loss for topology changes. Bot-AHGCN boosts 5.6% and 4% precision against Graph-ML and Graph-Cluster on Honeypot dataset and CTU-13 dataset.



(a) Embedding size. (b) Activation selection (c) Dropout size. (d) Learning rate.

Figure 6: Parameter sensitivity analysis.

Third, the overall performance of Bot-AHGCN is superior to that of GCN and HAN due to the following reasons. HAN can learn the importance of node-level and semantic-level for graph embedding, yet, it is limited by the inability to learn more high-order semantic relationships based on meta-graph, resulting in an inferior performance. Compare with Bot-GCN, our implemented Bot-AHGCN is capable of comprehensively handling the A_M and A_G, indicating that it can simultaneously leverage the interactive behavioral features among bots based on meta-paths and meta-graphs, respectively. Meanwhile, the similarity graph A_M and A_G themselves can automatically learn the behavioral characteristics of bots, making our method more effective and practical. In fact, A_M and A_G learned by weight-learning based similarity embedding possess the discriminative bot features, and they can achieve satisfactory results even if they are fed into traditional machine learning models (e.g., SVM, KNN). We implement two SVM models based on A_M and A_G, respectively, and they can achieve excellent results with the precision and Micro-F1 exceeding 91%, which verified that our established A_M and A_G can well characterize bots. Moreover, in order to verify the effectiveness of simultaneously utilizing both meta-paths and meta-graphs to model botnet detection, we implement two variants of Bot-AHGCN, namely GCN-AM and GCN-AG, respectively. GCN-AM only holds the behavioral patterns among bots based on meta-paths while ignore that of meta-graphs. On the contrary, GCN-AG only focuses on the impact of A_G on botnet detection. Overall, Bot-AHGCN outperforms GCN-AM and GCN-AG in terms of precision and Micro-F1, indicating that it is reasonable and effective to characterize bots from both meta-paths and meta-graphs simultaneously.

5. Stability and Scalability Evaluation

5.1. Parameters Sensitivity Analysis

In this section, we conduct a large volume of comparative experiments to analyze the sensitivity of different parameters in Bot-AHGCN. We mainly focus on these hyper-parameters, including embedding size of node attributes, dropout rate, learning rate, and activation function. Specifically, embedding size of node attributes is one of the key factors to Bot-AHGCN; improper embedding dimension can cause model overfitting or underfitting. Here, we restrict other parameters to fine-tune the embedding size in (30, 50, 80, 100). As illustrated in Figure 6(a), different embedding sizes show different detection performances, and the performance of our model is outstanding in terms of accuracy and stability when the embedding size is set to 100.

Dropout is an effective way of avoiding overfitting. As shown in Figure 6(c), we find that dropout will directly affects the generalization ability of Bot-AHGCN. Obviously, the model without dropout is extremely divergent in the test dataset, which means that the generalization of the model is too feeble to be practical. When we set $dropout = 0.5$, our model can converge to a stable range.

Learning rate is an important parameter for controlling the stride of gradient descent in minimizing the loss function of Bot-AHGCN, which determines whether the model can find a set of global optimal solutions. As illustrated in Figure 6(d), All error losses with different learning rates can collectively decrease to a specific range, which means that our model is valid. Here, we set $learning\ rate = 0.001$, it allows the model to achieve minimal error loss and fluctuation. In addition, we assess the effect of different activation functions in our model, and the results are shown in Figure 6(b).

In summary, Bot-AHGCN randomly initializes other parameters and optimizes the model using back-propagation and stochastic gradient descent. Here, we leverage Relu activation function to nonlinearize the weights and set the learning rate to 0.001. In order to avoid overfitting, we set the drop rate to 0.5, and adopt the early stopping strategy with

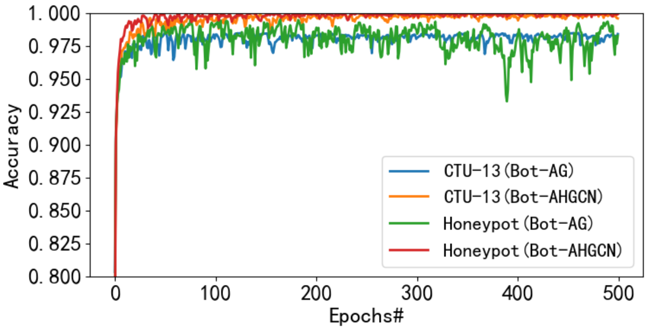


Figure 7: Availability evaluation

iteration 500, and attribute information of hosts is converted into the 100-dimensional vector by training a word2vec model [23].

