IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Framework for Network Topology Generation and Traffic Prediction Analytics for Cyber Exercises

DONG-WOOK KIM[1], GUN-YOON SHIN[1], YOUNG-HOAN JANG[1] SEUNGJAE CHO[2], KWANGSOO KIM[2], JAESIK KANG[2] and MYUNG-MOOK HAN[1]

[1]Department of AI Software, Gachon University, Sungnam-si 13120, South Korea
[2]Cyber Electronic Warfare R&D, LIG Nex1, Sungnam-si 13488, South Korea

Corresponding author: Myung-Mook Han (mmhan@gachon.ac.kr)

This work was supported by the Korea Research Institute for Defense Technology Planning and Advancement (KRIT) grant funded by the Korea government's Defense Acquisition Program Administration (DAPA) (No. KRIT-CT-21-037, Artificial-Intelligent Cyber Training System, 2021), and the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea (RS-2023-00211871).

ABSTRACT Today's cyber-attacks have become increasingly sophisticated and diverse, targeting systems that hold sensitive information, creating the need for continuous cyber exercise and skill development for cyber professionals. Because cyber exercises require training activities and environments that can support a variety of situations, significant technological efforts have been made to build training environments. In line with technological trends, current cyber exercise simulations are being studied to create various cyber scenarios that can help build an intelligent cyber battlefield using big data and artificial intelligence (AI). This requires a large amount and different types of data for learning, as well as a technical system that can manage and update them periodically. The objective of this study is to develop network topology generation and traffic prediction technologies based on intelligent network traffic analysis and AI models for cyber exercise technology systems. To automate training network scenarios, a path generation technology based on graph theory was developed, and the network environment was analyzed based on the amount of transmission by building a software-defined network capable of analyzing and predicting network traffic. A comparison of AI models such as long short-term memory (LSTM), bidirectional LSTM (BiLSTM), and gated recurrent units (GRU) to predict the amount of transmission showed good performance, with BiLSTM showing a better prediction error. The proposed methodology provides insights that can be used to adjust training scenarios during the network design and operation phases, which is expected to help manage the network, increase efficiency, and address security issues.

INDEX TERMS Cyber exercises, network topology, SDN networking, traffic matrix.

I. INTRODUCTION

With the rise of today's cyber threats, there is a growing demand for specialized cybersecurity professionals in universities, corporations, and the military. Cyber exercises serve to enhance the capabilities of the cybersecurity workforce by applying environments and technologies to educate and train them [1]. Globally, the development of cyber exercise environments is a major focus, most notably the National Cyber Range (NCR), first developed by the Defense Advanced Research Projects Agency (DARPA) [2].

A cyber range is a tool that simulates cyber-attacks to train cybersecurity professionals to respond to them. Most cyber ranges are limited in scope to meet specific requirements, and their command-and-control systems are designed to simulate cyber-attacks and defenses based on training scenarios. These cyber exercises provide a realistic network environment that enables highly realistic simulations. However, maintaining such a high-performance environment requires multiple resources and significant costs, including computing hardware, management software, and administrative time. There are several factors to consider when building a cyber exercise environment. To simulate virtual cyber threats in real time and communicate information among different users, the network topology and configuration must be modeled to reflect the actual network environment, which requires high

communication capacity. In addition, providing an independent lab environment for each user requires significant high-performance computing resources and capabilities [3].

Building a cyber exercise environment requires specialized technical staff to design the training environment and manage network and computing resources. With the growing importance of realistic cyber exercises, network emulators based on software-defined networking (SDN) are increasingly being used [4]. These emulators effectively create a variety of network training environments and enable users to improve their ability to manage and respond to cyber-attacks, especially in complex areas such as smart grids [5].

Scenario creation is a key task in cyber training that requires significant experience, time, and cost because it considers the configuration of the network topology and the resources required to operate the system. Most training systems organize scenarios based on static and consistent patterns, but this approach cannot effectively respond to unexpected cyberattack patterns or complex network environments. To provide participants with new experiences and different patterns in cyber exercises, managing unpredictable traffic segments such as traffic delays and rates of change is key. Therefore, we have developed a system for regulating and predicting different traffic patterns based on SDN. We create new patterns using a graph-based approach to extract hosts with large traffic concentrations and use deep learning modeling to provide insights into how these new patterns can be incorporated into the training network to create an enhanced training environment. We design and build network topology scenarios using a graph-based approach to coordinate network packet routing and collect traffic from SDN-based network environments to build a traffic matrix and provide a predictable data set. The traffic matrix represents the amount of traffic transferred between each node in the network, allowing you to understand and analyze the network traffic distribution process in detail. Leveraging this information, deep learning models can predict the volume of network traffic and provide insights for refining training scenarios. Therefore, we propose an effective method for controlling the flow of network traffic and adapting to changes in the network environment. The main contributions of this study are as follows:

(1) Uses a graph-based approach to extract hosts with large traffic concentrations and generate new patterns.

(2) Presents a methodology to compute the traffic matrix resulting from the network topology of an SDN environment and introduces a methodology to analyze the transmission volume by timestamp.

(3) Predicts the transmission volume of network traffic through regression models of LSTM, BiLSTM, and GRU based on deep learning models in the training network, and provides insights reflected in the network.

Using the proposed methods, based on a known network dataset, we can form a new network topology by combining frequently used overlapping network paths. By designing a traffic matrix to understand and analyze the amount of traffic transmitted by each node in the network, the amount of network traffic can be predicted using an artificial intelligence (AI) model. This information provides insight into adjusting the training scenarios, suggesting ways to effectively control the flow of network traffic and flexibly respond to the environment.

II. RELATED WORK

In this section, we analyze previous studies related to expert knowledge for cyber exercises. We focus on studies that have proposed techniques for creating training traffic scenarios that reflect the flow and patterns of network traffic, as well as SDN used to simulate training environments.

A. BACKGROUND AND TECHNOLOGY OF CYBER EXERCISES

Cybersecurity exercises are traditionally organized in groups of equal experts to assess participants and compare their performance to mitigate situations and train professionals. The best-known cyber defense exercises (CDX) propose common security requirements for building a secure environment in cyberspace and a training cycle to implement them [6]. To conduct cyber exercises, it is necessary to provide a variety of threat scenarios. In this process, the concept of the NCR was first developed by DARPA as a case study for joint military-civilian cybersecurity exercises [2]. Cyber range emulates complex network settings in an isolated environment and organizes the environment and procedures to safely perform cybersecurity tasks on assigned hosts or network infrastructure [7]. Running cyber range requires a network scenario based on the training objectives. This scenario provides the network vectors needed to simulate traffic and supports [8].

