Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.7632385

IMPLEMENTATION GRAPH CONVOLUTIONAL MODELS FOR DETECTING CYBER SECURITY ATTACKS AND MALICIOUS COMMUNICATION ACROSS APPLICATIONS

Kanniappan Elumalai
Research Scholar, Department of Computer Science and Engineering, Academy of Maritime Education and Training (AMET University)
* Correspondence: kanniappan1mam@ametuniv.ac.in

Dr. Duraimutharasan Bose
Professor, Head of Department of Computer Science and Engineering, Academy of Maritime Education and Training(AMET University), Chennai-India

Abstract
Microservices are now the most common architecture for designing software applications. Technology companies are also the target of an increasing number of cyberattacks on a daily basis, and new security solutions are in great demand. The use of intrusion detection system, that is characterized as the identification of irregular or unusual behaviour that occurs to a higher or smaller extent than typical instances in a data set, is one accessible measure. In this article, we extend previous work in which several real- world cyberattacks are launched against an operating microservices system, and the application activity is monitored and provided as distributed traces. A Stochastic Generative Adversarial Deep Convolution Infrastructure (SGADCI) is employed to model the collection of dispersed traces and to discover the geographical and temporal relationships of the data packets. Afterwards, the algorithm is used to produce predictions for current microservice activity, and cutoff point anomaly detection is employed to detect unusual microservice activity that indicates the presence of cyber security attacks that have been seeded, or anomalies. Cyberattacks such as brute force, batch activation of bot accounts, and disseminated denial of service were utilised to test this strategy.
Keywords – cyber security, attack detection, microservices, cyberattacks, deep neural networks.

I. INTRODUCTION

For almost twenty years, the Internet has been a vital part of international communication and has been deeply ingrained in people's daily lives everywhere [1]. As a result of advancements and decreased costs in this field, the Internet is now used by around 3 billion people throughout the world. The Internet has established a worldwide infrastructure that contributes billions to the world economy every year. The majority of today's economic, commercial, cultural, social, and governmental activities and contacts between nations, at all levels, are conducted in

cyberspace, including those involving people, non-governmental groups, and government and governmental institutions.

Most essential and confidential information is transported to this space, or has been produced in this space, and critical infrastructures and technologies either exist in cyberspace or are controlled, managed, and manipulated through this space. The majority of people time and energy are invested in participating in this area, and most media operations are shifted to it.

There has been a dramatic rise in the proportion of a
country's GDP attributable to online economic activity, and
cyberspace metrics now play a crucial role in measuring the level of development in different nations. Countries invest a
great deal of their material and spiritual resources here, and citizens' material prosperity and spiritual accomplishments derive from or significantly affect this realm [2].

In other words, many facets of people's life depend on this area, and any instability, insecurity, or difficulties there will have repercussions in many other areas of people's lives. However, the advent of the Internet has presented new threats to national security. Cyberspace's low barrier to entry, pseudonymise, ambiguity of the geographical area of threat, dramatic impact, and lack of public accountability have attracted strong and weak actors like governments, organised and terrorist groups, and even individuals, who pose threats like cyber warfare, cybercrime, cyberterrorism, and cyberespionage [3]. This is what sets cyber threats apart from conventional national security concerns, which are opaquer in disposition and those whose characters are government agencies and governments that are easily recognised in a particular geographic area, and what has challenged and rendered ineffective national security in its traditional sense in this space. Here, Fig 1 shows an overview of the cyber attack process.

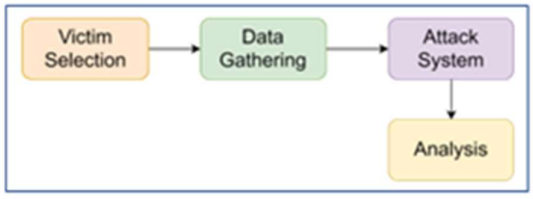

Fig. 1. Overview Of Cyber Attack Process

Researchers and experts have been considering the fallout from cyber strikes for over a decade. It is possible for a virus to attack the financial information of an economic structure or to disrupt a country's stock market; it is also possible for a virus to send the wrong message to a country's power plant, causing it to shut down and fail; and it is also possible for a virus to disrupt the air traffic control framework, resulting in air accidents. Therefore, it will be extremely challenging for experts to handle the multifaceted nature of the problem and give legal analysis unless countries agree on a common definition of a cyberattack. The question therefore arises as to whether or not any form of attack that actually occurs in cyberspace can be called an attack in the traditional and classic sense [4].

The continued and accurate identification of the legal repercussions of cyber-attacks relies on the development of a thorough definition of such attacks. Uncertainty about the prevailing legal approach, as well as differences in interpretation and practise, can lead to different and even conflicting legal results when a clear and complete definition is lacking. As a result, a thorough

investigation is required, and it is crucial to have a workable definition from the outset in order to explain, adapt, and analyse the issue. One of the world's biggest issues right now is cyber security. Day by day, hackers launch a wide array of sophisticated cyberattacks on a growing number of software companies.