5.2. Availability and Scalability Analysis

We further analyze the availability and scalability of Bot-AGHGCN. Figure 7 demonstrates the availability of Bot-AHGCN, its average true positive are 0.9912 and 0.9927 on Honeypot dataset and CTU-13 dataset respectively, which means that our method can effectively mitigate the problem of false alarm fatigue (i.e., misjudging the normal hosts as bots). Apparently, Bot-AHGCN is more effective and stable than GCN-AG as it shows less fluctuations in the optimization process.

We further evaluate the scalability by analyzing the computational complexity of Bot-AHGCN. Obviously, the complexity of Bot-AHGCN mainly consists of two parts: $O = P + Q$, where P and Q represent the computational complexity of similarity graph embedding (A_M and A_G) and graph convolution ($GCN(A_M, X)$ and $GCN(A_G, X)$), respectively. Particularly, given the AHIN $G = (V, E, A)$ and a meta-path ψ and a meta-graph ϕ, the time complexity of meta-path based similarity embedding is $P_\psi = O(V_\psi NN + E_\psi N)$, where V_ψ is the number of hosts (i.e., source IPs), E_ψ denotes the number of meta-paths connected nodes, and N is the size of adjacency matrix associated with the V_ψ. The time complexity is linear with the number of hosts and meta-paths. Correspondingly, we can obtain the computational complexity of meta-graph based similarity embedding $P_\phi = P_{\psi_i} \circ P_{\psi_j}$, where P_{ψ_i} and P_{ψ_j} represent the computational complexity of meta-paths ψ_i and ψ_j, and $\circ$ is Hadamard product. Furthermore, given a similarity embedding graph (adjacency matrix of hosts) $A \in \mathbb{R}^{N \times N}$ and feature matrix $X \in \mathbb{R}^{N \times D}$, the computational complexity of graph convolution is $Q = O(\|\xi\| NND)$, where N is the number of hosts, D is the feature dimensionality of host attributes, and $\|\xi\|$ is the number of edges in the graph, which is linear in our constructed similarity graphs. Comparing to the popular GCN model [19], Bot-AHCGN introduces the computational complexity P to measure the similarity between hosts, yet, the operation can be executed offline before detecting bots.

6. Related Work

In this section, we review some work related to bot detection, heterogeneous information networks, and graph convolutional networks.

6.1. Bot Detection

As mentioned in Section 1, the existing bot detection methods can be roughly divided into flow-based and graph-based. Particularly, flow-based bot detection methods utilize machine learning or deep learning to extract bot features from each individual network flow to discern malicious bots. Pektas *et al.* [26] proposed a botnet detection method based on network flow summary and deep learning. They applied deep neural networks to learn the attack behaviors from network flows and found that bots' behavior in terms of access record is useful for detecting botnets. Acarman *et al.* [27] studied the most expressive features in network flows for building an efficient botnet detection system, yet, these features may be obsolete as network flows are altered frequently for evading detection. Zeidabloo *et al.* [39] presented a bot detection approach by leveraging K-Means to find similar communication patterns and behaviors to identify suspicious clusters. Afterward, A larger volume of cluster-based botnet detection methods have been established [7, 5, 22, 2].

Stevanovic *et al.* [32] and Dua *et al.* [13] comprehensively compared the popular botnet detection method using machine learning, and they argued that the basic premise behind the methods is that bots pose patterns of network activity significantly different from behavior of legitimate hosts, and these patterns can be extracted by machine learning. Unfortunately, botnet attacks are becoming more and more concealed and sophisticated as botnet flows are often disguised as legitimate traffic. Therefore, such methods are noneffective for discovering the well-disguised bots. In addition, these methods ignore the topological structure among bots, inevitably leading to drop in detection performance.

