Research Article

Design of Graph-Based Layered Learning-Driven Model for Anomaly Detection in Distributed Cloud IoT Network

Osamah Ibrahim Khalaf, Kingsley A. Ogudo, and S. K. B. Sangeetha

1Al-Nahrain Nanorenewable Energy Research Center, Al-Nahrain University, Iraq
2Department of Electrical and Electronics Engineering Technology, Faculty of Engineering and the Built Environment, University of Johannesburg, South Africa
3Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, India

Correspondence should be addressed to S. K. B. Sangeetha; sangeets8@srmist.edu.in

Received 13 March 2022; Revised 20 March 2022; Accepted 2 April 2022; Published 20 April 2022

1. Introduction

Smart IoT devices are fast-gaining traction in a variety of IoT application sectors. With the expansion of IoT ecosystems, resistance behavior has increased in volume and durability, threatening system stability and thus affecting applications that are sensitive to delays. Anomalies in network settings are defined as abnormal, uncommon use of resources [1][2]. The urgent requirement to prevent or eliminate anomalies while preserving global stability corresponds with the necessity to improve existing approaches to overcome the challenges posed by anomaly complexity and the lack of labeled data. Machine learning (ML) methods have acquired a lot of traction and are now widely utilized to detect abnormal traffic patterns. To mitigate the effects of impending/current cyber-attack failures, layered networks have been used to learn the patterns of sent data, classify aberrant occurrences, and ensure high levels of reliability [3, 4] When using layered detection methodologies, security breaches are represented as abnormal data patterns that, when compared to the usual state, show the presence of malware, botnets, or infrastructure failures [5].

As a result of the introduction of the latest methods, graph-based anomaly detection research has gained momentum. Graphs are an important approach representing the connection between network nodes and their interrelationships in network-based anomalies. To protect the smooth operation of the network and the continuous provision of sensitive services, as in the case of critical IoT infrastructure, structure, behavior, and temporary data are associated with network nodes and edges, creating behavioral patterns that define normal or distorted behavior. To address the complex IoT network infrastructure, this study uses a graph-based learning approach [6, 7]. We propose a high-precision detection method within the IoT ecosystem that takes structural elements and relationships between surrounding nodes and their associated edges into account. We also go over the service limits that come with IoT device integration, such as
how to identify low-bandwidth and low-power requirements. We employ a distributed multidisciplinary system to tackle the problem of a region at risk of being uninformed of its anomalies, offering misleading information about its status, or having an abrupt internet connection in response to the difficulty posed by the natural disintegration of cyber-attacks [7, 8].

Anomaly detection of active nodes in the IoT network, as well as a graph network to show the actual network, will be used in distributed construction. Synergistically, the flow of information between neighboring agents will allow them to identify and protect the network [9]. There is no additional bandwidth or power consumption because there is no centralized access detection system to monitor and analyze all traffic flowing through network nodes. Therefore, there is a need for research on anomaly detection systems. Anomaly detection based on graphs has been widely utilized to prevent network failures while taking into account the mergers of the organizations involved, modeling their interactions, and merging their structural, content, and transitory properties [20]. In this paper, we propose a multi-agent system that detects abnormalities by utilizing the collaborative environment of smart agents. A Graph-based Layered Learning-Driven (GLLD) network is implemented by each agent. To that end, we develop an anomaly detection technique that aims to efficiently monitor the entire network architecture to resist cyber-attacks propagation. The remainder of the article is organized as follows: Section 2 provides background information on anomaly detection. The proposed methodology for implementing the graph-based anomaly detection method is described in Section 3. Section 4 presents the evaluation results of the proposed method compared to traditional ML techniques. Finally, Section 5 concludes the study.

2. Related Work

A lot of studies have gone into integrating and optimizing current algorithms to identify the best detection method during the previous few years. Aside from that, graph-based methodologies for identifying anomalies on nodes and edges have been developed, as well as clustering algorithms. Bayesian networks have been used to look at probabilistic tactics that take historical data into account. This section looks at some of the most well-known new anomaly detection techniques.