Twitter, Amazon, Netflix, and PayPal are just a few examples of well-known software companies that have adopted the microservices software architecture in recent years. As a result, those in charge of cyber security for these apps need cutting-edge tools for spotting cyber attacks. Researchers have looked at attacks on microservices applications by tracking how the programme as a whole behaves through the use of distributed tracing and then looking for any out-of-the- ordinary patterns to identify an attack. Microservice call graphs, in which the microservices themselves serve as the graph's nodes and API calls between them serve as the graph's edges, are one kind of distributed traces. Observing a series of these call diagrams over time can reveal interesting patterns in the microservice application's API call activity. The next step is to employ graph-based anomaly detection to check for discrepancies in the application's call flow that could point to a problem [5].

To improve upon our prior work in employing anomaly detection to identify cyber-attacks in microservice traffic, we suggest investigating the use of graph-based intrusion detection system on API call traffic graphs generated by the microservices applications. For incorporate the current spatial and temporal dynamics within the tracing data, we leverage the microservice interaction graph and information to train a graph convolutional neural network (GCNN). The abnormal microservice traffic brought on by different seeded cyberattacks is identified by employing a GCNN to model the appropriate basis and anticipate continuing activity. In this research, we employ a distributed monitoring tool to keep tabs on a microservices application in an effort to identify potential cyber security threats.

In addition, we construct a Stochastic Generative Adversarial Deep Convolution Infrastructure (SGADCI) built to learn the unidirectional behaviour of the traffic represented on a directed graph and then execute traffic prediction for future time steps. To conduct our experiment, we deploy a distributed software platform and mimic a variety of cyber threats. By utilising the SGADCI model, we are able to identify the abnormal microservice traffic that results from these attacks [6]. When a user makes a request to a programme, the programme responds by making a series of calls to connected microservices. With the use of distributed tracing, we can see that this chain of RPCs was successfully recorded.

The distributed traces of the programme are made up of the RPCs that are called regularly by users. To further understand the geographical relationships and temporal dynamics, the SGADCI model is taught to analyse this RPC traffic. This method is used to identify, within a predetermined time frame, any RPC dynamics that deviate from the typical behaviour of communication between microservices. This study aims to discover abnormalities by comparing the RPC communication from a regular data set with the RPC traffic computed in relation to a cyber security attack [7]. It's worth pointing out that the method employed is somewhat analogous to the one we developed for doing graph-based anomaly identification. Our unique method streamlines the GCNN model-training process and teaches itself to predict the spatial and temporal behaviour of RPC traffic. In contrast to methods that train many

models, each of which is responsible for learning a distinct component of the microservice application, our method just requires training a single model.

This paper an introduction section covers over view of cyber-attack and the fields. We can look at earlier research articles and studies done by other scholars throughout the world in the literature review. The methods section covers the methodology that has been used in this paper. The results and discussion section includes the results that we have gotten in this study. Conclusion and future scope section comprises the results. The machine learning approaches and their outcomes are detailed in this paper as well as the characteristics and other details about individual algorithms are also provided.

II. Literature Survey

This section provides a survey of relevant studies, such as various machine learning-based strategies for detecting anomalies in network data using graphs. Modeling data and uncovering its underlying behaviour are two applications of deep neural networks (DNN). For modelling time-ordered data, researchers often turn to recurrent neural networks (RNNs). Researchers employed a specific kind of RNN, called Long Short-Term Memory (LSTM) neuromorphic Transmission capacity and introduced by Hochreiter, to identify long or short-term relationships in cases; cases are the standard representation for such data series. By training on examples of common occurrences, our LSTM-based system was able to provide forecasts about when certain events will occur [8].

Learning and understanding the behaviour of logged instances from two accessible data sets was used to test the effectiveness of this framework, and the findings showed that it outperformed a previous technique. The traffic flow of a network domain has also been modelled using deep learning models. The modelling and analysis of graph structures and images are well suited to Convolutional Neural Networks (CNNs), a type of DNN. With enough data, the CNN model may figure out how the traffic flows in different areas. The authors suggested a convolutional neural network (CNN) model to learn incoming network traffic as pictures, which could preserve the temporal and spatial changes of the data and allow for the prediction of network traffic speed [9].

Two datasets modelling actual commuting patterns were used to evaluate the performance of this CNN technique. The CNN-based framework was compared to four commonly used statistical procedures including three deep learning-based models, and it was shown to have a higher accuracy by 42.91 percent compared to the other methods. To address this issue, Wu and Tan proposed a deep model with a convolutional neural network (CNN) and long short-term memory (LSTM) combined architecture (CLTFP), where CNN element was employed to encapsulate and learn the feature space of the traffic flow and the LSTM element was used to learn the long - term dependencies. In the end, the developed model was put to use for making short-term traffic predictions.

