Information Sciences 571 (2021) 527–542

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Semi-supervised anomaly detection in dynamic communication networks

Xuying Meng [a,b], Suhang Wang [c], Zhimin Liang [a], Di Yao [a], Jihua Zhou [d], Yujun Zhang [a,*]

[a] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[b] Purple Mountain Laboratories, Nanjing, China
[c] College of Information Sciences and Technology, Pennsylvania State University, State College, PA, USA
[d] Jin Mei Communication, Chogqing, China

ARTICLE INFO

Article history:
Received 6 May 2020
Received in revised form 8 February 2021
Accepted 12 April 2021
Available online 20 April 2021

Keywords:
Anomaly detection
Semi-supervised learning
Generative adversarial networks
Self-learning

ABSTRACT

To ensure the security and stabilization of the communication networks, anomaly detection is the first line of defense. However, their learning process suffers two major issues: (1) *inadequate labels*: there are many different kinds of attacks but rare abnormal nodes in mt of these atstacks; and (2) *inaccurate labels*: considering the heavy network flows and new emerging attacks, providing accurate labels for all nodes is very expensive. The inadequate and inaccurate label problem challenges many existing methods because the majority normal nodes result in a biased classifier while the noisy labels will further degrade the performance of the classifier. To tackle these issues, we propose SemiADC, a Semi-supervised Anomaly Detection framework for dynamic Communication networks. SemiADC first approximately learns the feature distribution of normal nodes with regularization from abnormal ones. It then cleans the datasets and extracts the nodes sasainaccurate labels by the learned feature distribution and structure-based temporal correlations. These self-learning processes run iteratively with mutual promotion, and finally help increase the accuracy of anomaly detection. Experimental evaluations on real-world datasets demonstrate the effectiveness of our SemiADC, which performs substantially better than the state-of-art anomaly detection approaches without the demand of *adequate* and *accurate* supervision.

1. Introduction

Communication networks are the foundation of all Internet-based applications. However, abnormal nodes keep launching different kinds of attacks for personal profits, leading to great losses to other nodes (e.g., crashes of server nodes). Aiming at detecting nodes that launch attacks, accurate and timely abnormal nodes detection is crucial for securing Internet-based applications in dynamic communication networks . It has attracted increasing attention and many efforts have been taken [1–5]. For example, Ban et al. [1] detect abnormal nodes by dense-block detection based on the assumption that abnormal entities share resources, e.g., IP (Internet Protocol) addresses, and the shared information will lead to dense blocks on tensors. Bars et al. [2] demonstrate that the involvement of abnormal nodes will lead to abnormal communication volume.

* Corresponding author.
E-mail address: nrcyujun@ict.ac.cn (Y. Zhang).

https://doi.org/10.1016/j.ins.2021.04.056
0020-0255/

Despite the initial success, most existing semi-supervised or supervised anomaly detection algorithms for communication networks assume that abundant accurate labeled data are available for training.

In real-world anomaly detection for communication networks, we are often faced with issues of *inadequate* and *inaccurate* labels. A toy example is shown in Fig. 1, where borders of the labeled normal and labeled abnormal nodes are colored gray and red respectively. First, as shown in the figure, the majority of nodes are labeled normal while only a small portion of nodes are labeled abnormal. In addition, among these abnormal nodes, there are many types of attacks (we differentiate them with different colors in Fig. 1) and each type of attack only has few labeled data, which makes the labeled abnormal nodes *inadequate*. Second, considering the vast network flows and the limited background knowledge of various new attacks (i.e., zero-day attacks), it is almost impossible to ensure that all labels are completely accurate. Many abnormal nodes, especially those launch new attacks whose behaviors are unknown, can be *inaccurately* labeled as normal nodes in the training data (e.g., n_3 is inaccurately labeled normal for lacking latest background knowledge). Though several efforts have been made to tackle inadequacy problem [3,6,7], few of them notice the inaccuracy issue. Such inadequate and inaccurate labels challenge many existing methods because the majority normal nodes will dominate the loss function of the classifier to be learned, which could result in a biased classifier, and the inaccurate labels further degrade the performance of the classifier.*Thus, anomaly detection algorithm that can effectively learn from inadequate and inaccurate labels is in great demand.*

In this paper, we study the novel problem of anomaly detection in dynamic networks with inadequate and inaccurate labeled data. We tackle challenges of inadequacy and inaccuracy based on observations from two perspectives, i.e., the *feature* perspective and the *structure* perspective. First, from the *feature* perspective, it is demonstrated that features of abnormal nodes are very different from the normal ones, and have high probability of being similar to the labeled abnormal [7]. Thus, if we can estimate the normal and abnormal feature distribution (i.e., feature distribution of normal and abnormal nodes), we can judge if a node is normal or abnormal by checking which distribution it fits. However, due to the inadequacy of labeled abnormal, the abnormal feature distribution cannot be well estimated. Moreover, nodes with unknown new attacks may not be similar to the existing labeled anomalies. Therefore, instead of estimating both normal and abnormal feature distribution, we only estimate the normal feature distribution for the abundant data of normal nodes. Also, as the abnormal nodes are those who have launched attacks, based on the matched signatures of previously known attacks or the losses of their attacks, it is much easier to ensure abnormal labels accurate [8]. Thus, although the labeled abnormal are inadequate, these accurate abnormal nodes can help alleviate the inaccuracy problem and regularize the estimation of normal feature distribution. Utilizing features of both abnormal and normal nodes, a good estimation of the normal feature distribution paves us a way to detect abnormal nodes.

From the perspectives of *structures* (or interactions, flows), the temporal correlations of different flows can imply the potential abnormality. For example, in Fig. 2, a recent event "WannaCry" contains two kinds of attacks [9], i.e., port scan attack and infiltration attack, where the infiltration can infect the victim to be abnormal if the victim's port 445 is open. In this event, there are two kinds of temporal correlations between these attacks, i.e., multi-step correlations between the same node pair (e.g., the infiltration attack between n_1 and n_2 comes after the port scan attack on the same node pair), and multi-hop correlations between different node pairs (e.g., port scan between n_3 and n_4 happens after n_3 gets infiltrated by n_1). If n_1 is labeled abnormal for its port scan behaviors in t_1, it has a higher chance to be detected as abnormal in t_2 by multi-step correlations. Also, the multi-hop correlations between (n_1, n_3) and (n_3, n_4) can help ensure the abnormality of n_3 in t_3. These temporal correlations can greatly help better detect the abnormal nodes.

Based on the analysis from both feature and structure perspectives, we aim to build robust anomaly detection in dynamic communication networks from inadequate and inaccurate labels by simultaneously exploring time-series feature similarities and structure-based temporal correlations. The main challenges are (i) how to effectively estimate the normal feature distribution given inadequate and inaccurate labels; and (ii) how to simultaneously exploit the time-series features and temporal correlations for anomaly detection. In an attempt to solve these two challenges, we propose a novel framework SemiADC, which leverages GAN (Generative Adversarial Networks) to estimate the normal feature distribution with the regularization from the labeled abnormal nodes. We simultaneously leverage the trained GAN and temporal correlations to detect abnormal nodes in a self-training manner with the iterative feature modeling and data cleaning. The main contributions are:

- We propose SemiADC, a novel semi-supervised anomaly detection framework for dynamic communication networks, detecting abnormal nodes without the demand of adequate and accurate supervision;
- We introduce a GAN-based model to approximately learn normal feature distribution with regularization from abnormal nodes;
- Inspired by self-training, we novelly improve the learning process of feature distribution with iteratively cleaned datasets based on the time-series features and temporal correlations; and
- We demonstrate the substantial improvement of our SemiADC compared to state-of-art anomaly detection approaches.

The rest of this paper is organized as follows. In Section 2, we outline the related work and formally define the problem in Section 3. We describe the technical details of SemiADC in Section 4. In Section 5, we experimentally demonstrate the effectiveness of our framework on real-world datasets, and conclude in Section 6.

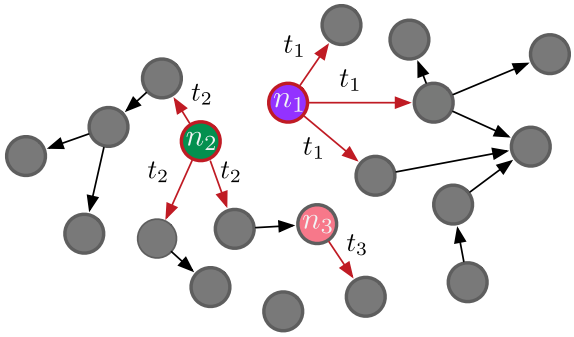


Fig. 1. Toy example of inadequate and inaccurate labels.

2. Related work

In this section, we review related work, including anomaly detection in communication networks and semi-supervised anomaly detection.

2.1. Anomaly detection in communication networks

Communication networks have become an integral part of daily life and business, however, abnormal nodes keep mounting different kinds of attacks, leading to great losses to the nodes in the networks. To detect abnormal nodes that launch attacks in communication networks, recent years have witnessed increasing attention on anomaly detection [10–17].

Some existing works treat this task as a general static classification problem [10,11] without considering the dynamic characteristics of communication networks. For example, Mukkamala et al. [10] train SVMs to differentiate anomalies from normal ones based on several discovered abnormal patterns of DoS attacks. Zhou et al. [11] utilize DNN to train a classifier to detect internet intrusion attacks. Several works try to capture these characteristics, but most of these works are from the perspective of *edges* or *graphs* [13–17]. For example, Huang et al. [13] design a data structure to detect abnormal edges based on the characteristics of traffic anomalies, such as heavy hitters and heavy changers. Nevat et al. [14] utilize the Markov chain to detect abnormal flows (i.e., edges) based on the state transitions of TCP (Transmission Control Protocol) flows. Yu et al. [15] utilize wavelet transform and DNN to detect false data injection attacks in the system (i.e., graph) based on the state deviation of their spatial and temporal data correlations. Eswaran et al. detect abnormal edges [16] and graphs [17] based on the observation that abnormal flows or graphs tend to happen as a burst of activities. However, these observations are targeted for particular attack categories like DoS or particular protocols like TCP; while for general attacks in communication networks, it leads to limited improvement in anomaly detection. In addition, most of these works assume that there are abundant accurate labeled data; while in the real world, considering the rare abnormal nodes and great amounts of normal nodes, it is expensive to provide adequate labels for abnormal nodes and requires up-to-date domain knowledge to accurately label the normal nodes. Thus, we are facing inadequate and inaccurate label problems.