Anomalies are often found using group-based techniques, approaches, or divisions. External or anomalies find new ways to hide in fast-growing social networks, making the old methods useless. In this study, anomalies in emails and Twitter networks are discovered using the in-depth reading ability to learn the topological features of a social network. We present horizontal reading based on graphs, how to find inaccuracies on social networking sites. Various steps for social networking statistics are considered using layers to investigate graph structure and performance of unusual nodes. The hidden layer of the layered network is important in determining the effect of compound mathematical compounds on anomaly discovery [10].

To maintain stable satellite performance, assess satellite status, and increase satellite efficiency, an anomaly discovery system based on graph neural network is used. With telemetry data, we begin by constructing a layered network graph model. The model graph design module collects feature attributes, while the location-dependent module and the temporary dependency module exclude location and time dependencies, respectively. Thereafter, data are predicted using a trained model, as well as the difference between expectations and interdependence, actual values are calculated. Finally, wavelet variation is used to measure data time. A dynamic threshold method based on time frame is used to detect inaccuracies in data collection. According to test results using satellite power system telemetry data, the accuracy of the proposed algorithm is more than 98 percent, with efficiency and endurance [9].

An autoencoder-based approach to developing widgets using Graph Neural Networks (GNN) for the anomaly is provided. To circumvent the known boundaries of GNN autoencoders, we create symmetric video that can recreate terminal and node structures simultaneously. We find that discriminators based on secretive location can help differentiate novel physics and SM signatures that compete with QCD jet offerings that limit sensitivity. To highlight the flexibility and widespread use of this method, we use W bosons, top quarks, and exotic hadronically decaying exotic scalar bosons [11].

The anomaly discovery of dynamic graphs is an important function for security, economic, and communications applications. Existing network-based strategies generally ignore changes in the layout of the subgraph associated with the target nodes in a given time window, instead of focusing on a good reading of the representation of the nodes. This study proposes StrGNN, an end-to-end Temporal Graph Neural Network architecture model to detect unusual edges on flexible graphs. After extracting the h-hop that closes the subgraph in the middle of the target edge, we suggest a node labeling function to identify the role of each node in the graph below. Using the graph conversion function and Filter Backup layer, a fixed size element is extracted for each summary/timestamp. We use gated recurrent units to capture temporary data to identify confusion using restored features. StrGNN is fully integrated into the real-world business security system and contributes significantly to the identification of complex threats and the effectiveness of event responses. Extensive testing of six benchmark datasets also demonstrated the effectiveness of StrGNN [12–14].

IoT networks are made up of thousands of different internet-enabled devices and may be used by millions of different end-users. Such a network should be considered under direct attack at all times. A security strategy based on IoT infrastructure device interaction can lead to faster and more efficient and effective detection and response. As a result, modern design methodologies are needed to help ensure security on IoT network infrastructure while also lowering latency effects thanks to graph representation benefits. Table 1 depicts the performance comparison of traditional methods. The authors primarily use the hand-crafted feature extraction method in the aforementioned works.
The trained classifiers have some drawbacks, such as low adaptation for datasets from varied network contexts, because the feature set strongly relies on experts’ domain expertise. As a result, it is crucial to reduce the reliance on expert knowledge in the traffic feature extraction process. The raw traffic data is turned into fixed-length feature vectors in the proposed system. Then, using the basic idea of a graph-based learning model, the parameters are trained.

3. System Methodology

We suggest developing modern design methods that can help achieve security on IoT network infrastructure while providing lower latency effects through graph representation benefits. The main contribution is a distributed system that is powered by a graph-based reading model. The graph network can fully detect the complexity of node-to-node relationships and strongly demonstrate their dependence. Compared with existing neural network methods, the proposed method uses graph-based layered learning to adapt to the design of the target agent’s graph and provide a high level of interaction and complex learning strategies [15, 16].

Only nearby agents can share observations. In this case, the central authority is not required to achieve shared information globally. As a result, bandwidth requirements and computer power are reduced. An important part of the current effort is the ability to effectively identify attacks and malicious incidents and avoid consequences based on the views of the nodes and the surrounding environment. Integrating a neural network model frees up the burden of processing multiple volumes of data in an IoT environment depending on the content of the edge and the node and structural features [17].