After comparing the CLTFP model's predictions to those of other models including the long short-term memory (LSTM), the shallow neural network (SNN), and the stacked autoencoder, it was found that the CLTFP model provided the most accurate and evenly distributed forecasts. KungHsiang's work on graph neural networks (GNNs) has recently gained traction for its ability to describe nodes and relationships in contexts as diverse as life science and social

networks. Spatial-temporal GNNs (STGNNs) are a subset of GNNs designed to concurrently capture spatial and temporal aspects inside associated data graphs in order to predict future activity in a variety of contexts [10].

Timely congestion prediction is critical for effective traffic control, as Yu et al. point out, but classical mathematical methods, such as linear regression, are not well-suited to long-term traffic forecasting in the future. It was proposed to use a STGNN to simulate the time-series bayesian inference issue in a traffic domain. To facilitate quick training only with STGNN and to capture the spatial and temporal information, the road segment network was modelled on graphs utilising convolution structures [11]. This method was tested on a number of real-world traffic data sets, and the findings demonstrate that it readily converges and surpasses state-of-the-art benchmark models.

Stochastic Generative Adversarial Deep Convolution Infrastructure (SGADCI) is a recent GCNN that is a state- of-the-art model built for understanding the intricate spatial and temporal details of traffic patterns. The authors explain how spatiotemporal traffic forecasting may be used in the field of transportation networks. They suggested thinking of traffic as an interactive phase transition on a graph structure. Simulations of traffic growth activity are made after learning from the ground truth observations. Two datasets were used to evaluate the approach, both of which contained data on actual traffic on roads [12]. The initial data collection is four months' worth of traffic data collected by 207 sensors in various locations around Los Angeles County.

After being put through its paces, this framework was shown to produce better results than the state-of-the-art baseline architectures by a rate of return of 13% to 18%. This research builds upon the work of Chen et al., who employed a GCNN to identify anomalies in actual RPC traffic. The latter intended to find vulnerabilities in the cyber defences protecting the hundreds of RPCs that emerged as a result of using many different microservices [13]. In this study, we used a two-stage procedure to track and record RPC traffic and identify outliers. To begin, a density-clustering method was used to the data collected from the recorded RPC traffic of running microservice functionality in order to identify patterns in the data linked to RPC chains. Chain patterns like this might be thought of as a subset of the larger microservice architecture. Then, the erratic RPC prediction problem is tackled by using a GCNN to model the individual components of the RPC traffic and to pick up on the traffic's spatio- temporal relationships. These GCNNs allow for a series of separate predictions to be made for each of the current modules. Two case studies made up of real-world suspicious network threat scenarios, such as a batch enrollment of bot identities and account breaking, were used to assess the efficacy of this method. When simulating network traffic over time, Le et al. employ a traffic dispersion graph approach. This strategy is split into two parts: the first, which learns the graph's static characteristics, and the second, which models the interdependence of the TDGs' temporal dynamics over time [14].

DDoS attacks, port scanning, and Internet worms are all examples of illicit computer activity that contribute to what is known as "anomalous traffic." Using this TDG model, we were able to spot and investigate the root causes of abnormalities in the form of time-varying, non-normal network traffic. Both sets of communication traces were used to determine how well this TDG technique could spot a cyberattack, and the results were unanimous: the procedure

was highly effective. By training a hybrid CNN/LSTM deep learning algorithm dubbed STDeepGraph on network flow traffic, Yao et al. propose a high-level attack framework for packet forwarding data. In order to represent the communication structure of a network, this study use a temporal communication graphand a distance graph kernel to map the data into a multidimensional space [15].

Temporal aspects of the network flow were extracted by the LSTM component, while spatial features were extracted by the CNN component. The model then uses a softmax segmentation function to determine if the traffic on the network is safe or harmful. The STDeepGraph was tested in two separate trials utilising data sets of actual network attacks in which different types of attacks were interspersed with normal traffic. Accuracy was one of the measures used to assess the model's efficacy [16]. Based on the findings, this strategy is superior to the status quo in terms of reliability and loss. In their paper, Lee et al. (2020) offer a deep learning model that uses a directed graph of traffic-based data, which changes over time, to understand the data's spatio-temporal patterns. Non- Euclidean distance was utilised to forecast dynamic anomalies using the model.

To achieve this, we used an already-existing data entity's affinity score. Then, a cutoff point is determined to identify out-of-the-ordinary activities. Two sets of traffic data, one from the internet and one from public transportation, were used to assess the effectiveness of the method. Accuracy in predicting new relationships and absolute differences in predicted affinity scores were used to gauge the model's performance. Competent findings comparable to those of state-of-the-art methods were demonstrated by the model [17].

III. Methods

The major contributions of the research are stated as follows:

- We represent the complete polylithic infrastructure of a microservices platform, including the connections between the various services, using a directed graph.
- To illustrate how network applications from one distributed software node to its neighbours might be compared to a diffusion process, we can use the network graph.
- By gauging the atypicality of RPCs initiated as a result of cyber security breaches, we are able to identify their injection into microservice traffic and so uncover their source.