To meet actual needs, we tackle the anomaly detection problem from the *nodes*' side with observations of time-series features and structure-based temporal correlations.

2.2. Semi-supervised anomaly detection

Semi-supervised learning is a branch of machine learning, which utilizes both labeled and unlabeled data to improve the model's accuracy. There are different kinds of semi-supervised methods, e.g., self-training, transductive-learning-based, co-

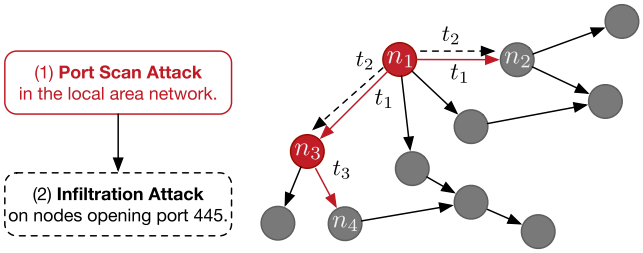


Fig. 2. Simplified process of "WannaCry", where the red nodes are labeled abnormal, the red links denote port scan attack, the black dashed links denote infiltration attack, and the black links are normal flows.

training, graph-based and generative-network-based schemes [18]. In this work, we focus on self-training and generative-network-based schemes for anomaly detection.

For self-training schemes, a.k.a self-learning, they iteratively retrain the detector with both updated labeled data and the most confident predictions [19], and there are several works utilizing self-training for anomaly detection [20,21]. For example, Ashfaq et al. [20] propose a self-training semi-supervised anomaly detection framework that evaluates fuzziness scores of unlabeled flows, updates flow labels by the scores, and retrains the classifier for abnormality with the updated datasets. However, these works assume the existing labels are fully accurate, and can not be easily adapted to our settings.

For the generative-network-based schemes, the recent most popular and efficient one is the GAN-based model, which can learn feature distribution and generate high-quality features that even can not be distinguished from the realistic ones [22–24]. For example, Akcay et al. [22] propose an encoder-decoder-encoder pipeline and evaluate the abnormality of each node based on the normal features. Li et al. [23] utilize RNN and GAN to learn time-series features of normal nodes for abnormality estimation. Zenati et al. [24] propose a BiGAN-based framework utilizing normal data to detect anomalies. These works are based on the assumption that all anomalies have different features to normal nodes, thus they only utilize the adequate labeled normal nodes and waste the existing labeled abnormal ones. Moreover, they believe all the normal labels are fully correct, while abnormal nodes can be inaccurately labeled as normal for the large number of normal nodes and for lacking knowledge of new attacks, which will directly harm the performance of anomaly detection.

Different from these existing works, we take advantage of both self-training and generative-network-based semi-supervised learning, learning normal feature distribution by generative networks with regularization from abnormal nodes, and utilizing self-learning to update the mislabeled nodes and retrain the generative-network-based detector.

3. Problem statement and notations

As abnormal nodes behave normally in most of the time, we construct node representations and labels in small sliding time windows to capture the temporal periodicity and guarantee the timeliness of detection at the same time.

Traditionally, attacks are filtered by signatures of abnormal interactions (i.e., directed links from the attackers n_i to the victims n_j) [8]. We keep the filtered abnormal interactions as (n_i, n_j, t) and include them into the set $\mathcal{S}$. Let $\mathbf{Y}^t_{n_i} = 1$ if node n_i is labeled abnormal for launching attacks in the t-th time window. Other nodes without matched signatures of known attacks are left unlabeled, and $\mathbf{Y}^t_{n_j} = 0$ if n_j is one of unlabeled nodes in the t-th time window. With the definition of $\mathbf{Y}$ and its corresponding cases, we can define the *label condition*. Considering their labels and corresponding time windows, we have two kinds of label conditions, i.e., the labeled abnormal $\mathcal{N}_l = \{(n_i, t)|\mathbf{Y}^t_{n_i} = 1\}$ and approximate normal $\mathcal{N}_u = \{(n_i, t)|\mathbf{Y}^t_{n_i} = 0\}$, and we further divide $\mathcal{N}_u$ into three groups: (1) The majority of nodes in $\mathcal{N}_u$ are normal nodes and we use $\mathcal{N}_{u\_nrm}$ to denote them. (2) Some abnormal nodes, which are mistakenly ignored during labeling due to the large number of flows although their attacks are previously *known*. We use $\mathcal{N}_{u\_knw}$ to denote the set of abnormal nodes with *known* attacks. (3) Some abnormal nodes, which are mislabeled due to lacking the latest knowledge of *unknown* zero-day attacks. We use $\mathcal{N}_{u\_unk}$ to denote the set of abnormal nodes with *unknown* attacks. In summary, we have four kinds of label conditions for anomaly detection, i.e., $\mathcal{N}_l, \mathcal{N}_{u\_nrm}, \mathcal{N}_{u\_knw}$ and $\mathcal{N}_{u\_unk}$.

All node interactions in the t-th time window construct the link set $\mathcal{A}_t$, each (n_i, n_j, t_k) represent a link from n_i to n_j in the k-th time slot of the t-th new time window. Additionally, we use $\mathbf{X}^t_{n_i} \in \mathbb{R}^{m\times w}$ to denote feature matrix of node n_i in the t-th time window, where m is the feature dimension and w is the number of time slots in each time window. The labeled abnormal feature matrix (i.e., the feature matrix of labeled abnormal nodes) and approximate normal feature matrix (i.e., the feature matrix of unlabeled nodes) are denoted as $\mathbf{X}_l$ and $\mathbf{X}_u$ based on their label conditions. With the aforementioned notations and definitions, we can now formally define the problem of semi-supervised anomaly detection for dynamic communication networks as follows:

Given $\mathbf{X}, \mathbf{Y}, \mathcal{S}$ and $\mathcal{A}_t$, we seek to learn a robust model to clean the training set (i.e., detect and move the unlabeled abnormal nodes of $\mathcal{N}_{u\_knw}$ and $\mathcal{N}_{u\_unk}$ from $\mathcal{N}_u$ to $\mathcal{N}_l$), and detect abnormal nodes in the t-th new time window.

4. Proposed framework

We propose Semi-supervised Anomaly Detection in dynamic Communication networks (SemiADC), to detect abnormal nodes with inadequate and inaccurate labels. To meet this goal, we first approximately model normal features with these labels (in Section 4.1). After that, we utilize the learned normal feature model (in Section 4.2.1) and corresponding temporal correlations (in Section 4.2.2) among selected flows (in Section 4.3) to iteratively clean the inaccurate labels and finally improve the anomaly detection performance in a self-learning way (in Section 4.4).

The overall structure of SemiADC is shown in Fig. 3. It consists of five major components: a generator G to generate synthetic normal features $G(\mathbf{Z})$ from random noise $\mathbf{Z}$, an encoder E to encode $\mathbf{X}_u$ into $E(\mathbf{X}_u)$ and $\mathbf{X}_l$ into $E(\mathbf{X}_l)$, a discriminator D to differentiate the synthetic tuple $(G(\mathbf{Z}_u), \mathbf{Z}_u)$ and the realistic tuple $(\mathbf{X}_u, E(\mathbf{X}_u))$, a classifier C to guide the generated tuple to be different from $(\mathbf{X}_l, E(\mathbf{X}_l))$, and a perceiver P to perceive the abnormalities based on the learned neural networks (G, E, D and C) and the graph $\hat{\varphi}$ with both labeled and potential abnormal nodes.

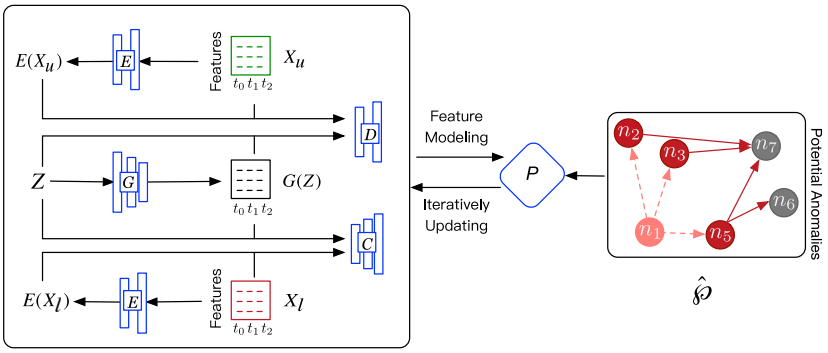


Fig. 3. An illustration of SemiADC with five components. The arrows show the input and output of each component. On the left, labeled, unlabeled and synthetic features ($\mathbf{X}_l, \mathbf{X}_u$ and $G(Z)$) are colored red, green and gray, respectively. On the right, labeled abnormal nodes and potential abnormal nodes are colored red and pink, and abnormal links in $\mathcal{S}$ and potential abnormal links in $\hat{\mathcal{S}}$ are represented as red lines and pink dashed lines separately.

Next, we will show how to design G, E, D and C for time-series feature modeling, and how to utilize P to clean $\mathbf{Y}$ and improve the learning results of G, E, D and C.

4.1. Learning the normal feature distribution

In order to clean the datasets and detect the abnormal nodes, we first model normal feature distribution utilizing both the approximate normal set $\mathcal{N}_u$ and the labeled abnormal $\mathcal{N}_l$ based on three observations. (1) Considering various and inadequate abnormal nodes, the adequate normal features can provide relatively unbiased guidance for anomaly detection based on facts that most abnormal features cannot well fit normal feature distribution [24,23,25]. (2) Real normal nodes in $\mathcal{N}_{u\_nrm}$ take the majority of the unlabeled set $\mathcal{N}_u$, thus the unlabeled data can be utilized to approximately learn the normal feature distribution and we also demonstrate it in Section 5. (3) Although the number of labeled abnormal is small, it is much easier to ensure the accuracy of labeled abnormal nodes and their features can provide help to regularize and bound the learning results of normal feature distribution.