The detection of low anomaly and distributed functionality provided by the connection to all physically connected nodes provides an additional value. Injecting graphs to show network links and also to show that agents are very effective in terms of speedy recovery from an attack, even if a significant number of nodes are compromised. To our knowledge, graph-based learning models have never been widely used in IoT communication environments, and have never been used to identify dangerous sources or predict future attacks [18, 19].

Graph-based learning models can work directly in a graph structure, with node devices as nodes and communication channels as edges, and the IoT device network can be viewed as a graph with node devices as nodes and communication channels as edges. By using this strategy, we can extract many of the features found in the network and use them to separate nodes. Every node in the graph is given a label that describes the behavior of the node. This data format represents the first method. An effective way to allow messaging over connected organizations is provided, using the edge and transmitters as graph nodes and their sub-links as the ends of the graph.

3.1. Design of IoT Network. A standard IoT network infrastructure is presented, which can be used for a variety of IoT application environments. We explore a three-layer IoT infrastructure: IoT devices, Fog Nodes, and clouds are all part of the IoT ecosystem. Management applications that deal with sensitive data while operating at Fog Nodes, located near edge devices, reduce the delay associated with transferring large amounts of data to a central computer, as is the case in most IoT cases. As a result, agents may be deployed using Fog Nodes or Edge Forwarders. In the first case, the agent is considered to be an operating system on a virtual server or container, and secondly, Edge transmitters can support agent performance with a dedicated low-power AI processor.

A feature that should be emphasized is the local performance of traffic monitoring that is performed on each agent. Each agent is responsible for a subset of total transfer traffic known as its location, which includes agents directly related to it [13, 14]. The proposed design, shown in Figure 1, includes IoT devices and Edge/Core transmitters. Forwarders host AI agents to facilitate traffic monitoring from IoT devices. The transmission controller provides traffic flow to agents such as raw data, using a known standard for traffic monitoring.

3.2. Feature Extraction. To provide the appropriate features at the bottom of the graph below and vertices, the raw data should be considered in advance. Data flow fields specify network statistics and offer an in-depth overview of crucial information such as contracts, IP addresses, and their corresponding ports, as well as extra test data such as the total number of bytes passed, packets, duration, and start time for each flow. The current method involves the removal of features to provide attributes to the network’s nodes and edges, which necessitates the reconstruction of flow-level information at the node level. This action should result in the inclusion of the elements that are most closely related to the aberrant traits, while unwanted noise is removed.

In this regard, the proposed confusing method requires the following input data: start time, duration, protocol, source device, target device, IP address, connection direction, packet number, and forwarded bytes. The network is
expected to include a large number of nodes to match the graph creation and attribute functions. To assist in the creation of a common and unparalleled pattern distribution, the interim rate of data, which refers to the total length of time spent on traffic, is expected to be significant. We use the above information to generate important features linked to the presence of abnormalities. The exchange of information in a database is represented by a flow sequence with a recurring source and destination, which defines multiple relationships. To provide features for each location and edge of the graph, we combine statistics to provide information for each node, which leads to the creation of vectors for different node features and terminals.

3.3. Graph-Based Layered Learning-Driven Model. Assuming that $G$ is a graph and $t$ is a time window, the vector feature $v = p, b, t$ containing multimedia records is defined as $v = p, b, t$ containing multiple index records, where $p$ be the number of packets, $b$ is the number of bytes, and $t$ is the connection time. The graph is often used to process structural data while
considering its relevance. Despite these advantages, as data from previous and next neighbors are included in the processing function, computer complexity increases. A graph-based layered learning model was developed to address this problem and to improve the message transmission process, suggesting that the node vector is represented as vectors of the integrated and modified element of its neighbor.

The basic calculation block takes the graph as inserted and generates a new graph model after performing calculations between the nodes and the edges of the graph. Let us name the basic elements of a graph, nodes and edges, \( v \) and \( e \), respectively. To manage message transmission between network nodes, horizontal graph-based models advise displaying the node vector as integrated and modified with its neighbor feature variables. Therefore, the repetition of this merging process, where \( n \) is the multiplication number, downloads the structural details within the network area, including the neighbors within the \( n \)-hops range.