Currently, two types of traffic regeneration and traffic generation methods are being explored for the next generation of cyber range to provide network traffic in the background [9]. Traffic replay technology adjusts the collected traffic data and replays the traffic data again, while traffic generation entails building a mathematical model to generate new traffic. Du *et al.* [10] found that traffic replay can provide statistical data on the network based on a variety of usage patterns and trends, but the efficiency may be reduced due to the large number of IP mappings; and the traffic generation method generates new traffic by learning the characteristic distribution of traffic in the actual network, but it contains a lot of redundant data, or the traffic method should be considered. From this perspective, for efficient scenario operation for cyber exercises, we focused our research on the following detailed technical studies to pattern specific scenarios of training traffic.

B. CYBER EXERCISE SCENARIO BUILDING AND GRAPH-BASED TECHNIQUES

Depending on the network infrastructure environment, cyber exercise scenarios are typically generated by users adjusting traffic flows and patterns or by deep learning by optimizing routing policies. In general, researchers have proposed generalizations of automatic modeling to combine abnormal events with normal traffic. Park *et al.* [11] presented the problem that although dedicated cyber ranges resembling real-world environments have been built, real-world cyber warfare is an integrated and organic environment, which leads to poor interoperability, unlike constructed cyber ranges. To address this problem, a configuration plan and operational functions for building a multifaceted cyber range reflecting the characteristics of each attack group were proposed and tested for their impact on distributed denial of service (DDoS) attacks. The study presented a practical cyber range structure and scenario construction plan, and with each battlefield management system as a distributed center, the servers were synchronized during the simulation to coordinate the battlefield situation through synchronization among centers. For scenario construction, a configurable network topology was designed, and normal and abnormal traffic distribution plans were generated based on learning and testing information. A separate scenario database was created to improve resource sharing, connectivity, and reusability.

Protogerou *et al.* [12] reported that the number of interconnected edge devices has significantly increased due to the proliferation of the Internet of Things, and graph-based anomaly detection solutions are gaining attention as methods to prevent network anomalies, such as assuming cohesion between related entities and modeling interrelationships. In response to the characteristics of cyber-attacks such as DDoS, a distributed detection method that efficiently monitors the entire network infrastructure to detect abnormal events has been proposed. To improve the notification process, the study proposed a method to represent a node's vector as an aggregated and transformed feature vector of its neighboring nodes using a graph neural network (GNN). By analyzing existing studies, the working principle of the proposed method was explained by utilizing the function of the GNN to isolate individual inputs and pass messages between graph nodes, proving that it is an ideal method.

Chen *et al.* [13] noted that the rapid development of user applications has led to a continuous increase in network traffic, raising the issue of routing optimization. They presented the problem that deep reinforcement learning (DRL), a traditional routing optimization solution, could not handle graph-like information in the network topology due to difficulties in generalization when the topology changes. Therefore, an autoGNN, which combines GNN and DRL, was proposed to automatically generate routing policies, and the proposed system was validated by testing. In the study, Markov modeling was used to construct the interaction process, and function approximation was used to construct a scenario to represent the deep neural network (DNN) of the DRL agent.

Rebecchi *et al.* [14] suggested the need for cyber exercises because cyber-attacks are becoming more widespread and sophisticated, and from a security perspective, the widespread use of virtualization could lead to infrastructure manipulation and extensive damage by drastically increasing the number of exposed attack vectors. The study defined a methodology for monitoring technical key performance indicators for system performance and efficiency, effectiveness of training and learning modules, and perceived quality of experience metrics using the SPIDER cyber range for incident response exercises and reverse engineering.

Barsellotti *et al.* [15] argued that the DDoS detection system based on conventional machine learning and deep learning techniques is not flexible enough to respond to different networks and traffic. In this study, the traffic data was transformed into the flow-to-traffic graph (FTG) structure, and the flow-level graph was processed in GNN to generate representation vectors to output the final prediction results for each flow. The CIC-IDS2017 dataset was used, and the performance of the approach was evaluated to illustrate the improvement caused by using the proposed approach compared to existing methods.

C. SDN AND TRAFFIC PREDICTION TECHNIQUES

SDN is an important research topic for organizing network environments for cyber exercises. In the area of cyber exercises, research is actively conducted on attack scenarios, exercise methods, and measurement of exercise effectiveness. Research is being conducted to make various deep learning and machine learning methods easily accessible for traffic prediction using SDN simulators.

Software-defined networking (SDN) is an approach to networking that uses software-based controllers or application programming interfaces (APIs) to generate traffic on the network and communicate with the hardware infrastructure [16]. Simulators that support SDN include Network Simulator 3 (ns-3) [17], Mininet [18], and the Objective Modular Network Testbed in C++ (OMNeT++) [19]. NS-3 is a popular free and open-source simulator for network research, with advantages in analyzing and visualizing results for different network protocols; Mininet allows system development, sharing, and experimentation, especially advantageous for real hardware deployment, and can create realistic virtual networks running real kernels, switches, and application code on a single machine (VM, cloud, or native) in seconds with a single command. OMNeT++ is a modeling framework developed as an independent project to support sensor networks, wireless ad-hoc networks, Internet protocols, performance modeling, and more. Each simulator has its own strengths and weaknesses, and it is important to choose the right tool to support simulation testing of cyber exercises.

Researchers are working on individual simulator studies to develop cyber exercise capabilities and combine them with learning models to analyze network traffic predictions. Leon *et al.* [20] proposed a practical cyber exercise

framework known as ADLES. It is designed to simplify and streamline exercise sharing and collaboration by providing tools for specifying outcomes for scenarios, networks, and computing resources in hands-on cyber exercises. Ullah *et al.* [21] proposed an SDN-enabled framework for efficient detection of cyber-attacks, including multi-class attacks, based on network flows. In the process, the study introduced approaches for deep learning models and focused on evaluation. Azzouni *et al.* [22] presented NeuTM, a framework for network traffic matrix prediction based on long short-term memory recurrent neural networks (LSTM RNNs). An input methodology for the traffic matrix was presented, focusing on the suitability of the LSTM model for traffic prediction. Torres *et al.* [23] proposed an SDN framework to test the quality of service (QoS) of the mechanisms and introduced a scheme for replaying and modifying packet capture (PCAP) files.

While these studies have introduced SDN technology and methodologies for network design in cyber exercises, a more sophisticated approach to learning and predicting patterns in network traffic is needed by consolidating these studies and technologies. In this context, my contribution focuses on the generation of concentrated traffic patterns and the prediction of SDN simulation traffic using LSTM, BiLSTM, and GRU models. This innovative approach not only fills the existing gaps, but also provides deeper insights into traffic behavior in SDN environments.