Recently, GAN-based models have been proven to be very useful in estimating the data distribution and can generate "real" samples from a random "noise" [26]. Though GAN can approximate the normal data distribution and generate realistic data samples, it is difficult to find the proper $\mathbf{Z}$ to evaluate if a given feature matrix $\mathbf{X}^t_{n_t}$ fits the normal node distribution; while our ultimate goal of learning data distribution is to judge if a given sample is abnormal or not by estimating how well it fits the learned distribution. Thus, instead of using GAN, we adopt BiGAN [27], i.e., Bidirectional GAN. BiGAN introduces an encoder E, which eases the calculation of reconstruction error for a given input $\mathbf{X}^t_{n_t}$ (i.e., we do not need to take time to find the proper $\mathbf{Z}$ for the $\mathbf{X}^t_{n_t}$). A data sample that fits the learned normal feature distribution should have a small reconstruction error while an anomaly should have a large reconstruction error. Thus, BiGAN is good for our goal. Its objective function $\mathcal{V}(G, E, D)$ is

$$\min_{G,E}\max_{D}\mathbb{E}_{\mathbf{x}_u\sim p_{\mathcal{N}_u}}[\log D(\mathbf{X}_u, E(\mathbf{X}_u))] + \mathbb{E}_{\mathbf{Z}\sim p_z}[\log(1 - D(G(\mathbf{Z}), \mathbf{Z})] \tag{1}$$

where $p_{\mathcal{N}_u}$ is the feature distribution of the approximate normal nodes $\mathcal{N}_u$, and p_z is the distribution of random noise such as from Gaussian distribution. In Eq. (1), G, E are trained to minimize the objective function and make generated tuple $(G(\mathbf{Z}_u), \mathbf{Z}_u)$ close to realistic tuple $(\mathbf{X}_u, E(\mathbf{X}_u))$, whereas, the discriminator D is trained to maximize the objective function and distinguish between $(G(\mathbf{Z}_u), \mathbf{Z}_u)$ and $(\mathbf{X}_u, E(\mathbf{X}_u))$. Through this mini-max game, D can finally help G's generation to be close to the realistic data $\mathbf{X}_u$. With BiGAN, we can calculate the reconstruction error of an input $\mathbf{X}^t_{n_i}$ as $\|\mathbf{X}^t_{n_i} - G(E(\mathbf{X}^t_{n_i}))\|_F$, which can be used for estimating the abnormality.

For better anomaly detection performance, we further introduce a classifier C to ensure that the generated normal features from G are different from the existing labeled abnormal ones. Specifically, the objective $\mathcal{V}(G, E, C)$ can be formalized as

$$\max_{G,E,C}\mathbb{E}_{\mathbf{x}_l\sim p_{\mathcal{N}_l}}[\log C(\mathbf{X}_l, E(\mathbf{X}_l))] + \mathbb{E}_{\mathbf{Z}\sim p_z}[\log(1 - C(G(\mathbf{Z}), \mathbf{Z}))] + \mathbb{E}_{\mathbf{x}_u\sim p_{\mathcal{N}_u}}[\log(1 - C(\mathbf{X}_u, E(\mathbf{X}_u)))], \tag{2}$$

where $\mathbf{X}_l$ is a realistic abnormal feature matrix and it can be regarded as a sample from the feature distribution of the labeled abnormal nodes $\mathcal{N}_l$. The optimization of Eq. (2) is: (1) for abnormal features $\mathbf{X}_l \sim p_{\mathcal{N}_l}$, the classifier C should classify it as abnormal, i.e., $C(\mathbf{X}_l, E(\mathbf{X}_l) = 1$; while for normal data $G(\mathbf{Z})$ or $\mathbf{X}_u \sim p_{\mathcal{N}_u}, C$ should classify it as normal, i.e., $C(G(\mathbf{Z}), \mathbf{Z}) = 0$ or $C(\mathbf{X}_u, E(\mathbf{X}_u)) = 0$; and (2) the generator G should try to generate $G(\mathbf{Z})$ that will be classified as normal. With the guidance of C and $\mathcal{N}_l$, we can further ensure that the generator fits the distribution of realistic normal nodes and far away from existing labeled abnormal nodes, which would be useful to estimate the abnormality in Section 4.2.1.

Combining the Eqs. (1) and (2), we arrive at

$$\mathcal{V}(G,E,C,D) = \min_{G,E,C}\max_{D}(\alpha\mathcal{V}(G,E,D) - \beta\mathcal{V}(G,E,C)), \tag{3}$$

where α and β control the relative contributions. The loss functions for G, E, D and C are given as

$$\mathcal{L}_G = \alpha\mathbb{E}_{\mathbf{Z}_u\sim p_z}[\log(1 - D(G(\mathbf{Z}),\mathbf{Z}))] - \beta\mathbb{E}_{\mathbf{Z}\sim p_z}[\log(1 - C(G(\mathbf{Z}),\mathbf{Z}))] \tag{4}$$

$$\begin{aligned}\mathcal{L}_E = {} & \alpha\mathbb{E}_{\mathbf{X}_u\sim p_{\mathcal{N}_u}}[\log D(\mathbf{X}_u, E(\mathbf{X}_u))] - \beta\mathbb{E}_{\mathbf{X}_l\sim p_{\mathcal{N}_l}}[\log C(\mathbf{X}_l, E(\mathbf{X}_l))] \\ & -\beta\mathbb{E}_{\mathbf{X}_u\sim p_{\mathcal{N}_u}}[\log(1 - C(\mathbf{X}_u, E(\mathbf{X}_u)))]\end{aligned} \tag{5}$$

$$\mathcal{L}_D = -\alpha\mathbb{E}_{\mathbf{X}_u\sim p_{\mathcal{N}_u}}[\log D(\mathbf{X}_u, E(\mathbf{X}_u))] - \alpha\mathbb{E}_{\mathbf{Z}\sim p_z}[\log(1 - D(G(\mathbf{Z}),\mathbf{Z})] \tag{6}$$

$$\begin{aligned}\mathcal{L}_C = {} & -\beta\mathbb{E}_{\mathbf{X}_l\sim p_{\mathcal{N}_l}}[\log C(\mathbf{X}_l, E(\mathbf{X}_l))] - \beta\mathbb{E}_{\mathbf{Z}\sim p_z}[\log(1 - C(G(\mathbf{Z}),\mathbf{Z}))] \\ & -\beta\mathbb{E}_{\mathbf{X}_u\sim p_{\mathcal{N}_u}}[\log(1 - C(\mathbf{X}_u, E(\mathbf{X}_u)))].\end{aligned} \tag{7}$$

Considering node features evolve with time, we adopt Long Short-Term Memory (LSTM) [28] as the basic network for BiGAN, especially for C to capture the feature periodicity.

4.2. Exploring potential abnormal nodes

To clean the datasets and provide a more accurate model, the unlabeled abnormal nodes $\mathcal{N}_{u\_knw}$ and $\mathcal{N}_{u\_unk}$ should be detected from the approximate normal set $\mathcal{N}_u$ and moved into the labeled abnormal $\mathcal{N}_l$.

4.2.1. Feature-based abnormality evaluation

We first utilize the learned feature distribution to analyze the unlabeled data from $\mathcal{N}_u$. In order to filter out $\mathcal{N}_{u\_knw}$ and $\mathcal{N}_{u\_unk}$, we need to calculate the abnormality value $AnoS^t_{n_i}$ for each (n_i, t) in $\mathcal{N}_u$, where nodes in unlabeled abnormal sets $\mathcal{N}_{u\_knw}$ and $\mathcal{N}_{u\_unk}$ should receive higher value than unlabeled normal set $\mathcal{N}_{u\_nrm}$. As discussed in Section 4.1, we will use learned feature model to evaluate the abnormality $AnoS^t_{n_i}$ by

$$\begin{aligned}AnoS^t_{n_i} = {} & \gamma\|\mathbf{X}^t_{n_i} - G(E(\mathbf{X}^t_{n_i}))\|_F + \delta D(\mathbf{X}^t_{n_i}, E(\mathbf{X}^t_{n_i})) \\ & +(1-\gamma-\delta)C(\mathbf{X}^t_{n_i}, E(\mathbf{X}^t_{n_i})),\end{aligned} \tag{8}$$

where the first term $\|\mathbf{X}^t_{n_i} - G(E(\mathbf{X}^t_{n_i}))\|_F$ calculates reconstruction error compared to the generated normal features with the Frobenius norm, the term $D(\mathbf{X}^t_{n_i}, E(\mathbf{X}^t_{n_i}))$ denotes the discriminator's confidence of $\mathbf{X}^t_{n_i}$ sampled from reality, and $C(\mathbf{X}^t_{n_i}, E(\mathbf{X}^t_{n_i}))$ implies the feature similarities compared to the existing labeled abnormal nodes. γ and δ are scalars to control the relative contributions.

The proposed *AnoS* can meet the needs of different label conditions. If a node is largely different to the normal features (i.e., high $\|\mathbf{X}^t_{n_i} - G(E(\mathbf{X}^t_{n_i}))\|_F$), has a high possibility to be from real world (i.e., high $D(\mathbf{X}^t_{n_i}, E(\mathbf{X}^t_{n_i}))$), and is similar to the labeled abnormal features (i.e., high $C(\mathbf{X}^t_{n_i}, E(\mathbf{X}^t_{n_i}))$), it has a big chance to be an abnormal nodes and will receive a relatively high *AnoS*. To be specific, we have *three cases* for different label conditions:

- *Case* (1): From the perspective of time-series features, abnormal features are different from those from unlabeled normal $\mathcal{N}_{u\_nrm}$[22–24] and similar to those from labeled abnormal $\mathcal{N}_l$ [10,20]. Obviously, $\mathcal{N}_l$ can achieve higher *AnoS* than $\mathcal{N}_{u\_nrm}$.
- *Case* (2): For $\mathcal{N}_{u\_knw}$, as they are inaccurately trained as normal nodes, they will receive a similar value on the first two terms of *AnoS*. However, referred to those correctly labeled $\mathcal{N}_l$, they can get a high $C(\mathbf{X}^t_{n_i}, E(\mathbf{X}^t_{n_i}))$ value based on the same previously known attacks. Thus, *AnoS* for $\mathcal{N}_{u\_knw}$ will be higher than $\mathcal{N}_{u\_nrm}$.
- *Case* (3): There are similarities between different attacks, for example, the large hitters and heavy changers characteristics [13]. For $\mathcal{N}_{u\_unk}$, those unknown attacks also show feature similarities with the previously known ones [8,12]. Thus, the feature similarities will help provide relatively high $C(\mathbf{X}^t_{n_i}, E(\mathbf{X}^t_{n_i}))$ for $\mathcal{N}_{u\_unk}$, and *AnoS* for $\mathcal{N}_{u\_unk}$ will be higher than $\mathcal{N}_{u\_nrm}$.