We use a perceptron network to send messages from the edges to their related nodes and change the automation features. The structure was chosen because of its ability to work quickly and produce high-quality results. We see the exchange of information as a series of actions to be completed by each agent:

(i) Find out who your neighbors are and what they are up to

(ii) Update \( e_1, e_2, ... e_n \) edges’ attributes

(iii) Update node’s attributes

\[ f(x) \]

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure3}
\caption{Code Red worm attack dataset - probability density function vs number of packets.}
\end{figure}

\[ \text{True positive rate} \]
\[ \text{False positive rate} \]

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure4}
\caption{ROC curve comparison.}
\end{figure}
After exchanging information with nearby agents, each agent uses the data acquired to train and test the anomaly detection algorithm abnormalities.

There are two types of anomalies to consider:

(i) Now, at time $t$, the forwarder is abnormal

(ii) The forwarder will be infected by malware that spreads throughout the network, i.e., at time $t+1$, the forwarder will be infected

The perceptron layer of the node also is responsible for changing the element that represents the node element in response to collected data. The node element vector itself is passed through layers, which are then separated to determine the probability of $j$ (each node) of the node is possible. We create effective information sharing in an attempt to detect attacks and distribute them over the network, demonstrating the use of a multi-agent system compared to previous centralized designs. We used a threshold value of 0.50 to produce normal and abnormal events due to the potential for the outcome.

### Dataset Evaluation

At this stage of the study, we consider using a pre-trained model that can detect unusual after learning multiple network traffic patterns. The use of a
buffer, on the other hand, may address the need to ensure continuous network traffic data. Although many online datasets were found showing normal and unusual network traffic of various types, the graph-based descriptive data structure was not taken into account in sufficient quantities. In addition to displaying the actual network settings, the above datasets do not exclude fields such as source, destination, traffic statistics, and start connection time, or do not have annotation data.

To solve this problem, we created data showing structural links between IoT nodes and edges. After defining the distribution of data across all normal and opposite patterns, the production of synthetic data can begin. All generated datasets follow the same distribution while separating the total numbers of packets transferred, received, and their time, to obtain a complete picture of the proposed road capabilities and to assess various network traffic conditions.

3.4.1. BoT IoT Dataset. To re-establish data interchange, we analyzed the current dataset, which was recorded at UNSW Canberra’s Cyber Range Lab. Botnet, Normal, and Background traffic are all labeled. It consists of a variety of attack situations. All of the cases are botnet instances originating from a single malicious IP and carrying out malicious operations while exploiting a variety of protocols. Each situation was documented on a piece of paper that followed the prescribed format [19][21].

3.4.2. Code Red Worm Attack Dataset. We have imitated the attack of worms on the data database. It can detect the
difference between normal and unusual network traffic flow. We have installed an invalid botnet signature stream on the network, which includes 50 transmitters and 120 IoT devices communicating with data according to BoT IoT distribution. Properly representing the DDoS attack on an infected transmitter, we have created an offline environment, which has resulted in hacking features being given to malicious hackers. The effect of distribution was also described in this way. We explain how a graph-based network can detect attacks before they spread across the network by testing these attacks [13][22].

Preprocessing the high dimensional dataset is done using one-hot encoding and label encoding. One-hot encoding addresses the ordering issue. Many columns in the dataset are having column names with bridge-type values. Therefore, categorical values use label encoding. A standard scaler is also applied. Figures 2 and 3 depict the comparison between probability density function and a number of packets for BoT IoT and Code Red worm attack dataset, respectively. Results according to Accuracy, ROC AUC score, Bandwidth, and Power Consumption compared to those of ML algorithms (SVM, Decision Tree, Random Forest), which are often used for ambiguous detection, highlight the high performance provided by Recommended Action.

We examined the increase in bandwidth and power consumption caused by the confusing acquisition method in the case of using a mid-range conferencing strategy. After declaring the number of transmitters involved in network traffic and the flow of communication that occurs every second, we analyzed the results.