III. PROPOSED METHOD

In this section, we propose a training network topology generation and traffic analysis framework for generating cyber exercise traffic distribution scenarios. The framework consists of four main steps: available data, its combination with network topology generation, network traffic generation, and network performance evaluation.

A. GENERAL FRAMEWORK

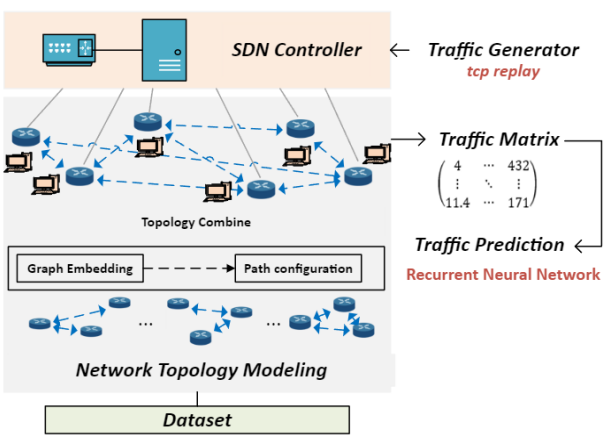

FIGURE 1. Network topology creation and simulation framework.

We implement a network topology for new traffic generation patterns in the training network and propose a predictive network simulation. We extract traffic-intensive hosts from existing network datasets and use graph algorithms to combine them. To illustrate the stages of the framework shown in Fig. 1, a traffic dataset was first collected from a real network environment. Two types of data, actual raw packet files from the CIC-IDS2017 dataset and processed network data for machine learning, were used. In the second step, we proposed a network topology generation model that generates host paths based on Dijkstra's algorithm for traffic distribution scenarios. In the third step, an SDN environment configuration method was proposed to combine the generated network topologies to model a virtual network environment. SDN provides the flexibility to control the network topology and traffic flow to create a training network environment. In this environment, traffic is generated according to the network topology to create the traffic matrix. The traffic matrix captures all traffic flows within the network and provides data to analyze the performance of the network. In the final step, the network performance is evaluated by three models: RNN-based LSTM(Long Short-Term Memory), BiLSTM(Bidirectional Long Short-Term Memory), and GRU(Gated Recurrent Unit) to predict the network traffic transmission amount for the traffic matrix. Predictive evaluation allows us to optimize performance and prevent potential network problems.

B. INTRODUCTION TO DATASETS

For data collection, this study used CIC-IDS2017 [24], an intrusion detection system dataset provided by the Canadian Cyber Security Institute at the University of New Brunswick, Canada. This dataset contains PCAP data collected from an emulated environment over a five-day period and a comma-separated value (CSV) dataset file generated by CICFlowMeter [25]. These two types of datasets serve different purposes. The CSV file provides the information needed to construct the network topology, while the original PCAP data is used to generate traffic in an SDN environment. The following types of datasets we used are presented in Table 1.

TABLE I
AVAILABLE DATASETS

Types	CIC-IDS2017
CIC-DIS2017 dataset (CSV)	Monday, normal activity, 11.0 G Tuesday, attacks + normal activity, 11 G Wednesday, attacks + normal activity, 13 G Thursday, attacks + normal activity, 7.8 G Friday, attacks + normal activity, 8.3 G
CIC-DIS2017 dataset (CSV)	PCAP's matching host pairs were reconstructed for 14 high-frequency host pairs, based on the network topology configuration, for distribution.

C. DATASET PREPROCESSING

Before constructing the network topology, we pre-processed the data in CSV file format to obtain statistics between source and destination IPs for the dataset. Several key factors were considered to keep the data consistent and

accurate. First, data with duplicate attributes or row values were removed. As these duplicates led to overfitting of the data and reduced the generalization ability of the model, they were removed, as were data samples with missing values. Missing values compromise the completeness of the data and can bias the results of the analysis. Finally, any negative values in the data set were also removed. Negative values are generally not allowed as they may indicate errors in the data.

Preprocessing converted the original dataset of 85 attributes into a data set with only 66 attributes, representing each data sample with more concise and unambiguous attribute values while preserving the original label values.

D. NETWORK TOPOLOGY CREATION

Network topology is created after data preprocessing. To create a network topology, the network flow is analyzed based on the 4-tuple (source IP, destination IP, source port, and destination port) attribute information of the network. Because the transfer of network packets occurs between a client and a server, it is represented by embedding a port pair in a graph that represents the connection between the source and destination ports. The graph is represented in the form $G = (V, E)$, where V and E represent the vertices and edges, respectively. The vertex consists of a unique pair of source and destination ports in a network traffic dataset, represented as $V = \{(SPort_1, Dport_1), (SPort_2, Dport_2), (SPort_3, Dport_3)$, and $(SPort_n, Dport_n)\}$. The edge represents the transmission of a packet from a given source port to a destination port, which is defined in the form $E = \{(SPort_n, Dport_n, Packet_n)\}$, where each $Packet_n$ represents a connection between the given source and destination ports, that is, a connection between a pair of vertices $(SPort_n, Dport_n)$. Each edge contains various attributes that characterize the packet. These attributes can include packet size, transmission time, and protocol type and can be used to effectively graph the patterns and structure of network traffic.

In the defined graph, the directional flow along the vertices is organized in terms of source and destination ports, which requires the definition of the start and end nodes of the vertices. To determine them by the in-out degree, this study satisfies the following conditions:

- The start and end vertices are denoted by s and t, respectively.
- V and E denote the set of vertices and edges in t he graph.
- In-degree(v) is the number of vertices entering v ertex v, and out-degree(v) is the number of verti ces leaving vertex v, as defined by the following equations:

$$s \in V \text{ and } In\text{-}degree(s) = 0 \quad (1)$$

$$t \in V \text{ and } Out\text{-}degree(t) = 0 \quad (2)$$

A path-finding algorithm is used to generate a topology from a graph consisting of vertex pairs of start and end vertices. The shortest path algorithm Dijkstra [26] is used, which is very effective in finding a minimum weight path between two vertices in a directed graph. The Dijkstra algorithm explores all paths between the end nodes relative to the start node in the entire network. The steps of the algorithm are as follows:

Algorithm 1: Shortest path search using Dijkstra's method

```
DIJKSTRA(G,w,s)
# Select the start node, s∈V and in-degree(s)=0
1.INIT-SINGLE-SOURCE(G,s)
# Initialize the list to store visited nodes, t∈V and out-degree(t)=0
2.S <- NULL
3. Q <- V[G] # Initialize the queue to store unvisited nodes
4. while Q != NULL # Repeat until all nodes are visited
5. do u <- EXTRACT-MIN(Q) # Select the node with the least weight in the queue
6. S <- S + U # Add the selected node to the list of visited nodes
7. for each vertex v <- Adj[u] # Tour all nodes associated with the selected node
8. do RELAX(u,v,w) # Update the weights of nodes associated with the selected node
```

The resulting network topology is very useful for visually understanding the structure and flow of the network, enabling analysis of how data flows through the network, which nodes handle the most traffic, which paths are most efficient, and so on. This topology can also be used as a basis for optimizing network performance and responding quickly to failures.