To further validate the correctness of the *three case*, we conduct t-test using two datasets we introduce in Section 5.1.1. In detail, we construct vectors $\mathbf{a}_l, \mathbf{a}_{u\_knw}, \mathbf{a}_{u\_unk}$ and $\mathbf{a}_{u\_nrm}$ to denote the $AnoS^t_{n_i}$ of all label conditions (n_i, t) in $\mathcal{N}_l, \mathcal{N}_{u\_knw}, \mathcal{N}_{u\_unk}$ and $\mathcal{N}_{u\_nrm}$ separately. For t-test on two vectors $\{\mathbf{a}_l, \mathbf{a}_{u\_nrm}\}$ of *case* (1), the null hypothesis is $H_0 : \mathbf{a}_l \leqslant \mathbf{a}_{u\_nrm}$ while the alternative hypothesis is $H_1 : \mathbf{a}_l > \mathbf{a}_{u\_nrm}$, where the null hypothesis indicates that nodes with label conditions in $\mathcal{N}_l$ are likely to obtain smaller *AnoS* values than that in $\mathcal{N}_{u\_nrm}$; therefore if we reject the null hypothesis, the *case* (1) is verified. Based on the datasets DARPA1998 and CICIDS2017 of Section 5.1.1, the null hypothesis is rejected at the significant level $\alpha = 0.01$ with p-value of $5.02e^{-122}$ and $9.99e^{-200}$, respectively. For *case* (2) on the two datasets, the null hypothesis $H_0 : \mathbf{a}_{u\_knw} \leqslant \mathbf{a}_{u\_nrm}$ is rejected at the significant level $\alpha = 0.01$ with p-values of $1.39e^{-50}$ and $9.99e^{-200}$ when 10% abnormal nodes are inaccurately labeled in the two datasets for *case* (2). For *case* (3), as there are different attack types with different influences on the normal

feature distribution results, we train the normal feature distribution model for each attack types. The null hypothesis $H_0 : \mathbf{a}_{u\_unk} \leqslant \mathbf{a}_{u\_nrm}$ is rejected at the significant level $\alpha = 0.01$ among all the 14 attack types in the dataset CICIDS2017. However, H_0 can not be rejected with $\alpha = 0.01$ among all the 35 attack types in the DARPA1998. In summary, we verified the correctness of *case* (1) and (2), but for *case* (3), without knowledge of the unknown attacks, the *AnoS* of unlabeled abnormal $\mathcal{N}_{u\_unk}$ can be close to $\mathcal{N}_{u\_nrm}$, and there still exist some mistakes just depending on the feature similarities.

To further enhance the abnormality evaluation results with structure information in Section 4.2.2, we construct a relatively big potential abnormal node set $\mathcal{N}_\varphi$ with a loose criteria for label cleaning. In detail, we put those (n_i, t) in the history datasets into $\mathcal{N}_\varphi$ if $AnoS^t_{n_i}$ exceeds a threshold φ. The $\varphi = \mu + \eta\sigma$ can be defined by the Cantelli's Inequality [29,30], where μ and σ are the expected value and variance of the unlabeled nodes' *AnoS* scores, and η controls the false positive upper bound. Also, in the new time window, all nodes are unlabeled, and they can be regarded as nodes in $\mathcal{N}_u$. To detect anomalies in the dynamic communication networks, based on the *AnoS* scores of nodes in the t-th new time window, we construct the potential abnormal node set $\mathcal{N}_{\varphi,t}$, which only includes links in the t-th time window.

4.2.2. Structure-enhanced abnormality evaluation

To further refine the detection results, we exploit the temporal correlations of links between nodes. Since the processes of utilizing temporal correlations for label cleaning and anomaly detection are similar, we use label cleaning with the $\mathcal{N}_\varphi$ as an example. As links are changing frequently and there are more than one links in one time window, great amounts of redundant links will waste time and may influence the exploitation, we only utilize a few potential abnormal links to help enhance abnormality evaluation. For each $(n_i, t) \in \mathcal{N}_\varphi$, we select potential abnormal links in the t-th time window with source node n_i by the selection mechanism (to be introduced in Section 4.3), include them into potential abnormal link set $\hat{\mathcal{S}}$, and construct the graph $\hat{\wp}$ by $\hat{\mathcal{S}}$ and $\mathcal{S}$, where the link from n_i to n_j in t_k is denoted as $l^{t_k}_{n_i,n_j}$. In the constructed $\hat{\wp}$, we consider two popular temporal correlations between links (as shown in Fig. 4), i.e., multiple steps and multiple hops [31–33]. We only explain the correlations between two links here for simplification although they can actually happen among more links. For multi-hop attacks in Fig. 4(a), to increase the success rate of attacking n_5, attacker n_7 launches several attacks with multiple hops to control n_6, n_3 and n_4, and make them simultaneously launch attacks on n_5. For multi-step attacks in Fig. 4(b), attacker n_1 may first launch several attacks to test the vulnerability of the victim n_4 before the disastrous attack happens. Obviously, if one of the two links is in the labeled nodes' link set $\mathcal{S}$, it would help the other link and the corresponding abnormal node being recognized. For example, the labeled abnormal (n_2, t_6) will help detect (n_6, t_7) by the multi-hop correlation between $l^{t_6}_{n_2,n_6}$ and $l^{t_7}_{n_6,n_5}$. Similarly, the labeled abnormal (n_1, t_1) will help detect (n_1, t_3) by the multi-step correlation between $l^{t_1}_{n_1,n_4}$ and $l^{t_3}_{n_1,n_4}$. Note that it is necessary to identify the same abnormal nodes in different time window, such as (n_1, t_1) and (n_1, t_3) in Fig. 4(b), which is because although what we want is to find the abnormal nodes, yet in the training sets we would like to identify each $\mathbf{Y}^t_{n_i}$ to prepare for testing nodes in new time windows.

As the temporal correlation is among links of at least two abnormal nodes, it helps recognize the real anomalies $\mathcal{N}_{u\_knw}$ and $\mathcal{N}_{u\_unk}$ from the potential abnormal node set $\mathcal{N}_\varphi$. In detail, we have *three cases* for different temporal correlations:

- *Case* (1): for $(n_6, t_7) \in \mathcal{N}_\varphi$ in Fig. 4(a), there is *a path* from the labeled abnormal n_2 to the potential abnormal n_6;
- *Case* (2) for $(n_7, t_5) \in \mathcal{N}_\varphi$, there are three *reversed paths* from the labeled n_2, n_3 and n_4 to the potential abnormal n_7;
- *Case* (3) for $(n_1, t_3) \in \mathcal{N}_\varphi$ in Fig. 4(b), we can regard it as *a path to itself* based on the correlation between $l^{t_1}_{n_1,n_4}$ and $l^{t_3}_{n_1,n_4}$.

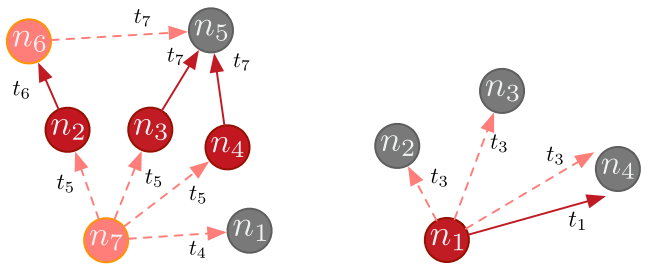
In summary, the temporal correlations provide an undirected path starting from the labeled abnormal nodes to the unlabeled abnormal nodes.

The undirected paths with temporal correlations can be treated as a diffusion process among abnormal nodes, where abnormality is transferred from one node to the other linked one [34]. After adding all links of $\mathcal{S}$ into $\hat{\wp}$, we conduct random walks for each potential abnormal $(n_j, t) \in \mathcal{N}_\varphi$. The probability of walking from the labeled abnormal n_i to the potential abnormal n_j is based on the relative abnormality, which can be formulated as

$$Tr_{ij} = \frac{AnoS^t_{n_j}}{\sum_{(n_k,t_\eta)\in\mathcal{R}} AnoS^{t_\eta}_{n_k}}. \tag{9}$$

Here, $\mathcal{R}$ contains the potential abnormal nodes (n_k, t_η) which has an undirected path from n_k to the labeled n_i like the above *three cases*, and $(n_j, t) \in \mathcal{R}$. Since the diffusion process can help recognize the real anomalies $\mathcal{N}_{u\_knw}$ and $\mathcal{N}_{u\_unk}$ from the potential abnormal node set $\mathcal{N}_\varphi$, we update *AnoS* to *AnoST* as

$$AnoST^t_{n_j} = AnoS^t_{n_j} + \tau \sum_{n_i\in\mathcal{N}_l} \frac{NUM_{ij}}{NUM_i}, \tag{10}$$



(a) Multi-hop attacks (b) Multi-step attacks

Fig. 4. Subgraphs of the $\hat{\varphi}$ showing temporal correlations, where the red links and nodes are from $\mathcal{S}$, and the dashed pink links are from $\hat{\mathcal{S}}$. The pink and gray nodes are from $\mathcal{N}_\varphi$, denoting the real abnormal and the mistakenly-included normal nodes separately.

where the second term represents the estimation of temporal correlations between links of the potential abnormal $n_j \in \mathcal{N}_\varphi$ and the labeled abnormal $n_i \in \mathcal{N}_l$. Considering the diffusion process with the *three cases* for different temporal correlations, nodes have higher possibilities to be abnormal if they have higher possibilities to be easily reached from labeled abnormal nodes. Similar to our scenario with abnormality, this observation for diffusion model is widely used in social recommendation, where preferences can be transferred from one user to the other linked one (i.e., friend), and users have more possibility to like one item if they can be easily reached from those who have the same preference [35]. Thus, starting from labeled abnormal nodes in $\mathcal{N}_l$, if we arrive the node n_j many times (i.e., n_j has many temporal correlations with nodes in $\mathcal{N}_l$), n_j has a higher possibility to be abnormal.