4. Experimental Analysis

We studied the BoT IoT dataset, which was recorded at UNSW Canberra’s Cyber Range Lab, to re-establish data interchange. Botnet traffic, normal traffic, and background traffic are all identified. It is made up of a range of attack scenarios. All of the incidents include botnet instances that originate from a single malicious IP address and carry out malicious operations using a range of protocols. We have replicated a worm attack on the data database. It can distinguish between regular and abnormal network traffic flows. On the network, we installed an invalid botnet signature stream, which consists of 50 transmitters and 120 IoT devices communicating with data in accordance with BoT IoT distribution.

In the ROC diagram shown in Figure 4, we compared the accuracy of the detection with the ROC score. The accuracy comparison of various epochs is depicted in Figure 5. Figures 6 and 7 depict the bandwidth and power consumption comparison, respectively. Three previous ML categories were trained and evaluated on the same databases as the proposed method, and the results were compared. Compared with the rest of the methods, the layered learning models based on the graphs surpass the residual methods in the diagrams. Two sub-cases were also considered, one of which explained the case in which three unusual transmitters went online at the time of the attack. The fact that the proposed strategy transcends complete computer precision highlights the importance of multi-agency collaboration and the exchange of information on healthy nodes directly related to detecting attacks on their neighbors.

Because feature vectors are only exchanged for surrounding agents using directly attached edges, data load and data processing are avoided in one central location. Risk-connected transmitters have no effect on a graph-based reading system in the event of a broadcast attack, indicating that surrounding agents are aware of anomalies. In particular, the performance of the proposed graph-based model is demonstrated. Even though all methods offer high accuracy, the proposed method significantly surpasses other methods in terms of bandwidth and power consumption. This effect is measured by the reduction of scattered element vectors alternating between the surrounding agents, as opposed to the transmission of large messages to the central processing unit that occurs in central systems. This use of low-cost resources is critical to the IoT ecosystem.

Compromised transmitters are connected to the Internet, as evidenced by bandwidth and power consumption data, leading to a decrease in the traffic transmitted across the network. The accuracy of all the centralized methods is reduced due to the reduced number of monitors managing to detect inaccuracies, while the graph-based linked agents are not affected. Because smart agents interact with traits and can differentiate the risk of neighbor infection, a graph-based layout approach accurately detects growing threats. The remaining algorithms, on the other hand, do not adequately detect the spread of attack sources. This is evidenced by the fact that in the early stages of learning, distorted nodes are considered normal because the attack has not yet spread.

Two sets of data have been developed: one for scattering worms and one for spreading the pattern of root scattering to branches. In all of the cases, we check that the proposed algorithm outperforms even in training time as depicted in Figure 8. Training and measuring 0.02 learning rate and 5000 weight loss, selected after fine-tuning, both applied to each database. The processing time for calculating node/edge features increases as the collected traffic is longer, which increases training time. When the flow value of each node indicates the number of edges per node, data processing becomes more difficult.

5. Conclusion and Future Work

This paper outlines a method for recognizing abnormal events. This method creates an edge and node classifier that estimates the chance of infection on a node and its vertices using a graph-based layered learning mechanism and fully linked networks. We were able to transmit information inside an agent’s neighborhood using the graph representation, assisting in the detection of anomalies and security breach occurrences based on their interrelationships. To address the problem of an insufficient number of monitors against distributed attack patterns and compromised nodes being unaware of their infection status, we leveraged the perspective that a node has of its nearby nodes’ health condition. To train and test the graph-based layered learning
model, we used real-world datasets with conventional traffic patterns, as well as data generated using an aberrant distribution. The results demonstrated that the method could be used to ensure network stability and flawless functioning in the face of dispersed assault situations, with high accuracy and low bandwidth and power consumption values. Traditional methods of deterring potential attackers and cyber-criminals include using encryption on edge devices to prevent unwanted access, while data encryption techniques are required for network security. In future work, attack detection, harmful patterns, and abnormal behavior, as well as the necessary mitigation and avoidance methods, will be focused on.

**Data Availability**

No data were used to support this study.

**Conflicts of Interest**

The authors declare that they have no conflicts of interest.

**References**