Microservice Architecture (MSA) networked monitoring, detection and classification is provided here. Decoupling the application into numerous smaller interconnected services is the basic idea behind the microservice architecture also known as microservices. Each microservice is responsible for one discrete business process inside the larger functionality of the programme, such as user registration or database querying. A single application server is a well-defined gateway that works with other micro in a microservices applications, but may be designed, tested, scaled, and deployed separately from the others.

Remote Procedure Call (RPC) A user's request for a RESTful API or an RPC can trigger a call to this interface. Distributed tracing is the ability to record and track in real-time how a process flows via a cloud-native, distributed system. An HTTP request from a user in a microservices application will often trigger a chain reaction across numerous microservices. Afterwards, the

whole execution chain of these microservices is captured in a distributed trace. Each unit that makes up a single trace is called a "span," and all spans in a given trace have the same traceID. With its own spanID, a recorded span stands in for a single microservice action carried out in reaction to an HTTP request from a user. The name, time stamp, and runtime of the microservice activity being called are all details that may be found in the span.
Anodot defines anomaly detection as "the process of looking for and identifying instances of unusual behaviour or events within a data set." These outlying examples either have a higher or lower frequency than the rest of the data collection, or they deviate in some other way from the norm. Anomalies can manifest in a variety of real- world contexts, such as the detection of hostile action through military surveillance, the manifestation of medical conditions through imaging, or the existence of cyberattacks within a computer system. In the context of data analysis, the term "anomaly detection" refers to the practise of looking for outliers within a set of data.

Group anomaly detection (GAD) is a type of anomaly detection that looks for clusters of data that exhibit abnormal behaviour when compared to other data in the same category. The use of graph-based anomaly detection has been used in many different industries, from banking and healthcare to law enforcement and even network-level IT security. To the best of our knowledge, the authors are the first to use this method of anomaly detection in a microservices setting. Initially, they locate groups of RPCs that have common functionality inside the programme as a whole, which may be thought of as its subsystems. The second phase involved training a SGADCI for each current RPC subsystem to use for predicting future events and finding outliers.

Stochastic Generative Adversarial Deep Convolution Infrastructure (SGADCI) Fig 2 depicts the system's overall structure. To study the geographical and temporal relationships of the application traffic throughout a time period. The goal of our research is to develop a method for predicting microservice traffic at a later time step using just historical data.

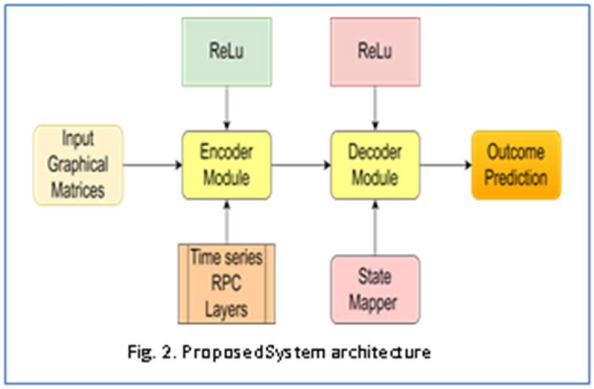

Fig. 2. ProposedSystem architecture

In contrast, we learn the application's typical behaviour by training a single SGADCI model with a more generic unified RPC traffic data set. There is no longer any need to isolate individual subsystems and train separate models for each one. In contrast to the RPC clustering strategy, which proposes using different subsystem models to identify anomalous traffic, using a single model to do so with a more unified data set promotes simplicity and makes the single

model more resilient. Our unique approach consists of providing a more streamlined and generalised way for training a SGADCI to understand the geographical and temporal characteristics of microservices communication and implement graph-based anomaly detection.

Furthermore, we detail how anomalous traffic is identified by comparing the anticipated and real traffic and using the SGADCI model to make adjustments to both of these variables. As a starting point for our strategy, we leveraged an existing microservices application. By making HTTP API calls to the target application and recording the subsequent traces with a distributed tracing tool, we were able to generate synthetic datasets consisting of microservice RPC traffic statistics.

DeathStarBench microservices application we used was created by the Yale University Design and laboratory and is available as open source. The different programmes within this open-source measurement suite have been cited several times in the context of performance management and root-cause investigation of microservices.

Calls to remote procedure call endpoints (RPCs) are common in microservices traffic, and they are used when one service has to communicate with another service that also provides collaborative capabilities. Each RPC call is represented as a node on a directed acyclic graph. A reference pair of RPCs is represented as a node on the graph. When two nodes in the graph have the same source or target RPC, the edge connecting them is given a weight. This method encourages scalability and brings attention to the infrastructure of the microservices program and the many inter-relationships between the microservices. The directed graph is, in turn, expressed as a weighted adjacency matrix. In a more mathematical representation, we model the architecture of the services and applications as a directed graph with weights.