In detail, for each node n_i in $\mathcal{N}_l$, there are m_1 times of m_2 steps random walk. The number of random walk is m_1, the maximum step of each random walk is m_2, thus each labeled abnormal n_i will pass at most $m_1 \times m_2$ nodes in the random walk process. As m_2 is the maximum step and there may no enough links of n_i, the actual walking steps from n_i is NUM_i rather than $m_1 \times m_2$. Among those NUM_i steps, n_j has been passed NUM_{ij} times. We set $Y^t_{n_j} = 1$ for those who receive top K $AnoST^t_{n_j}$, and add (n_j, t) into $\mathcal{N}_l$. With the updated dataset, we can retrain the model in Section 4.1 with a more precise input dataset. As we have η to control false positive upper bound and limit the number of potential abnormal nodes in $\mathcal{N}_\varphi$, we will only return $|\mathcal{N}_\varphi|$ nodes and stop the cleaning process if the $K < |\mathcal{N}_\varphi|$. After the cleaning process finishes, with the $\mathcal{N}_{\varphi,t}$ in the new time window, we can filter the potential abnormal link set $\hat{\mathcal{S}}_t$ from $\mathcal{A}_t$ based on the trained attention model (Section 4.3). The temporal correlations between links of $\hat{\mathcal{S}}_t$ and $\mathcal{S}$ on the graph $\hat{\varphi}_t$ will also help enhance the correctness of abnormality evaluation for anomaly detection.

4.3. Attention for time slots

In each time window, there are more than one time slot, which have different contributions to the abnormal feature modeling since abnormal nodes may not behave abnormally through the whole time window. In addition to the feature modeling, the time slots' different importance also affects structure-based exploration. Temporal correlations only happen among links of several time slots other than links of the whole time window. These useless links, for example, the links $l^{t_3}_{n_1,n_2}$ and $l^{t_3}_{n_1,n_3}$ in Fig. 4(b), will enlarge the size of $\mathcal{R}$ for Eq. (9) and highly increase the computation complexity for the random walk on $\hat{\varphi}$.

To capture the different importance of time slots in each time window and reduce the computation complexity, we adopt the attention mechanism [36,37]. Specifically, the hidden state $\{\mathbf{h}_{t,i}\}_{i=1}^{w}$ of the first layer of classifier C is modified with attention weights $\alpha_{t,i}$ and aggregated to form the latent vector $\mathbf{h}_t$ for each time window,

$$\begin{aligned} u_{t,i} &= tanh(\mathbf{W}_s \mathbf{h}_{t,i} + \mathbf{b}_s), \\ \alpha_{t_i} &= \frac{exp(\mathbf{u}^T_{t,i}\mathbf{u}_s)}{\sum_i exp(\mathbf{u}^T_{t,i}\mathbf{u}_s)}, \quad \mathbf{h}_t = \sum_i \alpha_{t,i}\mathbf{h}_{t,i}, \end{aligned} \tag{11}$$

where we feed the slot i-th hidden state $\mathbf{h}_{t,i}$ through a MLP to get $\mathbf{u}_{t,i}$ as a hidden representation of $\mathbf{h}_{t,i}$. Also, the slot importance $\alpha_{t,i}$ is measured as the similarities between $u_{t,i}$ and a slot level context vector u_s (which is randomly initialized and jointly learned during the training process like [36]). The slot importance $\alpha_{t,i}$ implies each slot's local relative contribution, while the final representation $\mathbf{h}_t$ will help globally estimate nodes' abnormality for C. In the structure-based random walk, we can only include links of the time slot with the highest $\alpha_{t,i}$ into $\hat{\mathcal{S}}$, which will largely reduce the number of involved links and reduce the computation complexity.

Algorithm 1: Label Cleaning

Input: $\mathbf{X}_l$, $\mathbf{X}_u$ $\mathcal{N}_l$, $\mathcal{N}_u$, $\mathcal{S}$

1. **for** number of training iterations **do**
2. **while** not converge **do**
3. Randomly sample mini-batch from $\mathbf{X}_u$ and $\mathbf{X}_l$
4. Randomly sample noise $\mathbf{Z}$ for latent normal features
5. Obtain generated $G(\mathbf{Z})$, encoded $E(\mathbf{X}_u)$ and $E(\mathbf{X}_l)$
6. Update discriminator parameters θ_D by Eq. (6)
7. Update classifier parameters θ_C to minimize Eq. (7) and calculate attention parameters α by Eq. (11)
8. Update generator parameters θ_G to minimize Eq. (4)
9. Update encoder parameters θ_E to minimize Eq. (5)
10. Add label condition into $\mathcal{N}_\varphi$ by Eq. (8)
11. **for** (n_j, t) in $\mathcal{N}_\varphi$ **do**
12. Add its links in the highest $\alpha_{t,i}$ slot to $\hat{\mathcal{S}}$
13. Add $\mathcal{S}$, $\hat{\mathcal{S}}$ into $\hat{\wp}$
14. **for** n_i in $\mathcal{N}_l$ **do**
15. Random walk from n_i with Eq. (9) in $\hat{\wp}$
16. Add nodes with top K *AnoST* into $\mathcal{N}_l$ by Eq. (10)
17. Update $\mathcal{N}_u$, $\mathbf{X}_l$ and $\mathbf{X}_u$ based on updated $\mathcal{N}_l$

4.4. Proposed framework – SemiADC

With the introduction of modeling normal features and exploring potential abnormal nodes, we first iteratively train our model to clean the labels in a semi-supervised way. The details are shown in Algorithm 1. We first train the GAN-based model from line 2 to line 9 with $\mathbf{X}_l$ and $\mathbf{X}_u$. After convergence, we select the potential abnormal nodes $\mathcal{N}_\varphi$ with the most contributive links $\hat{\mathcal{S}}$ from line 10 to line 12. After including abnormal links $\mathcal{S}$ and potential abnormal links $\hat{\mathcal{S}}$ to graph $\hat{\wp}$ in line 13, we calculate the arrived times for each potential abnormal nodes from line 14 to line 15, add the most potential abnormal nodes into $\mathcal{N}_l$ in line 16, and retrain the normal feature model from line 2 to 17 with the updated $\mathbf{X}_l$ and $\mathbf{X}_u$. After several training iterations, we can get a more precise normal feature model.

Algorithm 2: The Proposed Framework SemiADC

Input: $\mathbf{X}_l$, $\mathbf{X}_u$ $\mathcal{N}_l$, $\mathcal{N}_u$, $\mathcal{S}$, $\hat{\mathcal{S}}_t$

1. Conduct label cleaning by Algorithm 1
2. # In the new time window t
3. Calculate the *AnoS* and construct $\mathcal{N}_{\varphi,t}$ by Eq. (8) with trained model
4. **for** (n_j, t) in $\mathcal{N}_{\varphi,t}$ **do**
5. Add its links in the highest $\alpha_{t,i}$ slot to $\hat{\mathcal{S}}_t$
6. Add $\mathcal{S}$, $\hat{\mathcal{S}}_t$ into $\hat{\wp}_t$
7. **for** n_i in $\mathcal{N}_l$ **do**
8. Random walk from n_i with Eq. (9) in $\hat{\wp}_t$
9. Detect abnormal nodes with *AnoST* by Eq. (10)

As a deep learning model, it takes a relatively long time to train the model, but the process of anomaly detection is fast and timely for the data in new time windows (or in the testing sets). The details are shown in Algorithm 2. To detect abnormal nodes in the t-th new time window, we first calculate a rough abnormality value *AnoS* for each testing node and construct $\mathcal{N}_{\varphi,t}$ in line 3. Then we utilize the trained attention model to select the potential abnormal links $\hat{\mathcal{S}}_t$ from line 4 to line 5, refine the abnormality estimation with temporal correlations from line 6 to line 8, and get the abnormal nodes by calculating the refined abnormality value *AnoST* in line 9.

Let fully connected neural networks E, G, D and C contain N_E, N_G, N_D, N_C layers with $d_{E_i}, d_{G_i}, d_{C_i}, d_{D_i}$ hidden units, and the first layer of C for $\mathbf{X}$ is LSTM with d_l hidden units. For N testing samples, the computation cost of *AnoS* is $O(N(md_{E_1} + \sum_{n=2}^{N_E} d_{E_{i-1}} d_{E_i} + d_{E_{N_E}} d_{G_1} + \sum_{n=2}^{N_G} d_{G_{i-1}} d_{G_i}))$ for the first term, $O(N((m + d_{E_{N_E}}) d_{D_1} + \sum_{n=2}^{N_D} d_{D_{i-1}} d_{D_i}))$ for the second term, and $O(N((m + w + d_{C_1} + d_{E_{N_E}}) d_l + \sum_{n=2}^{N_C} d_{C_{i-1}} d_{C_i}))$ for the third with consideration of the attention mechanism. With the $\mathcal{N}_\varphi$ based on *AnoS*, the cost of calculating attention weight for $\hat{\mathcal{S}}$ selection is $O(|\mathcal{N}_\varphi| \times w)$. Finally, the cost of transition probabil-

ity and random walk is $O(|\hat{\mathcal{S}}| + |\mathcal{S}|m_1m_2)$. Note that in our experiments, N_E, N_G, N_D, N_C are less than 3, $d_{E_i}, d_{G_i}, d_{C_i}, d_{D_i}$ are less than 20, and they are fixed. Thus, the time complexity is not large. Specially, we run the experiments on a Ubuntu PC with i7-7800X CPU, 16GB memory, and one 1080ti GPU, making the experimental details the same as that in Section 5.1 the testing time for each (n_i, t) is about 10^{-5} second.

5. Experimental analysis

In this section, we conduct experiments to demonstrate the effectiveness of our proposed framework SemiADC. Through the experiments, we aim to answer four questions: (1) Can SemiADC improve the anomaly detection performance under inaccurate labels from unlabeled abnormal sets $\mathcal{N}_{u\_knw}$ and $\mathcal{N}_{u\_unk}$? (2) How robust is SemiADC under different percentages of inaccurate labels? (3) How the self-learning process affects the SemiADC in training data cleaning and anomaly detection? and (4) How hyper-parameters (e.g., α and τ) affect the anomaly detection performance?

Next, we will first introduce the experiment settings followed by experiments to answer these four questions.

5.1. Experimental settings

5.1.1. Datasets

Two publicly available datasets DARPA1998[1] and CICIDS2017[2] are used for evaluation, where DARPA1998 is widely used for anomaly detection in communication networks [1,17,20] and CICIDS2017 is similar to DARPA1998 but with different and new attack types. DARPA1998 contains 3013862 IP-IP network flows with their corresponding real-world data packages for seven weeks, including 9008 source IPs, 21022 destination IPs and 35 attack types. Similarly, CICIDS2017 is composed of 2830743 IP-IP network flows with their corresponding real-world data packages for one week, including 17005 source IPs, 19112 destination IPs and 14 attack types.