E. NETWORK EMULATORS AND TRAFFIC REPLAY

For network topology configuration, PCAP data modification, and traffic replay for traffic data distribution, we combined the Mininet emulator, Tcpreplay, and Bittwiste to implement an efficient cyber exercise traffic distribution simulation. Mininet [18] is an SDN that enables the creation of components such as virtual routers, switches, and hosts on a single system, simulation of their management, and testing of network protocols and applications without physical hardware. Network behavior is monitored by a POX controller [27], which provides centralized network management capabilities to manage and control network flow.

A network traffic replay tool known as Tcpreplay [28] was used along with the Mininet emulator to randomly generate network topologies and traffic from the hosts. Using the actual network traffic data contained in the PCAP file, the tool can recreate traffic scenarios, reproduce network traffic conditions, and evaluate the performance of the system. To reproduce the traffic, the Bittwiste [29] tool was used to modify the data in the PCAP file to match the host environment of the generated topology. The tool can be used to implement and test different situations depending on the training scenario, as it can modify information such as the IP address and timestamp of each traffic unit.

In a simulation environment implemented in this way, the Tcpreplay and Bittwiste tools are used to manipulate and replay the original PCAP data in the network topology created by Mininet to establish the network topology configuration environment. The rest of the detailed topology configuration environment consists of the configuration environment shown in Fig. 2.

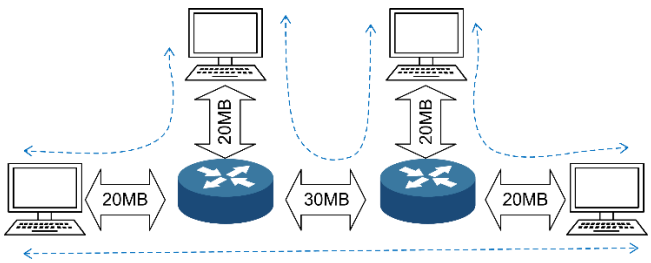

FIGURE 2. Mininet network topology configuration scenario.

Each host and router has a maximum transmission bandwidth of 20 MB, and the router-to-router transmission bandwidth is 30 MB, which provides sufficient bandwidth for communication between the various elements of the network while maintaining a balance to ensure effective information flow between the various elements of the network. These bandwidth settings also help to simulate the various traffic situations that may occur in a real-world network environment, which is an important configuration factor when building different scenarios for cyber exercises. By integrating these three tools, an effective simulation system was designed and implemented to reproduce and analyze cyber exercise traffic distribution in a realistic network environment.

F. NETWORK EMULATORS AND TRAFFIC REPLAY

The network traffic captured by the monitoring is stored in the form of a traffic matrix, which contains comprehensive information about the overall traffic flow within the network [30]. The traffic matrix is computed by the POX controller of the Mininet emulator to distribute packets in the set network topology and to observe the amount of traffic. A traffic matrix generally represents the amount of traffic exchanged between each pair of nodes in a network during a given time period, and the varying traffic can be used to analyze the performance of the network or predict patterns.

To calculate the variation in traffic, the difference between the two-time measurements is determined and used to calculate the difference between the current and previous traffic conditions. This difference represents the variation in traffic, which is divided by the elapsed time to obtain the *change in traffic per hour* (ΔT). That is, ΔT can be calculated by subtracting the current traffic conditions from the previous traffic conditions and dividing the difference by the elapsed time (duration):

$$\Delta T = \frac{Latest\ traffic - Previous\ traffic}{Duration} \quad (3)$$

In (3), the most recent and previous traffic are matrices representing the states of the traffic, and each component of these matrices represents the amount of traffic at a given time. Therefore, the change in traffic for each component can be obtained by calculating the difference between these two matrices. This method makes it possible to identify and reflect changes in network status in real time, which is advantageous for analyzing the performance of a dynamically changing network environment.

G. NETWORK TRAFFIC PREDICTION AND PERFORMANCE EVALUATION

Predicting network traffic plays an important role in managing networks, increasing efficiency, and addressing security concerns. Understanding traffic patterns and predicting future traffic changes can help optimize network resources, avoid traffic bottlenecks, and proactively detect anomalies. Therefore, artificial intelligence models are used to evaluate such analytical models. Predictive models are divided into linear and nonlinear models. Linear models assume that the relationship between input and output is linear, meaning that a small change in the input will be reflected in a small change in the output, indicating a constant rate of change. Although the advantages of such linear models lie in their interpretative power and computational efficiency, they are limited by their inability to capture complex or nonlinear data patterns. In contrast, nonlinear models assume that the relationship between input and output is nonlinear, meaning that a small change in the input can cause a large change in the output, and the rate of change may not be constant. Nonlinear models better capture complicated data patterns and can better reflect complex real-world scenarios. However, they are more computationally intensive, prone to overfitting, and difficult to interpret.

In this study, we used nonlinear machine learning models based on recurrent neural networks (RNNs). RNNs are neural networks that can effectively model sequential data and are particularly useful for problems where order is essential, such as natural language processing, speech recognition, and time series prediction [31]. RNNs have a recursive structure in which the current output depends on previous computations, but they suffer from long-term dependency. The problem with this is that as the sequence gets longer, the ability to retain and learn the initial information deteriorates. Modified RNNs such as long short-term memories (LSTMs) [32] and gated recurrent units (GRUs) have been proposed to address these problems.

LSTMs are a class of RNNs designed to address the problem of long-term dependency by introducing the concept of gates to decide whether to retain or discard information. This allows the LSTM to learn better about long sequences and identify the significant parts of each sequence. BiLSTM [33] processes sequence data in both directions. It uses not only past information but also future information, which is useful when future information is

needed to make accurate predictions. GRU [34] is a simplified version of LSTM that performs as well as LSTM but with better computational efficiency. GRU simplifies the model by combining the two gates of LSTM (hide and discard) into one and combining the cell state with the hidden state.