The mathematical representation of H, the corresponding graph, is given by following equation:$H = < O, F, B >$,where O is the set of all distinct source- destination pair nodes found in the traffic, F is the collection of all periphery created among RPCs once nodes exchange whether it is a source or a destination value, and B is a normalised adjacency matrix representing the degree of adjacency among each node, we get the formula. If there is just one path between two RPC nodes, then that path is given more importance. The following weights are applied to each relationship between pairs of nodes x and y. In the event that node x and node y share an RPC source or destination, then there will be a 0.5-weighted edge from node x to node y and a 0.5-weighted edge from node y to node x.

A 1.0-weighted dependence edge from node x to node y exists when a source node is also a target node. Properties for each node in a directed graph in a microservices application are stored in an O by N matrix, where O is the number of vertices in the graph and O is the number of attributes. In our study, we focus on the specific issue when just a single characteristic indicating application traffic is recorded. Simply the total number of times this RPC pair is executed. In order to construct this traffic matrix, we first need to compile a complete record of all RPC calls. There will be U discrete points in history that we'll specify. Let's make things easy for ourselves and give each time step the same amount of time.

We loop over the recorded RPC calls, counting the number of instances each source-destination pair of RPC calls was conducted at each time interval. It returns a sequence of traffic matrices,

$Yu{-}U{+}1$ to Yu, where Yu stands for the traffic matrix Y at time step u, given the directed graph H. Mathematically speaking, traffic forecasting is defined as the process of attempting to foretell the behaviour of traffic based on historical data collected from the same network domain. Our aim for the traffic forecasting issue is to create a function that transfers U signals on the RPC graph at time step u to U future time steps, where H is the directed graph and U is the time series of U traffic matrices.

$$[Y^{(u-U+1)}, \dots Y^{(u)}]g(.)[Y^{(u+1)}, \dots Y^{(u+U)}]$$

Multiple data representations can be learned by GCNNs since they are built to work with graphs. Directed graph models can be used as a basis for developing such representations due to their ability to capture the details of dependencies across space. By using this modelling, we are able to represent the random nature of the traffic. By analogizing the traffic flow to a diffusion process, we are able to train the RPC traffic modelled on H with the SGADCI model using a diffusion convolutional technique. This action may be thought of as a random hop from one node to the next in the graph H. In addition, the dynamic flow of traffic between a node and its neighbours may be represented as a weighted population of infinite random walks across H. To ensure that the model is able to consider data from both upstream and downstream sources, we also implement a diffusion process in the opposite direction. To calculate the resultant diffusion convolution layers over the transportation attribute matrix
Y, we use the given H.

$$X_{*H}Y = \sum_{e=0}^{l-1}(X_0(E\bar{0}^1B)^e + X_{\emptyset}((E^{-1}B)^e)Y$$

where l is the maximum allowable number of steps in the diffusion process and B is H's adjacency matrix. The in- degree ($E1$) and out-degree ($E0$) diagonal matrices allow for learning from both upwards and downwards traffic, correspondingly; $X0$ and $X1$ are the trainable filtering used to the multimodal diffusion process. $E1$and $E0$ symbolize the in-degree ($E1$) and out-degree ($E0$) orthogonal matrices.

Algorithm 1: Matrix Creator

Input: O number of remote procedure calls
Output: the matrix B

1. **Initialize an empty matrix B.**

2. Define shape of the matrix as $size(O) * size(O)$
3. for j = 0 to size(O) do
for k = 0 to size(O) do

if($source(O[j] == O[k])||destination(O[j] == O[k])$ then
$W[j,k] = W[k,j] = 0.5$
if
($source(O[j]) == (destination(O[k]))$ then
$W[k,j] = 1$
if($destination(O[j]) == source(O[k])$ then
$W[j,k] = 1$
end

end

We employ an RNN's encoder-decoder architecture to make use of the SGADCI model and so capture the long - term dependencies of something like the microservices traffic. The RPC communication matrix anywhere along time series are read by the encoder and converted into a vector form. The decoder consults this vector in order to foretell the likely traffic production of future time steps, based on the training data. This encoder- decoder architecture is based on a simplified but well- defined variation of a recurrent neural network called the Gated Recurrent Unit (GRU). By exchanging the GRU's matrix multiplication functionality for the described diffusion convolution operation, we are able to simulate spatial and temporal dependencies together in our work. Dissemination Recurrent neural Fully Convolutional Unit is the result of this process. To complete the SGADCI, these units were layered in a sequential order to produce a series of layers. We can see the encoder-decoder structure that forms the backbone of the SGADCI.

$$s^u = \rho(X_{s*H}[Y_u, i_{u-1}] + c_s)$$

$$v^u = \rho(X_{v*H}[Y_u, i_{u-1}] + c_v)$$

$$d^u = \delta(X_{d*H}[Y_u(s_u \odot i_{u-1}] + c_d)$$

$$i_u = v^u \odot i_{u-1} + (1 - v^u) \odot d_u$$

where Yu , iu, su, vu, and du are the refresh gate, upgrade gate, and cell state at time step u; and Yu, iu, su, vu, and du are the input traffic network matrix and the desirable outcome at time step u. For each equation, we use filters denoted by the letters Xs, Xv, and Xd,, where H stands for the stated diffusion convolution operation. RNN layers are constructed with this cell. The SGADCI model may now be trained with sequential input and long-term dependencies can be captured thanks to the addition of these layers.