We adopt some pre-processing steps for these two datasets. Following the guidance[3], we first extract 41 features for each data package, including continuous features (like start time, duration) and discontinuous features (like protocols and services). For those discontinuous ones, we use one-hot vectors to represent these features [24], which results in 56 features for the DARPA98 and 98 features for CICIDS2017. For both datasets, each time window contains 10 time slots. Considering the attack frequency, the time slot size is set to be 10 min for DARPA1998 and 6 s for CICIDS2017. Regarding (IP, time) as the unique identifier, there are $|\mathcal{N}_l| = 27309$ and $|\mathcal{N}_{u\_nrm}| = 8297$ for DARPA1998; and $|\mathcal{N}_l| = 1006$ and $|\mathcal{N}_{u\_nrm}| = 189965$ for CICIDS2017. As to the value of each feature in each time slot: if it is a continuous feature, the value is the average of the corresponding feature values (e.g., average flow duration in a time slot); if it is a one-hot encoding feature, the value is the occurrence time (e.g., the frequency of using UDP protocol in a time slot).

Note that, although one of the attacks "smurf" in DARPA1998 lasts for a long time and construct huge amounts of time windows, leading to a large $\mathcal{N}_l$, yet other attacks except "smurf" only occupy 2% of the whole datasets, and most of these attacks appear in less than 10 time windows, thus it is still consistent to our inadequate abnormal label settings.

In the following experiments, we use 60% of both $|\mathcal{N}_l|$ and $|\mathcal{N}_{u\_nrm}|$ for training, leaving 40% for testing, where we ensure the interactions of the training dataset happen before the testing dataset. Also, as there are no exact unlabeled abnormal $\mathcal{N}_{u\_knw}$ and $\mathcal{N}_{u\_unk}$ as ground truth, we will construct $\mathcal{N}_{u\_knw}$ and $\mathcal{N}_{u\_unk}$ in Section 5.2.

5.1.2. Compared methods

To evaluate the effectiveness of SemiADC, we compare SemiADC with representative and state-of-the-art approaches:

- REPEN [30]: this method learns customized REPresentations for a random nEarest Neighbor distance-based approach, ensuring the representations of outliers can get higher scores with prior knowledge of outliers' features.
- GANomaly [22]: this method designs an encoder-decoder-encoder pipeline for GAN, and evaluate the abnormality of each node based on the normal features.
- GAN-AD [23]: this method utilizes RNN and GAN to learn time-series features of normal nodes and get the abnormality estimation for each node.
- ALAD [24]: the Adversarially Learned Anomaly Detection (ALAD) method designs a BiGAN-based framework for learning closer $\mathbf{X}$ and $G(E(\mathbf{X}))$ efficiently and the abnormality estimation is just based on normal features.
- ADC: this is a variant of the proposed model, where we only use *AnoS* as an abnormality estimation without the self-learning process.

As some of these baselines cannot deal with two-dimensional features, i.e., $\mathbf{X}^t_{n_i}$ is a matrix, we utilize PCA to reduce features into 1×10 to provide a fair comparison with the same input while maintaining the time-series characteristics. For each approach, the parameters are tuned via cross-validation on training data. We set $\alpha = 0.8$ and $\beta = 0.2$ to control the relative

[1] https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset
[2] https://www.unb.ca/cic/datasets/ids-2017.html
[3] http://kdd.ics.uci.edu/databases/kddcup99/

contribution of normal and abnormal features, and the $\gamma, \delta, 1-\gamma-\delta$ are set to be 0.8, 0.1, and 0.1 for three terms of *AnoS* respectively. The final *AnoST* is obtained with $\tau = 1$. More detailed discussion on parameter selection will be presented in Section 5.5. Every experiment is conducted five times and the average results are reported.

5.1.3. Implementation details

For SemiADC, the model architectures for both datasets are the same. As SemiADC has four neural network components, i.e., G, D, E and C, we present the details of each component separately. (1) For generator, we use a three-layer MLP with the number of hidden units in each layer as 16, 12 and 10, respectively. (2) For the encoder, we adopt a two-layer MLP with 12 hidden units and 16 hidden units, respectively. The input dimension and output dimension of the whole encoder are 10 and 16. (3) For discriminator, the first 12-unit dense layer separately handles the two inputs, i.e., the latent vector **Z** and the input **X**, and then concatenate them as input to the second 16-unit dense layer. (4) The classifier contains two inputs. For input **Z**, the first layer is the same to the discriminator. For input **X**, a two-unit LSTM layer is used with attention mechanism and the output of attention mechanism is put into a 12-unit dense layer. After that, we concatenate the output for **Z** and **X** and put them into the dense layer with 16 units. To optimize the deep network, Adam algorithm is adopted with the learning rate as 0.01 and momentum decay as 0.5.

5.1.4. Evaluation metrics

We utilize four popular evaluation metrics *precision*, *recall*, $F1$ and *AUC* (Area under ROC Curve) to evaluate the performance of anomaly detection, which lie in $[0, 1]$ and a higher value means better performance. As the outputs of the above algorithms are abnormality estimation, we take the top M as abnormal. To calculate the aforementioned *precision*, *recall* and $F1$, M is set to be the number of abnormal instances in the testing sets for both datasets.

5.2. Performance comparison of anomaly detection

To meet the main targets of our experiments and answer the *first question*, we will evaluate the *precision*, *recall*, $F1$ and *AUC* on the testing sets. As there are two reasons leading to inaccurate labels, we have two kinds of label condition, i.e., $\mathcal{N}_{u\_knw}$ with *known* attacks, and $\mathcal{N}_{u\_unk}$ with *unknown* attacks. Since their settings of training sets are totally different, we will evaluate their anomaly detection performance separately in Section 5.2.1 and 5.2.2. Besides, we will provide a real example in Section 5.2.3 with explainable results.

5.2.1. Anomaly detection under inaccurate labels from $\mathcal{N}_{u\_knw}$

We first conduct the experiments when $\mathcal{N}_{u\_knw} \neq \varnothing$ and $\mathcal{N}_{u\_unk} = \varnothing$. To simulate the settings of $\mathcal{N}_{u\_knw}$, we randomly select 10% of each attack in $\mathcal{N}_l$ from the training sets, and replace their labels as normal. For SemiADC, we conduct 6 times of self-learning iterations and update K label conditions in each iteration (K is set to be 300 for DARPA1998 and 20 for CICIDS2017 based on their numbers of abnormal nodes). The experiment is conducted five times and the average anomaly detection performances with standard deviation are reported in Table 1 and Table 2. From the table, we observe:

- By utilizing abnormal nodes, our ADC can outperform GANomaly and GAN-AD, which is because GANomaly and GAN-AD fully depend on the inaccurate labeled normal nodes, while ADC can handle this situation based on the regularization of labeled abnormal $\mathcal{N}_l$.
- With only normal nodes, ALAD has similar performance to ADC for its optimized BiGAN-based framework.
- Our SemiADC achieves substantial improvements than other baseline methods in both datasets. Specially, from the comparison to the ADC, the great performance increase should owe to both temporal correlations and the self-learning process.

Based on the aforementioned observations, we conclude that SemiADC outperforms the state-of-the-art anomaly detection approaches under inaccurate labels from $\mathcal{N}_{u\_knw}$.

5.2.2. Anomaly detection under inaccurate labels from $\mathcal{N}_{u\_unk}$

We conduct the experiments of $\mathcal{N}_{u\_unk} \neq \varnothing$ and $\mathcal{N}_{u\_knw} = \varnothing$. To simulate the settings of zero-day vulnerability, we extract time windows containing "Warezclient" and "Bot" attacks from DARPA1998 and CICIDS2017 separately, and replace all their labels as normal in the training sets. The average results of anomaly detection on the testing sets are shown in Table 3 and Table 4, from which we can observe:

- Compared to inaccurate labels from $\mathcal{N}_{u\_knw}$ with unknown attacks, most of the results decrease, which shows there are more difficulties to tackle unknown attacks than the known ones.
- ADC is overpassed by some of the baseline methods. It is because nodes with zero-day attacks are mislabeled as normal, which contaminate both abnormal and normal datasets. ADC has bad effects from both abnormal nodes and normal nodes, while the other baselines are only affected by the normal nodes.

Table 1
Anomaly detection performance comparison under inaccurate labels from $\mathcal{N}_{u\_knw}$ with known attacks in DARPA1998.

Methods	DARPA1998			
	precision	recall	F1	AUC
REPEN	0.8849 ± 0.1184	0.9267 ± 0.0408	0.9053 ± 0.0385	0.8410 ± 0.0337
GANormaly	0.8792 ± 0.0655	0.9262 ± 0.0567	0.9021 ± 0.0551	0.7943 ± 0.1011
GAN-AD	0.9132 ± 0.0311	0.9188 ± 0.0375	0.9160 ± 0.0339	0.8453 ± 0.0022
ALAD	0.9116 ± 0.0137	0.9343 ± 0.0202	0.9231 ± 0.0073	0.8397 ± 0.023
ADC	0.9168 ± 0.0283	0.9279 ± 0.0394	0.9223 ± 0.0338	0.8673 ± 0.0673
SemiADC	**0.9595 ± 0.0029**	**0.9603 ± 0.0027**	**0.9599 ± 0.0027**	**0.8938 ± 0.0176**

Table 2
Anomaly detection performance comparison under inaccurate labels from $\mathcal{N}_{u\_knw}$ with known attacks in CICIDS2017.

Methods	CICIDS2017			
	precision	recall	F1	AUC
REPEN	0.8509 ± 0.0242	0.8617 ± 0.0310	0.8563 ± 0.0275	0.8995 ± 0.0028
GANormaly	0.8622 ± 0.0643	0.8842 ± 0.0265	0.8731 ± 0.0416	0.7786 ± 0.0644
GAN-AD	0.8667 ± 0.0643	0.8918 ± 0.0506	0.8791 ± 0.0554	0.9439 ± 0.0443
ALAD	0.8885 ± 0.0308	0.8990 ± 0.0203	0.8951 ± 0.0310	0.9786 ± 0.0238
ADC	0.8821 ± 0.0517	0.9231 ± 0.0769	0.9021 ± 0.0540	0.9847 ± 0.0050
SemiADC	**0.9387 ± 0.0132**	**0.9402 ± 0.0146**	**0.9394 ± 0.0139**	**0.9921 ± 0.0222**

Table 3
Anomaly detection performance comparison under inaccurate labels from $\mathcal{N}_{u\_unk}$ with unknown attacks in DARPA1998.