To effectively train a time series prediction model, the original time series data set is converted to generate input-target pairs at each time point. For given time series data, $\{x_t\}_{t=1}^{N}$, where N represents the total length of the time series. The method for constructing the input feature matrix X and the corresponding target vector Y is as follows.

Each i-input vector X_i consists of consecutive data points during the look-back period L, starting at time i.

$$X_i = [x_i, x_{i+1}, \dots, x_{i+(L-1)}] \quad (4)$$

Where $i = 1, 2, \dots, N - L$. The corresponding target value Y_i is defined as the data point at the next time to be predicted by the input vector X_i:

$$Y_i = x_{i+L} \quad (5)$$

By preparing the data set in this way, the model can use sufficient past information to predict Y_i from each X_i and learn the temporal patterns and structures of the time series data.

These three models are evaluated based on Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and R-squared, which are the differences between the predicted and actual values of network traffic time series data. The RMSE value is a measure of the model's prediction error, with values closer to zero indicating better prediction performance. Since the RMSE is rooted in the mean of the squares of these prediction errors, it is a measure of the average error over all prediction errors. The RMSE is calculated as follows:

$$RMSE = \sqrt{\left(\frac{1}{n}\right)\sum_{i=1}^{n}(y_i - x_i)^2} \quad (6)$$

where n is the number of samples, y_i is the ith observation, x_i is the ith prediction, and Σ is the sum over all samples. One of the main characteristics of the RMSE is that it gives more weight to large errors. This is due to the squaring process, which can cause the model to avoid large errors. For this reason, the RMSE is sensitive to outliers and is compared to the mean absolute error (MAE), which is less sensitive to outliers.

MAE (Mean Absolute Error) is the average absolute error, which converts the difference between the predicted and actual values to an absolute value and then calculates the average. All errors are weighted equally. The MAE is calculated as follows:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (7)$$

MAPE (Mean Absolute Percentage Error) is the mean absolute percentage error, which calculates the percentage of prediction error compared to the actual value and then averages it. Because it is a relative error measure, it can be difficult to use when the actual value is close to 0. The MAPE is calculated as follows:

$$\mathrm{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \quad (8)$$

R-squared is an indicator of how well a regression model explains the variation in the data. A value closer to 1 indicates that the model does a better job of explaining the variation in the data. R-squared is calculated follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (9)$$

RMSE, MAE, MAPE, and R-squared are widely used metrics to evaluate the performance of regression analysis. Because they can be applied to any time series data, such as traffic change rate, they evaluate the overall performance of the model.

IV. EXPERIMENTAL RESULTS

In this section, we present an analysis of the experimental results of generating training traffic topologies using the proposed methodology. The experiment was performed in two steps. First, the Dijkstra algorithm was used to generate graph paths for the network topology. Second, the generated topologies were combined to build a network scenario, send traffic, and calculate the expected throughput of the entire network.

A. EXPERIMENTAL ENVIRONMENT

For the experiment, the environment was set up as shown in Table 2 below.

TABLE II
EXPERIMENTAL ENVIRONMENTS

Component	Specification
CPU	Intel Xeon E5-2620 v4
memory	48 GB
OS	Windows 10, WSL2 (Ubuntu 20.04)
language	Python 3.8
library	networkx, tensorflow, sklearn

The Python language was used on a personal computer to generate the network topology on the running computer. A CSV file of the known CIC-IDS2017 dataset was used to generate a network graph using the NetworkX library. The Mininet emulator was installed in the Ubuntu 20.04 Windows Subsystem for Linux 2 (WSL2) environment to simulate network traffic. The network configuration environment was set up using the PCAP data provided by CIC-IDS2017, followed by the calculation and collection of the traffic matrix.

B. EXPERIMENTAL RESULTS OF NETWORK TOPOLOGY CREATION

In the first experiment, the network topology was generated based on the CIC-IDS2017 dataset using the methodology introduced earlier. The path generation results for the Dijkstra algorithm were checked in the dataset in terms of source and destination ports. The results of verifying the graph configuration information for the CIC-IDS2017 dataset were as follows: it consisted of 14,385 start nodes and 124 end nodes, for a total of 1,783,740 graph paths. The results of the path configuration using the Dijkstra algorithm are shown in Table 3.

TABLE III
RESULTS OF NETWORK TOPOLOGY CREATION

Graph path length	Number of scenarios	Number of redundant paths
7	192	8
6	30	21
5	657	20
4	1,141	25
3	510	3
2	1	0
Total	**2,531**	**77**

These results provide the basis for explaining the number of paths and their redundancy as a function of the length of the graph, which facilitates the understanding of the process of path analysis and removal of redundant topologies.

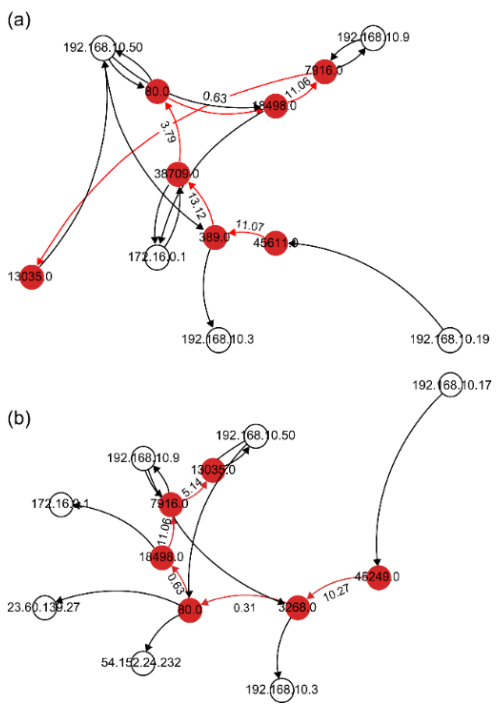

FIGURE 3. Path Graphs of Network Topologies. (a) Path length of 7 (b) Path length of 6.

In Table 3, "graph path length" refers to the total number of communication paths. For example, in Fig 3, if the path length is 7, the graph in (a) indicates that the path consists of seven communication structures. “Number of scenarios” refers to the total number of network topology scenarios for a given graph path length. For a path length of 7, there are 192 different network topology scenarios. “Number of Redundant Paths” refers to the number of scenarios that indicate the presence of redundant paths in the network topology at the same graph path length. For a path length of 7, there are eight redundant network paths. This indicates that, for a given path length, there may be multiple scenarios that follow the same path or have similar traffic patterns.