IV. Results and Discussion

In this part, we detail DeathStarBench, the microservice architecture platform we chose for our work, the software environment and tools we utilised to instrument our application, the results of our experiment, and the software libraries and hardware we employed. Our key contribution is a set of penetration testing tests in which we mimic three cyberattacks on the application. Distributed tracing was used to keep track of the traffic between the many microservices involved in each experiment, and the resulting data sets represented the activities of the various microservices-based applications. The established threshold-based anomaly detection approach was then utilised to detect the cyber-attacks after the SGADCI model was trained and assessed using the data as provided

We find that each attack consists of a high amount of simultaneous API calls, therefore we may identify a group abnormality by calculating the estimated traffic within a narrow time range as shown in Fig 3. Our experiment made use of DeathStarBench, a free and open-source benchmark suite that includes a number of different kinds of microservices apps. A number of complete software packages are currently on the market, including ones for online communication, finance, entertainment (including user- generated content like film reviews), and hotel booking. We choose to use SocialNetwork, a popular social networking programme,

for this study. The DeathStarBench suite was developed using a wide variety of languages, including but not limited to Python 3.7, node.js, Java, JavaScript, PHP, C, and C++.

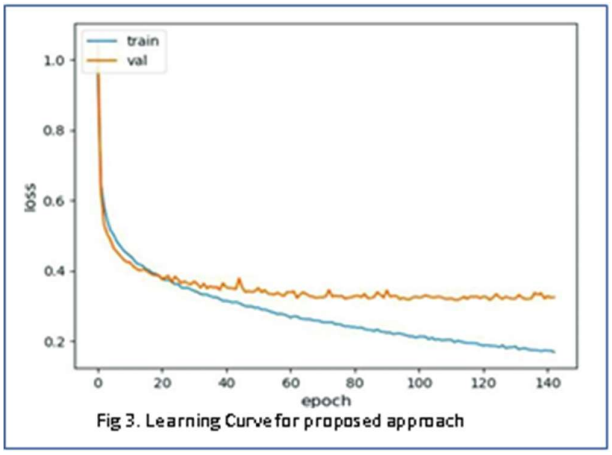

Fig 3. Learning Curve for proposed approach

Social networking programme that simulates a broadcasting network based on registered individuals and their follow connections with one another. The following operations are supported by this social networking app: a person who has signed up for the service and is logged in, who has uploaded a post that may contain text, hypertext links, or other information, who has broadcasted that post to other users, who has viewed the activity of other users, and who has been recommended as a user to follow. Whenever a user on the client side of the social network application makes an HTTP URL request, Nginx's load balancer and web browser module handle it. In order to communicate with the app's logic, the load balancer can receive the following API calls.

TABLE I. PREDICTION ERROR ANALYSIS FOR PASSWORD GUESSING ATTACKS

Prediction Error for Pwd guessing attacks (%)			
Nodes	CNN	GNN	SGADCI (Proposed)
100	29.63	20.12	12.56
200	34.35	22.54	13.56
300	32.52	32.01	18.65
400	31.52	28.65	16.52
500	29.65	30.02	13.52

Web requests are sent off to the relevant microservice by the web server. Information storage, search queries, as well as other services can be relied upon at a later time to facilitate various processes. For long-term data storage, the application's backend server relies on MongoDB; for caching, the server stores Memcached and Redis. In our study, we ran the social networking software three times, each time subjecting it to a new form of cyber-attack. All experiments had both "normal" application traffic and "abnormal" traffic due to "seeded" cyberattacks. The majority of the normal application traffic was made up of RPC traffic that was returned in response to API queries used to upload user-submitted content to the application.

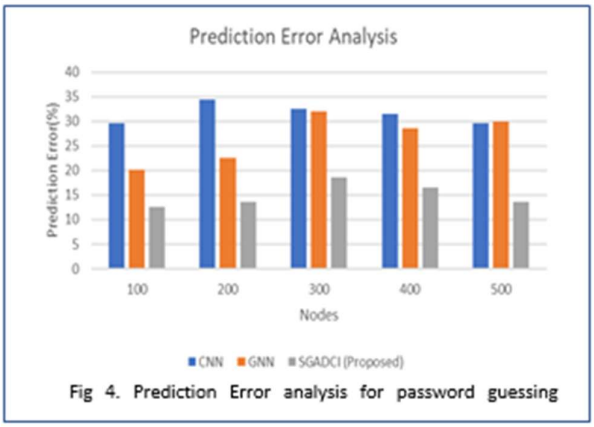

Fig 4. Prediction Error analysis for password guessing

The traffic also contains RPC calls for purposes such as user registration, logging in, and perusing the timelines of other users. Timestamped, distributed traces of microservice RPC communication sampled over 2 minutes were created synthetically for each experiment. The generated datasets included around 18,500 microservice calls. We were able to parse out 63 distinct microservice origin-destination pairs from these datasets. Following this, we calculated the traffic for every RPC node in each time frame. Table 2 summarizes the findings gained using 85% of the data as a training set and 15% as an evaluation set. We examined RPC traffic within the testing data set time window in which the cyberattacks were seeded to identify them.