Methods	DARPA1998			
	precision	recall	F1	AUC
REPEN	0.8857 ± 0.1208	0.9275 ± 0.0615	0.9061 ± 0.0435	0.8728 ± 0.0075
GANormaly	0.8325 ± 0.0342	0.9560 ± 0.0260	0.8900 ± 0.0309	0.7745 ± 0.2402
GAN-AD	0.9117 ± 0.0290	0.9251 ± 0.0424	0.9183 ± 0.0357	0.8706 ± 0.0400
ALAD	0.9045 ± 0.0266	0.9101 ± 0.0481	0.9094 ± 0.0354	0.8371 ± 0.0303
ADC	0.9055 ± 0.0236	0.8976 ± 0.0249	0.9015 ± 0.0243	0.8815 ± 0.0276
SemiADC	**0.9530 ± 0.0053**	**0.9556 ± 0.0032**	**0.9543 ± 0.0040**	**0.8909 ± 0.0230**

Table 4
Anomaly detection performance comparison under inaccurate labels from $\mathcal{N}_{u\_unk}$ with unknown attacks in CICIDS2017.

Methods	CICIDS2017			
	precision	recall	F1	AUC
REPEN	0.7155 ± 0.0031	0.7642 ± 0.0207	0.7390 ± 0.0095	0.9180 ± 0.0691
GANormaly	0.8744 ± 0.0810	0.8606 ± 0.1179	0.8674 ± 0.1382	0.8965 ± 0.0019
GAN-AD	0.8721 ± 0.0044	0.8641 ± 0.0043	0.8681 ± 0.0043	0.9450 ± 0.0431
ALAD	0.8670 ± 0.0083	0.8799 ± 0.0084	0.8734 ± 0.0084	0.9630 ± 0.0692
ADC	0.8865 ± 0.0208	0.8948 ± 0.0146	0.8907 ± 0.0169	0.9935 ± 0.0052
SemiADC	**0.9394 ± 0.035**	**0.9375 ± 0.058**	**0.9385 ± 0.0045**	**0.9941 ± 0.0207**

- With the utilization of temporal correlations and self-learning, SemiADC makes up to the deficiencies of ADC, and keep the substantial improvement than other baselines in both datasets.

Based on the observations of Section 5.2.1 and 5.2.2, we can answer the *first question* that SemiADC can outperform the state-of-the-art anomaly detection approaches when facing inaccurate labels from both $\mathcal{N}_{u\_knw}$ and $\mathcal{N}_{u\_unk}$.

5.2.3. Case study

To help understand the performance of our SemiADC in dynamic networks, we conduct a case study in DARPA1998 under inaccurate labels from $\mathcal{N}_{u\_knw}$, moving 10% $\mathcal{N}_l$ in the training set to $\mathcal{N}_{u\_knw}$ like Section 5.2.1. Since the interactions of the training dataset happen before the testing dataset, the abnormality evaluation on the testing dataset can be regarded as anomaly detection in the new time windows.

For the label cleaning performance in the training set, we can find an example from those who receive high *AnoST*, whose IP is "135.013.216.191", and the time window is from "06/09/1998 19:00:31" to "06/09/1998 20:40:31". It is labeled as normal at first and updated to be abnormal by SemiADC. To verify its correctness, we check the original network flows and data

packages, and find that this "135.013.216.191" does launch 930 times of "ipsweep" attack from "06/09/1998 19:00:31" to "06/09/1998 20:32:54". This explainable case proves the effectiveness on label cleaning.

Additionally, for the anomaly detection performance in the testing set, we look into an example from those who receive high *AnoST*, whose IP is "199.174.194.016", and the time window is from "07/16/1998 17:46:58" to "07/16/1998 19:26:58". It is predicted as abnormal by SemiADC. We further exam the attention score. We find that the first time slot (the first 10 min) receives the highest attention weight, which means that SemiADC detects that attacks may happen in this time window. To verify if this is correct, we check the original network flows and data packages. We find that this "199.174.194.016" does launch a "smurf" attack at "07/16/1998 17:46:58" and the corresponding flow lasts 1 min and 45 s. Although there are inaccurate labels in the training set, we can still accurately detect abnormal nodes in the testing set (i.e., in new time windows). This explainable case proves the effectiveness on anomaly detection in dynamic networks.

5.3. Sensitivity to the inaccurate labels

To evaluate the capacity to tackle inaccurate labels of different percentage and answer the *second question*, we randomly select x% of $\mathcal{N}_l$ and mislabel them as "normal" like Section 5.2.3, where x varies in $\{0, 10, \ldots, 50\}$. The results in both datasets are shown in Fig. 5 and Fig. 6, from which we can observe that, more inaccurate labels can bring worse effects on anomaly detection, thus both ADC and SemiADC decrease with the x%. Besides, SemiADC decreases slower than ADC, and keeps $F1$ and *AUC* to be higher than 0.85 and 0.8 in DARPA1998 even when half of abnormal nodes are labeled wrong, which proves the robustness of our model and answers the second question.

5.4. Influence of the self-learning process

The self-learning process can be used not only to improve cleaning performance in training data but also to refine abnormal node detection in the testing set. To answer the *third question* and evaluate the influence of the self-learning process, we will iteratively update K labels for 10 times based on *AnoST* value, where $K = 300$ for DARPA1998 and $K = 20$ for CICIDS2017. In each iteration, we will calculate the $F1$ and *AUC* to evaluate both the data cleaning performance in the training set and the anomaly detection performance in the testing set. Besides, to provide enough inaccurate labels, we set x to be 10 for DARPA1998 and 20 for CICIDS2017 based on the number of abnormal nodes.

After 10 times of self-learning iterations, we obtain the results in Fig. 7 and Fig. 8. We can observe that, with the increase of iteration times, cleaning performance on the training set first increases, suggesting that we do find the real abnormal nodes from the candidate abnormal, and correct their inaccurate labels. After the iteration times reach a certain point, the cleaning performance decreases because almost all abnormal nodes have been detected and cleaned in previous iterations, and no more abnormal nodes can be found. Similarly, the anomaly detection performance on the testing set first increases then decreases for the same reason.

5.5. Parameter analysis

There are two important pre-defined hyper-parameters, i.e., α and τ, controling the relative contribution of normal and abnormal feature, and the relative contribution of feature and structure modeling, respectively. To answer the *fourth question*, we conduct the anomaly detection with different parameters. The results on *AUC* are shown in Fig. 9 and Fig. 10.

We first evaluate the anomaly detection performance under different α. The value of α changes from 0.1 to 1, showing the increasing contribution of normal features. Accordingly, the value of β changes from 0.9 to 0, indicating the decreasing contribution of abnormal features. From Fig. 9, we can observe that the anomaly detection first increases with α and then decreases, which is good for parameter selection. Also, when $\alpha = 1$ and $\beta = 0$ (i.e., no contributions from negative features), we can achieve a high AUC results, showing that although both normal and abnormal features can provide help, normal features can have more impacts.

To evaluate the anomaly detection results under different τ, we change the value of τ from 0.1 to 1, indicating the increasing contribution of the structure-enhanced model. From Fig. 10, we can observe that the anomaly detection performance keeps increasing with τ, demonstrating the importance of utilizing structure-based temporal correlations.

6. Conclusions

In this paper, we investigate a novel problem of anomaly detection in communication networks with inadequate and inaccurate labels. We propose a new semi-supervised anomaly detection framework SemiADC based on GAN and self-learning to tackle the inadequacy and inaccuracy problems for dynamic communication networks. The proposed framework can generate latent normal features with regularization from existing abnormal ones, utilize self-training to clean the training set, and novelly improve the anomaly detection based on time-series feature similarities and structure-based temporal correlations. Experimental results on real-world datasets demonstrate the effectiveness and robustness of the proposed SemiADC, which significantly outperforms the state-of-the-art methods.

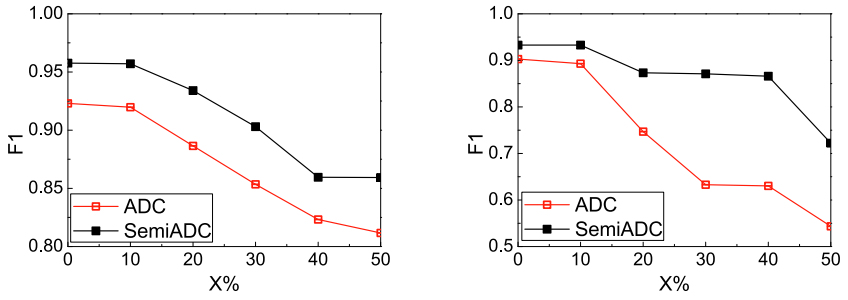


Fig. 5. Sensitivity to the *x*% in two datasets on *F*1.

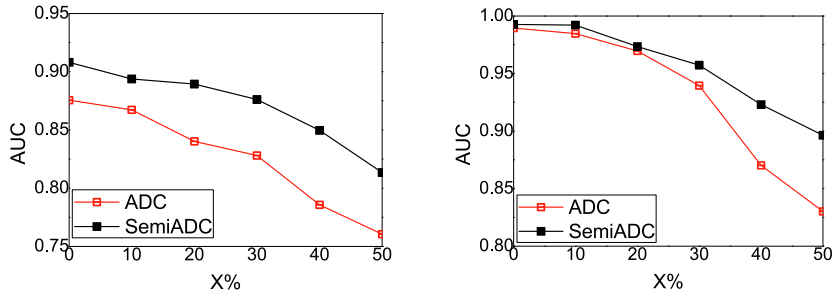


Fig. 6. Sensitivity to the *x*% in two datasets on *AUC*.

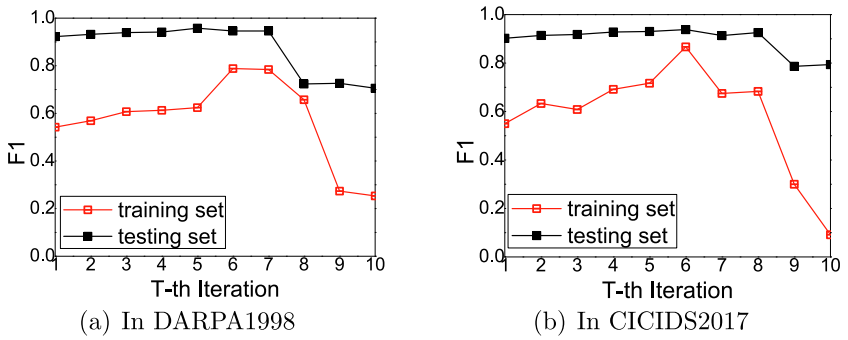


Fig. 7. Data cleaning performance in the training set and anomaly detection performance in the testing set on *F*1 with 10 times self-learning iterations.