The number of network traffic scenarios that can be generated for the graph path length, as well as the number of redundant paths, can be determined. This information is important for network simulation and optimization. The number of generated scenarios and redundant paths for the graph path length are summarized as follows: For a graph length of 7, a total of 192 topology scenarios were generated, of which 8 were redundant. For a graph length of 6, as shown in Fig. 3(b), 30 scenarios representing communication paths were generated, of which 21 were redundant. The graph with the largest number of scenarios was the one with a path length of 4, in which a total of 1,141 scenarios were generated with 25 redundant paths. The graph with the smallest path length of 2 generated only one scenario with no redundant paths. This suggests that as the graph path length decreases, the number of possible scenarios decreases, and the probability of overlap tends to decrease.

In general, the results showed that the number of redundant paths (77) was relatively small compared to the total number of scenarios (2,531). This represented approximately 3.04% of all scenarios. These percentages change with path length, with the highest redundancy rate (70%) for a graph path length of 6. This suggests that the scenarios generated at this graph path length followed similar paths or had the same traffic patterns. In contrast, the overlap rate (4.17%) was relatively lower for the graph path length of 7, indicating that a variety of scenarios can be generated from paths of this length.

C. EXPERIMENTAL RESULTS FOR NETWORK TRAFFIC PREDICTION

In this part of the study, we placed 14 hosts by combining graphs based on ports with high communication frequencies in the generated network topology scenario. Simulations were performed to predict the amount of transmission from each host for the entire network. The experimental data were collected by calculating the traffic matrix for approximately 5 d by reconstructing and sending PCAP files provided by the CIC-IDS2017 dataset using a Mininet emulator and POX controller.

To predict traffic patterns for the collected network traffic volume, the dataset was processed using a sliding

window method. The following Table 4 shows the environment settings of the LSTM, BiLSTM, and GRU models. Based on the LSTM model, BiLSTM has the most parameters and the longest computation time among the three models, and GRU has the fewest parameters and the shortest computation time.

TABLE IV
MODEL PROPERTIES

Model	Number of parameters	Operation time(sec)
LSTM	138,596	540.50
BiLSTM	276,996	969.24
GRU	109,196	650.24

Each model is trained over 200 epochs, and the best parameter set is retained for evaluation. In addition, 80% of the dataset is used as training data, the remaining 20% is used as test data, and the final set of parameters is considered the trained model and is used for prediction. The more parameters a model has, the more computation time and resources it requires. Therefore, the number of parameters represents the complexity of a model and is used to evaluate its efficiency.

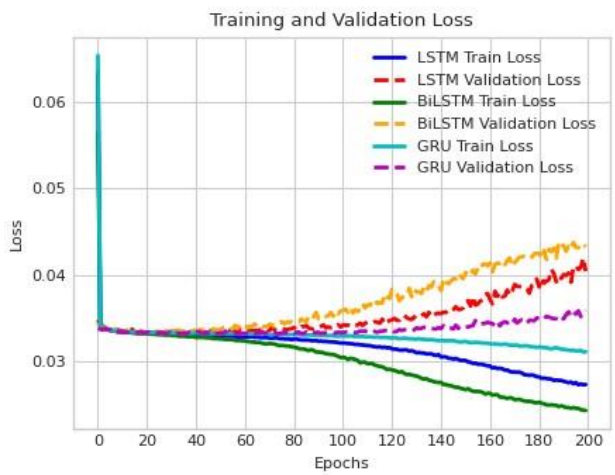

FIGURE 4. LSTM, BiLSTM and GRU loss and validation curves.

The following Fig. 4 shows the loss and validation loss of the LSTM, BiLSTM, and GRU learning models. All three models show that the training loss decreases with increasing epochs. This is the expected behavior during the learning process. The training loss of LSTM and BiLSTM tends to be more stable than the validation loss of GRU, especially in later epochs. However, the BiLSTM model stands out in that its validation loss differs the most from its training loss, with overfitting occurring in later epochs. Although the GRU model appears to have the most stable and consistent behavior in terms of training loss and validation loss, it may not be as stable against overfitting as training continues. Since the GRU model shows a stable trend in validation loss for patterns through training, its learning performance may be the strongest of the three models when compared to learning time.

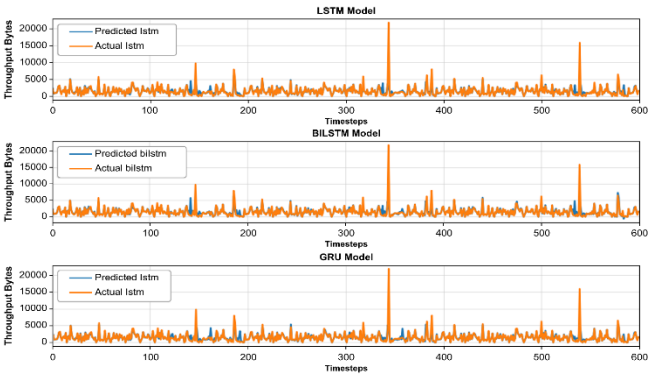

FIGURE 5. Learning model predictions versus ground truth.

In general, Fig. 5 shows that the difference between the predictions and the ground truth was not significant. However, the difference in prediction error was significant when traffic suddenly spiked or crashed, or just before a spike. In the interval between 100 and 150, the differences in predictions made by the LSTM and BiLSTM models were similar, but the GRU model has a more relaxed interval. Between 150 and 200, the LSTM was more accurate in predicting the amount of traffic. Between 200 and 300, there was no significant difference between the three models, and the GRU model made the most accurate predictions before the major turning point between 300 and 350. However, after the major turning point, the LSTM model made a more accurate prediction. Even after that, before the traffic spike between 500 and 550, the GRU model made a more accurate prediction and the BiLSTM model showed a large error. In general, there were errors before and after the traffic peaks. The GRU model made more accurate predictions before the peaks, while the LSTM and BiLSTM models made more accurate predictions after the peaks.