Another stream of studies utilize graph-based features (e.g., in-degree, out-degree, community structure, etc.) to detect botnets [12, 16, 1, 45, 43, 44, 21, 11]. Particularly, literature [43, 44, 21] have been presented community-based patterns to identify botnets. However, the community patterns or subgraphs mostly are feeble against various network flows with different topological structures and distributions. Daya *et al.* [11] proposed a two-phased graph-based bot detection approach that used both unsupervised and supervised machine learning. Sudipta *et al.* [9] proposed a graph-based feature clustering method, which can isolate bots in clusters of small sizes while containing the majority of normal nodes in the same big cluster. Overall, the majority of existing graph-based detection approaches focus on building a homogeneous graph that involves only hosts (each IP address is treated as a host), which can barely capture interactive semantic relationships among fine-grained network flow objects.

6.2. Heterogeneous Information Networks

Heterogeneous information networks (HIN) [33] can effectively handle richer entities and meaningful semantic information through nodes and links, and it can be regarded as a conceptual representation of graph with a wide variety of entities and relationships. It has been widely used in network analysis [34], social media analysis [41], and document classification [36]. Recently, HIN has attracted attention from different application areas such as malware detection [17], opioid user identification [14]. Indeed, demystifying botnet behavior is a challenging task because of its heterogeneity in terms of different flow objects and relationships. In this paper, we build attributed heterogeneous information networks (AHIN) to model various fine-grained network flow objects to comprehensively characterize the interactive behavioral patterns between bots.

6.3. Graph Convolutional Networks

Graph convolutional networks (GCN) [19] has become an effective tool for addressing the task of graph-based scenarios, such as semi-supervised node classification [19], event classification [28], clustering [8], link prediction [25], and recommended system [38]. Given a graph, GCN can directly conduct convolutional operation on the graph to learn the nonlinear embedding of nodes. In our work, to discern and reveal the behavioral patterns of bots, we utilize GCN to learn more comprehensive and discriminative bot features.

7. Conclusion

In this paper, we propose a novel bot detection framework, namely Bot-AHGCN, which characterizes the network flows as a multi-attributed heterogeneous graph, and then transforms botnet detection problem into semi-supervised node classification problem on the graph. More specifically, we first build an AHIN of network flows, which models source IPs, destination IPs, protocols, ports, requests, responses, and their interdependent relationships. Then, we present the weight-learning based embedding method to measure the similarity of hosts from the perspective of meta-paths and meta-graphs, respectively. After that, we perform graph convolution on the embedded host similarity graphs (i.e., adjacency matrix of hosts) to characterize more comprehensive and discriminative behavioral patterns among bots. Finally, we feed the learned features into a forward neural network to train an automated model to identify bots. Bot-AHGCN achieves better detection performance than the state-of-the-art methods in bot detection. Moreover, it presents better interpretability by introducing meaningful meta-paths and meta-graphs.

Acknowledgements

We are especially grateful to the editor(s) and reviewers for their valuable comments and suggestions that help improve the quality of this manuscript. This work was supported by the National Key R&D Program China (2018YFB0803

503), the 2018 joint Research Foundation of Ministry of Education, China Mobile (MCM20180507) and the Opening Project of Shanghai Trusted Industrial Control Platform (TICPSH202003020-ZC). Specifically, Qiben Yan is supported in part by the National Science Foundation grants CNS1950171, CNS-1949753.