TABLE 2. PREDICTION ERROR ANALYSIS FOR DDOS ATTACKS

Prediction Error for DDoS Attacks (%)			
Nodes	CNN	GNN	SGADCI (Proposed)
100	11.23	15.24	9.35
200	15.35	14.65	9.82
300	12.36	17.89	8.63
400	10.32	16.52	8.21
500	9.36	16.34	8.65

A batch registration attack is malicious software in which a hacker generates a huge number of dummy user accounts, sometimes known as bot accounts. They can be used for benign objectives like artificially inflating a page's "likes" count, or they can be used for harmful purposes like propagating malware on a system application, hacking services, and fraudulent online activities to affect public opinion. According to scientific research, a good way to prevent bulk user account creation from the same IP address, which is characteristic of a bot attack, is to create a baseline of typical application activity and then monitor aberrant requests. Any permutation of characters, digits, and special characters on the keyboard can be used in a brute force attack to guess a password.

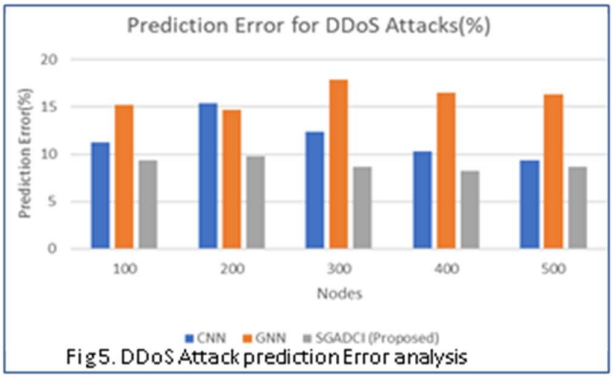

Fig5. DDoS Attack prediction Error analysis

With this method, hundreds of login requests are made per second. In this experiment, we introduced a brute-force attack into the testing data set within a single observed time frame. This malicious software will make ninety-two failed login attempts by using every key on the keyboard as if it were a password shown in as displayed in Fig 4 and Table 1.The resultant prediction error for a sample of eight RPC pair nodes at u is shown. These nodes represent both the random RPCs and the irregular traffic characteristic of a brute-force attack, as well as some of the more typical nodes involved in the application's capabilities, such as the ability to construct and update user posts. We also investigate distributed denial-of-service (DDoS) attacks as a third type of cyber- threat in this paper.

The goal of this type of attack is to make a service inaccessible to genuine customers by executing so many requests that they exhaust the system's resources. There are three distinct forms of distributed denial of service attacks: volumetric, protocol, and application layer. Application-layer attacks are those that may be transmitted to web servers via the Hypertext Transfer Protocol (HTTP), and their severity is quantified by the number of requests they receive per second. Our project's DDoS attack was an HTTP Flooding attack, which hits at the level of both web servers and applications. A denial- of-service attack is a type of cyber intrusion that involves sending a large number of HTTP requests, often GET or POST requests Our method minimises the prediction error for each node in real time to arrive at the threshold value for detecting anomalies as shown in fig 5. At the moment of testing, while the trained SGADCI model is live, it can be subjected to a second type of adversarial machine learning attack that compromises the model's integrity. An anomaly detector has been set up, and an adversary now hopes to manipulate its behaviour by altering legitimate ground truth data such that they no longer exceed the configured threshold. Countermeasures can be implemented to lessen the impact of adversarial attacks, such as smoothing the model's decision limits.

V. Conclusions

This research shown that a microservices-based application's polylithic behaviour can be represented as a microservice web application, and that a decentralized tracking mechanism can be used to keep tabs on API calls made by users. A state-of-the-art graph convolution network, the Stochastic Generative Adversarial Deep Convolution Infrastructure (SGADCI), is offered as a means of learning the microservice traffic and uncovering the geographical and temporal connections within the data. Our goal in doing this traffic forecasting was to anticipate microservice traffic in the future. To gauge the SGADCI's efficacy, we used threshold-based anomaly detection to look for unusual behaviour in microservices that may have been caused

by cyber-attacks. This paper builds upon our prior work showing how decentralized monitoring can be used to identify cyber security attacks in microservices.