TABLE V
COMPARE RESIDUALS FROM LEARNED MODELS

Section	Actual	Predicted	Residual
LSTM(Avg: 0.5297)			
245	0.918919	0.551719	0.3672
297	0.940383	0.486429	0.453954
318	0.812893	0.106689	0.706204
344	0.896931	0.034377	0.862554
354	0.909941	0.435028	0.474912
378	0.737254	0.370804	0.366449
383	0.687782	0.281635	0.406147
387	0.826255	0.172432	0.653823
390	0.833749	0.247548	0.5862
539	0.550507	0.131363	0.419144
BiLSTM(Avg: 0.5234)			
54	0.927358	0.468326	0.459032
245	0.918919	0.406534	0.512385
297	0.940383	0.405834	0.534549
318	0.812893	0.117376	0.695517
344	0.896931	0.037494	0.859436

378	0.737254	0.303323	0.433931
387	0.826255	0.415905	0.41035
390	0.833749	0.403092	0.430657
491	0.918747	0.450751	0.467996
539	0.550507	0.120123	0.430384
GRU(Avg: 0.5317)			
25	0.933746	0.49272	0.441026
155	0.92502	0.412397	0.512623
163	0.880508	0.420772	0.459736
318	0.812893	0.178517	0.634376
344	0.896931	0.11178	0.785151
378	0.737254	0.18661	0.550644
381	0.632307	0.188646	0.443661
387	0.826255	0.322454	0.503801
390	0.833749	0.261904	0.571845
539	0.550507	0.136676	0.413831

Table 5 compares the residuals between the actual and predicted values for the peak area to help understand Fig. 5. The residuals represent the difference between the actual and predicted values; large differences between these values may indicate that the model predictions are inaccurate.

In the case of the LSTM model, the minimum residual is observed to be approximately 0.3664 and the maximum residual is observed to be approximately 0.8626. This shows a large difference between the actual and predicted value at index 344. For BiLSTM, the average is slightly lower at 0.5234. The highest residual is again observed at exponent 344 with a value of 0.859436, which is very close to the highest residual of the LSTM model. For the GRU case, the average is 0.5317, which is slightly higher than for the LSTM and BiLSTM models. The highest residual is at index 344 with a value of 0.785151. This is slightly lower than the residuals for the same instance of the LSTM and BiLSTM models, indicating that the GRU model may perform slightly better in this case. All three models show similar performance in terms of residuals for the top 10 data points. Although there are some differences in the exact residuals for individual cases, the overall pattern is consistent across the models.

TABLE VI
REGRESSION MODEL PERFORMANCE METRICS

Model	RMSE	MAE	MAPE(%)	R-squared
LSTM	0.1701	0.1252	29.47%	0.6829
BiLSTM	0.1640	0.1169	28.84%	0.7334
GRU	0.1884	0.1220	30.66%	0.7215

Next, we evaluate the RMSE of the three models. Fig. 6 and Table 6 show the error size using RMSE, MAE, MAPE, and R-squared metrics for the LSTM, BILSTM, and GRU models. The average RMSE values of the LSTM, BILSTM, and GRU models according to Table 6 were 0.1701, 0.1640, and 0.1884, respectively. The BISTM model showed the best RMSE value, with a difference of about 0.01-0.02.

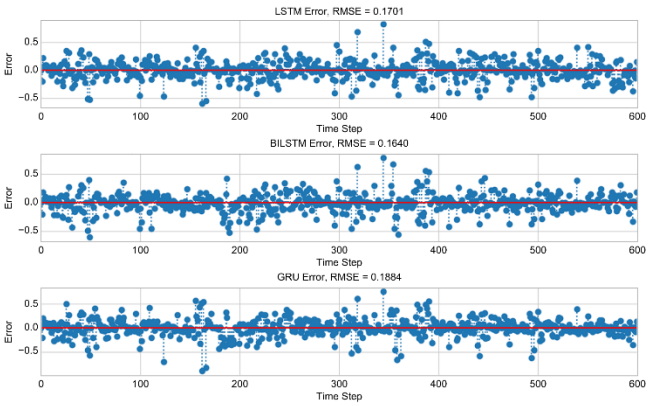

FIGURE 6. Evaluation of RMSE for the machine learning models.

In Fig. 6, the red lines represent each error bar as a measure of its true value. The GRU model has the highest number of error bars, while the BILSTM model has the highest number of errors, as indicated by its RMSE value. The GRU model shows the least amount of error variation during peak traffic periods. This seems to reflect the performance of the learning model for small changes. Overall, the LSTM and BILSTM models outperformed the GRU model with more points close to zero.

In Table 6, the Mean Absolute Error (MAE) is the average of the absolute value of the difference between the predicted value and the actual value. The BiLSTM model had the lowest MAE value of 0.1169, indicating that the BiLSTM model was the best in terms of the size of the average error.

In addition, the Mean Absolute Percentage Error (MAPE) is the absolute value of the prediction error divided by the actual value, averaged, and expressed as a percentage. This metric allows you to express as a percentage how large the prediction error is compared to the actual value. When calculating the MAPE, time steps with an actual value of 0 were excluded to fairly evaluate the predictive power of the model. The results showed that all three models had similar prediction error rates, but the BiLSTM model had the lowest MAPE value of 28.84%.

Finally, the R-squared is an indicator of how well the model explains the variability in the data. The closer the value is to 1, the better the predictive ability of the model. The BiLSTM model had the highest R^2 value of 0.7334. This indicates that the BiLSTM model best explains the variability of the data. Overall, the BiLSTM model showed the most consistent high performance of the three models. However, since the most appropriate model may differ depending on each metric, learning performance and traffic latency must be considered together.

The BiLSTM model appears to be the best performing model in terms of the mean of the top 10 residuals and the RMSE, MAE, MAPE and R-squared. LSTM comes in a close second for both metrics, indicating similar but slightly

inferior performance. The GRU model, while not drastically off, has a slightly higher prediction error, making it the least optimal of the three based on the metrics provided. These metrics provide insight into model performance, but computational efficiency, training time, etc. must be considered when selecting a model to deploy.

V. DISCUSSION

Our methodology emphasizes the importance of simulating real-world scenarios in cyber exercises. While many traditional cyber exercises focus on static patterns, the dynamic nature of today's cyber threats requires a shift toward adaptability and rapid response mechanisms. It helps to effectively respond to unpredictable network operations by deriving new patterns based on traffic concentration and exposing students to different scenarios using deep learning models. With the combination of SDN and deep learning, users and administrators can efficiently improve cyber training scenarios in a data-driven manner based on the network traffic distribution process. However, our method has its limitations. For example, the differences in the residuals of each deep learning model, as shown in Table 5, make it difficult for network administrators to obtain consistent information. To solve this problem, an analysis related to the QoS evaluation items of the distributed packets is required. In addition, hyperparameters such as learning rate, number of neurons, and dropout can be adjusted to maximize the performance of the learning model. Optimizing these hyperparameters is especially important given the highly volatile nature of traffic data. As the nature of cyber threats becomes more complex, compatibility and scalability issues between new traffic and existing systems must also be addressed. Finally, managing dynamic threat vectors that reflect real-world threat scenarios and improving system adaptability and responsiveness in controlled cyber training infrastructures will be key research areas. Finally, managing dynamic threat vectors that reflect real-world threat scenarios and improving system adaptability and responsiveness in a controlled cyber training infrastructure will be key research areas.