References

[1] Aggarwal, C.C., 2013. Outlier ensembles: position paper. Sigkdd Explorations 14, 49–58.
[2] Al-Duwairi, B., Al-Ebbini, L., 2010. Botdigger: A fuzzy inference system for botnet detection, in: 2010 Fifth International Conference on Internet Monitoring and Protection, IEEE. pp. 16–21.
[3] Albanese, M., Jajodia, S., Venkatesan, S., Cybenko, G., Nguyen, T., 2019. Adaptive cyber defenses for botnet detection and mitigation, in: Adversarial and Uncertain Reasoning for Adaptive Cyber Defense. Springer, pp. 156–205.
[4] Alieyan, K., ALmomani, A., Manasrah, A., Kadhum, M.M., 2017. A survey of botnet detection based on dns. Neural Computing and Applications 28, 1541–1558.
[5] Amini, P., Azmi, R., Araghizadeh, M., 2014. Botnet detection using netflow and clustering. Advances in Computer Science: an International Journal 3, 139–149.
[6] Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., et al., 2017. Understanding the mirai botnet , 1093–1110.
[7] Arshad, S., Abbaspour, M., Kharrazi, M., Sanatkar, H., 2011. An anomaly-based botnet detection approach for identifying stealthy botnets, in: 2011 IEEE International Conference on Computer Applications and Industrial Electronics (ICCAIE), IEEE. pp. 564–569.
[8] Chiang, W., Liu, X., Si, S., Li, Y., Bengio, S., Hsieh, C., 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. knowledge discovery and data mining , 257–266.
[9] Chowdhury, S., Khanzadeh, M., Akula, R., Zhang, F., Zhang, S., Medal, H., Marufuzzaman, M., Bian, L., 2017. Botnet detection using graph-based feature clustering. Journal of Big Data 4, 14.
[10] Daya, A.A., Salahuddin, M.A., Limam, N., Boutaba, R., 2019a. A graph-based machine learning approach for bot detection, in: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), IEEE. pp. 144–152.
[11] Daya, A.A., Salahuddin, M.A., Limam, N., Boutaba, R., 2019b. A graph-based machine learning approach for bot detection. arXiv: Cryptography and Security .
[12] Ding, Q., Katenka, N., Barford, P., Kolaczyk, E., Crovella, M., 2012. Intrusion as (anti) social communication: characterization and detection, in: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM. pp. 886–894.
[13] Dua, S., Du, X., 2016. Data mining and machine learning in cybersecurity. Auerbach Publications.
[14] Fan, Y., Zhang, Y., Ye, Y., Li, X., 2018. Automatic opioid user detection from twitter: Transductive ensemble built on different meta-graph based similarities over heterogeneous information network., in: IJCAI, pp. 3357–3363.
[15] Garcia, S., Grill, M., Stiborek, J., Zunino, A., 2014. An empirical comparison of botnet detection methods. Computers Security 45, 100–123.
[16] Henderson, K., Gallagher, B., Eliassi-Rad, T., Tong, H., Basu, S., Akoglu, L., Koutra, D., Faloutsos, C., Li, L., 2012. Rolx: structural role extraction & mining in large graphs, in: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM. pp. 1231–1239.
[17] Hou, S., Ye, Y., Song, Y., Abdulhayoglu, M., 2017. Hindroid: An intelligent android malware detection system based on structured heterogeneous information network, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM. pp. 1507–1515.
[18] Jaikumar, P., Kak, A.C., 2015. A graph-theoretic framework for isolating botnets in a network. Security and communication networks 8, 2605–2623.
[19] Kipf, T., Welling, M., 2016. Semi-supervised classification with graph convolutional networks. arXiv: Learning .
[20] Kolias, C., Kambourakis, G., Stavrou, A., Voas, J., 2017. Ddos in the iot: Mirai and other botnets. Computer 50, 80–84.
[21] Lagraa, S., François, J., Lahmadi, A., Miner, M., Hammerschmidt, C., State, R., 2017. Botgm: Unsupervised graph mining to detect botnets in traffic flows, in: 2017 1st Cyber Security in Networking Conference (CSNet), IEEE. pp. 1–8.
[22] Lu, W., Rammidi, G., Ghorbani, A.A., 2011. Clustering botnet communication traffic based on n-gram feature selection. Computer Communications 34, 502–514.
[23] Mikolov, T., Chen, K., Corrado, G.S., Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv: Computation and Language .
[24] Nagaraja, S., Mittal, P., Hong, C.Y., Caesar, M., Borisov, N., 2010. Botgrep: Finding p2p bots with structured graph analysis., in: USENIX Security Symposium, pp. 95–110.
[25] Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., Zhang, C., 2018. Adversarially regularized graph autoencoder for graph embedding , 2609–2615.
[26] Pekta, A., Acarman, T., 2018. Botnet detection based on network flow summary and deep learning. International Journal of Network Management 28, 20–39.
[27] Pektas, A., Acarman, T., 2017. Effective feature selection for botnet detection based on network flow analysis. ResearchGate , 1–5.
[28] Peng, H., Li, J., Gong, Q., Song, Y., Ning, Y., Lai, K., Yu, P.S., 2019. Fine-grained event categorization with heterogeneous graph convolutional networks. arXiv preprint arXiv:1906.04580 .
[29] Ryu, S., Yang, B., 2018. A comparative study of machine learning algorithms and their ensembles for botnet detection. Journal of Computational Chemistry 6, 119–129.
[30] Shang, J., Qu, M., Liu, J., Kaplan, L.M., Han, J., Peng, J., 2016. Meta-path guided embedding for similarity search in large-scale heterogeneous information networks. arXiv preprint arXiv:1610.09769 .
[31] Sonawane, S., 2016. A review on botnet and botnet detection methods. Int J Comput Sci Innov 1, 107–16.