VI. List of Abbreviations

GDP	Gross Domestic Product
API	Application Programming Interface
GCNN	GraphConvolutional Neural Network
SGADCI	Stochastic Generative Adversarial Deep Convolution Infrastructure
RPC	Remote Procedure Call
DNN	Deep Neural Networks
RNN	Recurrent Neural Networks
LSTM	Long Short-Term Memory
CNNs	Convolutional Neural Networks
CLTFP	CNN And LSTM To Forecast Future Traffic Flow
CNN	Convolutional Neural Network
GNNs	Graph Neural Networks
STGNNs	Spatial-Temporal Graph Neural Networks
PWD	Password
TDG	Traffic Dispersion Graph
STDeep Graph	Spatio-Temporal Deep Graph
HTTP	Hypertext Transfer Protocol
GAD	Group Anomaly Detection
MSA	Microservice Architecture
GRU	Gated Recurrent Unit
URL	Uniform Resource Locator
DDoS	Distributed Denial-Of-Service
REST	Representational State Transfer

VII. Declarations

7.1 Availability of data and materials

The dataset utilized in this study was obtained from the "Kaggle.com".

7.2 Competing Interests

We the authors declares that we have no competing interests.

7.3 Funding

We declare that we did not receive any specific grant from funding agencies in the public, commercial, or not for profit sectors.

7.4 Authors' Contributions

KE1, has made a significant contribution to the journal's concept, acquisition, and analysis. Dr.DB2 Supervises the result and conclusions. As the corresponding author of the manuscript, I hereby declare that both authors has read and approved the final manuscript.

7.5 Authors' Information

KanniappanElumalai1 Research Scholar, Department of Computer Science and Engineering, Academy of Maritime Education and Training (AMET University).
Dr.DuraimutharasanBose2 Professor, Department of Computer Science and Engineering, Academy of Maritime Education and Training (AMET University), 135, SH 49, Kanathur, Tamil Nadu 603112Chennai-India
email: hodcs@ametuniv.ac.in

7.6 Acknowledgements
We would like to thank and acknowledge the support offered by the supervisor and faculty of Computer Science and Engineering, Academy of Maritime Education and Training (AMET University), for their contributions to the experiments in the Computing center.

VIII. References

[1] Akhavan-Hejazi, H., Mohsenian-Rad, H., 2018. Power systems big data analytics: An assessment of paradigm shift barriers and prospects. Energy Rep. 4, 91–100.
[2] Beechey, M., Kyriakopoulos, K.G., Lambotharan, S., 2021. Evidential classificationand feature selection for cyber- threat hunting. Knowl.-Based Syst. 226,107120.
[3] Cao, Y., et al., 2019. A topology-aware access control model for collaborative cyber–physical spaces: Specification and verification. Comput. Secur. 87,101478.
[4] Damon, E., et al., 2014. Cyber security education: The merits of firewall exercises.In: Akhgar, B., Arabnia, H.R. (Eds.), Emerging Trends in ICT Security. Morgan Kaufmann, Boston, pp. 507–516.
[5] Edgar, T.W., Manz, D.O., 2017. Science and cyber security. In: Edgar, T.W.,Manz, D.O. (Eds.), Research Methods for Cyber Security. Syngress, pp. 33–62.
[6] Furnell, S., et al., 2020. Understanding the full cost of cyber security breaches.Comput. Fraud Secur. 2020 (12), 6–12.
[7] Huang, J., et al., 2020. Secure remote state estimation against linearman-in-the-middle attacks using watermarking. Automatica 121, 109182.
[8] Kim, Y.S., et al., 2020. Development of a method for quantifying relativeimportance of NPP cyber attack probability variables based on factor analysis and AHP. Ann. Nucl. Energy 149, 107790.
[9] Li, J., Sun, C., Su, Q., 2021. Analysis of cascading failures of power cyber–physicalsystems considering false data injection attacks. Glob. Energy Interconnect. 4 (2), 204–213.
[10] Marefati, M., Mehrpooya, M., Shafii, M.B., 2018. Optical and thermal analysis of aparabolic trough solar collector for production of thermal energy in different climates in Iran with comparison between the conventional nanofluids. J. Cleaner Prod. 175, 294–313.
[11] Nicholson, A., et al., 2012. SCADA security in the light of cyber-warfare. Comput.Secur. 31 (4), 418–436.
[12] Ogbanufe, O., 2021. Enhancing end-user roles in information security: Exploringthe setting, situation, and identity. Comput. Secur. 108, 102340.
[13] Patel, D.C., et al., 2021. Paradoxical motion on sniff test predicts greaterimprovement following diaphragm plication. Ann. Thorac. Surg. 111 (6),1820–1826.

[14] Qiu, W., et al., 2021. Time-frequency based cyber security defense of wide-areacontrol system for fast frequency reserve. Int. J. Electr. Power Energy Syst.132, 107151.
[15] Shamel, A., et al., 2016. Designing a PID controller to control a fuel cell voltageusing the imperialist competitive algorithm. Adv. Sci. Technol. Res. J. 10 (30).
[16] Tan, S., et al., 2021. Attack detection design for dc microgrid using eigenvalueassignment approach. Energy Rep. 7, 469–476.
[17] Zhao, Z.-g, et al., 2021. Control-theory based security control of cyber–physicalpower system under multiple cyber-attacks within unified model framework. Cogn. Robot. 1, 41–57