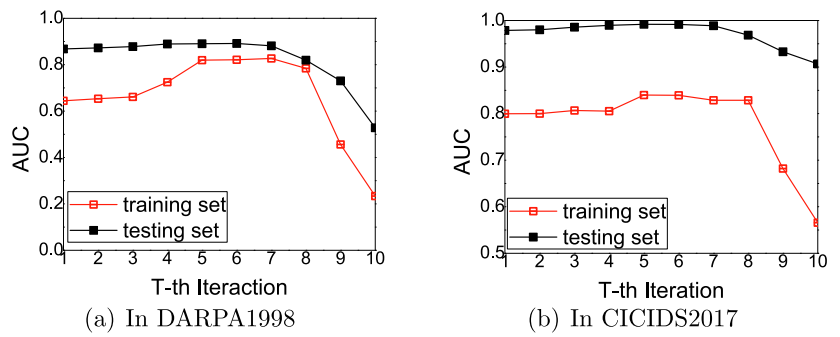


Fig. 8. Data cleaning performance in the training set and anomaly detection performance in the testing set on *AUC* with 10 times self-learning iterations.

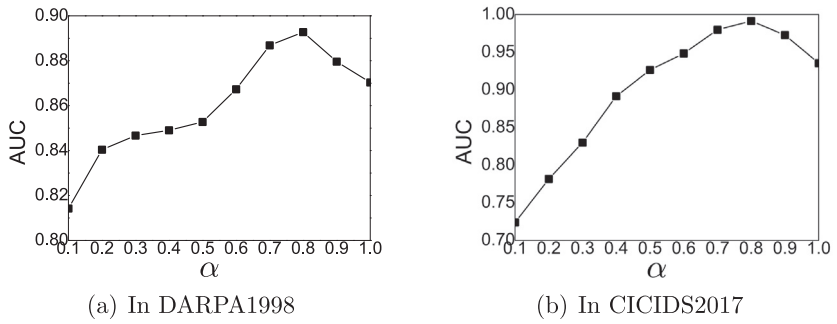


Fig. 9. Impact of α on anomaly detection performance..

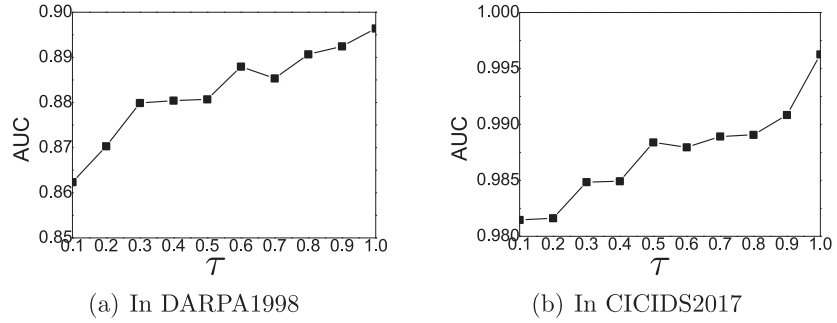


Fig. 10. Impact of τ on anomaly detection performance.

There are several interesting directions need further investigation. First, in this work, though we consider time series, we do not exploit the streaming data. Thus, we would like to explore online cyber anomaly detection with low costs in a streaming setting. Second, as timely supervisions can provide great help for detecting zero-day attacks, we would like to study how to exploit new labels for online learning. Third, based on the detected label conditions in this paper, to specify the exact abnormal flows would be another interesting future direction.

CRediT authorship contribution statement

Xuying Meng: Conceptualization, Methodology, Writing - original draft. **Suhang Wang:** Writing - review & editing. **Zhimin Liang:** Software, Validation. **Di Yao:** Writing - review & editing. **Jihua Zhou:** Supervision. **Yujun Zhang:** Funding acquisition, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported in whole or in part, by National Key Research and Development Program of China (2018YFB1800403), the research program of Network Computing Innovation Research Institute (E061010003), National Science Foundation of China (61902382, 61972381, 61672500) and the Strategic Priority Research Program of Chinese Academy of Sciences (XDC02030500).

References

[1] Y. Ban, X. Liu, L. Huang, Y. Duan, X. Liu, W. Xu, No place to hide: Catching fraudulent entities in tensors, in: WWW, 2019..
[2] B.L. Bars, A. Kalogeratos, A probabilistic framework to node-level anomaly detection in communication networks, INFOCOM, in, 2019.
[3] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, N.F. Samatova, Anomaly detection in dynamic networks: a survey, Wiley Interdisc. Rev.: Comput. Stat. 7 (3)..
[4] W. Cheng, K. Zhang, H. Chen, G. Jiang, Z. Chen, W. Wang, Ranking causal anomalies via temporal and dynamical analysis on vanishing correlations, in: KDD, 2016..

[5] R.A. Rossi, B. Gallagher, J. Neville, K. Henderson, Modeling dynamic behavior in large evolving graphs, WSDM, in, 2013.
[6] M. Ahmed, A.N. Mahmood, J. Hu, A survey of network anomaly detection techniques, J. Network Comput. Appl..
[7] R. Chalapathy, S. Chawla, Deep learning for anomaly detection: A survey, arXiv preprint arXiv:1901.03407..
[8] V. Jyothsna, V.V.R. Prasad, A review of anomaly based intrusiondetection systems, Int. J. Comput. Appl. 28 (7) (2011) 26–35.
[9] S. Mohurle, M. Patil, A brief study of wannacry threat: Ransomware attack 2017, Int. J. Adv. Res. Comput. Sci. 8 (5)..
[10] S. Mukkamala, A.H. Sung, Detecting denial of service attacks using support vector machines, FUZZ-IEEE, in, 2003.
[11] Y. Zhou, M. Han, L. Liu, J.S. He, Y. Wang, Deep learning approach for cyberattack detection, in: INFOCOM Workshops, 2018..
[12] J. Kim, S. Bu, S. Cho, Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders, Inf. Sci..
[13] Q. Huang, P.P.C. Lee, Ld-sketch: A distributed sketching design for accurate and scalable anomaly detection in network data streams, in: INFOCOM, 2014. .
[14] I. Nevat, D.M. Divakaran, S.G. Nagarajan, P. Zhang, L. Su, L.L. Ko, V.L.L. Thing, Anomaly detection and attribution in networks with temporally correlated traffic, IEEE/ACM Trans. Netw. 26 (1)..
[15] J.J.Q. Yu, Y. Hou, V.O.K. Li, Online false data injection attack detection with wavelet transform and deep neural networks, IEEE Trans. Ind. Inf. 14 (7) (2018) 3271–3280.
[16] D. Eswaran, C. Faloutsos, Sedanspot: Detecting anomalies in edge streams, in: ICDM, 2018..
[17] D. Eswaran, C. Faloutsos, S. Guha, N. Mishra, Spotlight: Detecting anomalies in streaming graphs, in: KDD, 2018..
[18] X. Zhu, A.B. Goldberg, Introduction to Semi-Supervised Learning, Morgan & Claypool Publishers, Synthesis Lectures on Artificial Intelligence and Machine Learning, 2009.
[19] Y. Li, J. Ye, Learning adversarial networks for semi-supervised text classification via policy gradient, KDD, in, 2018.
[20] R.A.R. Ashfaq, X. Wang, J.Z. Huang, H. Abbas, Y. He, Fuzziness based semi-supervised learning approach for intrusion detection system, Inf. Sci. 378..
[21] G. Pang, C. Yan, C. Shen, A. van den Hengel, X. Bai, Self-trained deep ordinal regression for end-to-end video anomaly detection, in: CVPR, 2020..
[22] S. Akcay, A.A. Abarghouei, T.P. Breckon, Ganomaly:Semi-supervised anomaly detection via adversarial training, ACCV, in, 2018.
[23] D. Li, D. Chen, J. Goh, S. Ng, Anomaly detection with generative adversarial networks for multivariate time series, arXiv preprint abs/1809.04758..
[24] H. Zenati, M. Romain, C. Foo, B. Lecouat, V. Chandrasekhar, Adversarially learned anomaly detection, in: ICDM, 2018..
[25] H. Zenati, C.S. Foo, B. Lecouat, G. Manek, V.R. Chandrasekhar, Efficient gan-based anomaly detection, arXiv preprint arXiv:1802.06222..
[26] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A.C. Courville, Y. Bengio, Generative adversarial nets, in: NIPS, 2014..
[27] J. Donahue, P. Krähenbühl, T. Darrell, Adversarial feature learning, arXiv preprint..
[28] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[29] D.P. Dubhashi, A. Panconesi, Concentration of Measure for the Analysis of Randomized Algorithms, Cambridge University Press, 2009.
[30] G. Pang, L. Cao, L. Chen, H. Liu, Learning representations of ultrahigh-dimensional data for random distance-based outlier detection, in: KDD, 2018..
[31] D.D. Clark, S. Landau, The problem isn't attribution: It's multi-stage attacks, in, in: Proceedings of the Re-Architecting the Internet Workshop, 2010.
[32] Y. Zhang, X. Luo, H. Luo, A multi-step attack-correlation method with privacy protection, J. Commun. Inf. Networks 1 (4) (2016) 133–142.
[33] M. Inokuchi, Y. Ohta, S. Kinoshita, T. Yagyu, O. Stan, R. Bitton, Y. Elovici, A. Shabtai, Design procedure of knowledge base for practical attack graph generation, in: AsiaCCS, 2019.
[34] J. Atwood, D. Towsley, Diffusion-convolutional neural networks, in: NIPS, 2016, pp. 1993–2001.
[35] L. Wu, P. Sun, Y. Fu, R. Hong, X. Wang, M. Wang, A neural influence diffusion model for social recommendation, in: SIGIR, 2019.
[36] Z. Yang, D. Yang, C. Dyer, X. He, A.J. Smola, E.H. Hovy, Hierarchical attention networks for document classification, in: NAANL, 2016. .
[37] H. Zhang, I.J. Goodfellow, D.N. Metaxas, A. Odena, Self-attention generative adversarial networks, in: ICML, 2019..