[32] Stevanovic, M., Pedersen, J.M., 2016. On the use of machine learning for identifying botnet network traffic. Journal of Cyber Security and Mobility 4, 1–32.

[33] Sun, Y., Han, J., Yan, X., Yu, P.S., Wu, T., 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. Proceedings of the VLDB Endowment 4, 992–1003.

[34] Sun, Y., Norick, B., Han, J., Yan, X., Yu, P.S., Yu, X., 2013. Pathselclus: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. ACM Transactions on TKDD 7, 11.

[35] Venkatesh, B., Choudhury, S.H., Nagaraja, S., Balakrishnan, N., 2015. Botspot: fast graph based identification of structured p2p bots. Journal of Computer Virology and Hacking Techniques 11, 247–261.

[36] Wang, C., Song, Y., Li, H., Zhang, M., Han, J., 2016. Text classification with heterogeneous information network kernels, in: 13th AAAI.

[37] Wang, X., Ji, H., Shi, C., Wang, B., Cui, P., Yu, P., Ye, Y., 2019. Heterogeneous graph attention network .

[38] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J., 2018. Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM. pp. 974–983.

[39] Zeidanloo, H.R., Manaf, A.B., Vahdani, P., Tabatabaei, F., Zamani, M., 2010a. Botnet detection based on traffic monitoring, in: 2010 International Conference on Networking and Information Technology, IEEE. pp. 97–101.

[40] Zeidanloo, H.R., Shooshtari, M.J.Z., Amoli, P.V., Safari, M., Zamani, M., 2010b. A taxonomy of botnet detection techniques, in: 2010 3rd International Conference on Computer Science and Information Technology, IEEE. pp. 158–162.

[41] Zhang, J., Kong, X., Yu, P.S., 2014. Transferring heterogeneous links across location-based social networks, in: Proceedings of the 7th ACM international conference on Web search and data mining, ACM. pp. 303–312.

[42] Zhao, H., Yao, Q., Li, J., Song, Y., Lee, D.L., 2017. Meta-graph based recommendation fusion over heterogeneous information networks, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM. pp. 635–644.

[43] Zhuang, D., 2017. Peerhunter: Detecting peer-to-peer botnets through community behavior analysis. arXiv: Cryptography and Security .

[44] Zhuang, D., Chang, J.M., 2019. Enhanced peerhunter: Detecting peer-to-peer botnets through network-flow level community behavior analysis. IEEE Transactions on Information Forensics and Security 14, 1485–1500.

[45] Zimek, A., Campello, R.J.G.B., Sander, J., 2014. Ensembles for unsupervised outlier detection: challenges and research questions a position paper. Sigkdd Explorations 15, 11–22.