VI. CONCLUSION

In this study, we created a graph-based network topology scenario to generate various traffic patterns for cyber training and proposed an SDN-based simulation methodology to reproduce the traffic. The simulation was evaluated by monitoring traffic metrics in the network and measuring the traffic volume. To predict the transmission volume according to the measured traffic volume, we evaluated the network using three models: LSTM, BiLSTM, and GRU.

With more network topology configurations, you can use the Monte Carlo methodology to perform a presumptive evaluation of the behavior of a complex problem or system by randomly selecting a representative sample of all situations. It is particularly useful for analyzing or predicting the characteristics of systems in which stochastic elements are strongly operative. However, we continue to focus on the accurate control and monitoring of network topology and traffic flows using SDN simulations in line with cyber training frameworks. The focus is on recreating specific scenarios from cyber exercises and analyzing the network's response to detect or resolve security issues.

REFERENCES

[1] M. Karjalainen and T. Kokkonen, "Comprehensive cyber arena; The next generation cyber range," *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, Genoa, Italy, 2020, pp. 11-16

[2] B. Ferguson, A. Tall and D. Olsen, " National cyber range overview," *2014 IEEE Military Communications Conference*, Baltimore, MD, USA, 2014, pp. 123-128.

[3] S. K. Sharma and J. Sefchek, "Teaching information systems security courses: A hands-on approach," *Computers & Security*, vol. 26, no. 4, pp. 290–299, 2007.

[4] O O. Romanov, M. Nesterenko, A. Marinov, S. Skolets and H. Burlaka, "SDN network modeling using the GUI MiniEdit," *2022 IEEE 16th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, Lviv-Slavske, Ukraine, 2022, pp. 01-06.

[5] A. Bretas, N. Bretas, J. B. London and B. Carvalho, *Cyber-Physical Power Systems State Estimation*. San Diego, CA, USA:Elsevier, 2021.

[6] E. Seker and H. H. Ozbenli, "The concept of cyber defence exercises (CDX): Planning, execution, evaluation," *2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, Glasgow, UK, 2018, pp. 1-9.

[7] J. Vykopal, M. Vizvary, R. Oslejsek, P. Celeda and D. Tovarnak, "Lessons learned from complex hands-on defence exercises in a cyber range," *2017 IEEE Frontiers in Education Conference (FIE)*, Indianapolis, IN, USA, 2017, pp. 1-8.

[8] M. Karjalainen and T. Kokkonen, "Comprehensive cyber arena; The next generation cyber Range," *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, Genoa, Italy, 2020, pp. 11-16.

[9] O. A. Adeleke, N. Bastin, and D. Gurkan, "Network traffic generation: A survey and methodology," *ACM Computing Surveys*, vol. 55, no. 2. Association for Computing Machinery (ACM), pp. 1–23.

[10] L. Du, J. He, T. Li, Y. Wang, X. Lan, and Y. Huang, "DBWE-Corbat: Background network traffic generation using dynamic word embedding and contrastive learning for cyber range," *Computers & Security*, vol. 129. Elsevier BV, p. 103202, Jun-2023.

[11] M. Park, H. Lee, Y. Kim, K. Kim, and D. Shin, "Design and implementation of multi-cyber range for cyber training and testing," *Applied Sciences*, vol. 12, no. 24, p. 12546.

[12] A. Protogerou, S. Papadopoulos, A. Drosou, D. Tzovaras, and I. Refanidis, "A graph neural network method for distributed anomaly detection in IoT," *Evolving Systems*, vol. 12, no. 1, pp. 19-36, Jun. 2020.

[13] B. Chen, D. Zhu, Y. Wang, and P. Zhang, "An approach to combine the power of deep reinforcement learning with a graph neural network for routing optimization," *Electronics*, vol. 11, no. 3, p. 368, Jan. 2022

[14] F. Rebecchi et al., "A digital twin for the 5G era: the SPIDER cyber range," *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Belfast, United Kingdom, 2022, pp. 567-572.

[15] L. Barsellotti, L. De Marinis, F. Cugini and F. Paolucci, "FTG-Net: Hierarchical flow-to-traffic graph neural network for DDoS attack detection," *2023 IEEE 24th International Conference on High Performance Switching and Routing (HPSR)*, Albuquerque, NM, USA, 2023, pp. 173-178.

[16] H. Abdelgader Eissa, K. A. Bozed and H. Younis, "Software defined networking," 2019 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), Sousse, Tunisia, 2019, pp. 620-625.

[17] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," Modeling and Tools for Network Simulation. Springer Berlin Heidelberg, 2010, pp. 15–34.
[18] R. L. S. de Oliveira, C. M. Schweitzer, A. A. Shinoda, and Ligia Rodrigues Prete, "Using Mininet for emulation and prototyping software-defined networks," 2014 IEEE Colombian Conference on Communications and Computing (COLCOM). IEEE, Jun-2014.
[19] OMNeT++, [Online]. Available: https://omnetpp.org/intro/. Accessed on: Sep 12, 2023.
[20] D. Conte de Leon, C. E. Goes, M. A. Haney, and A. W. Krings, "ADLES: Specifying, deploying, and sharing hands-on cyber-exercises," *Computers & Security*, vol. 74, pp. 12–40, 2018.
[21] I. Ullah, B. Raza, S. Ali, I. A. Abbasi, S. Baseer, and A. Irshad, "Software defined network enabled fog-to-things hybrid deep learning driven cyber threat detection system," *Secur. Commun. Networks*, vol. 2021, 2021
[22] A. Azzouni and G. Pujolle, "NeuTM: A neural network-based framework for traffic matrix prediction in SDN," *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, Taipei, Taiwan, 2018, pp. 1-5
[23] J. B. Torres, J. E. Regencia, and W. E. S. Yu, "Real network traffic data with PCAP in a software-defined networking test framework for quality of service mechanisms," *IT Convergence and Security. Lecture Notes in Electrical Engineering*, Singapore, 2021, pp. 153-161.
[24] I. Sharafaldin, A. H. Lashkari and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization", *Proc. ICISSP*, pp. 108-116, 2018.
[25] A. H. Lashkari, G. D. Gil, M. S. I. Mamun and A. A. Ghorbani, "Characterization of tor traffic using time based features", Proc. 3rd Int. Conf. Inf. Syst. Security Privacy, pp. 253-262, 2017, [online] Available:http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006105602530262.
[26] S. Kadry, A. Abdallah and C. Joumaa, "On the optimization of dijkstras algorithm" *in Informatics in Control Automation and Robotics*, Springer, pp. 393-397, 2011.
[27] S. Kaur, J. Singh and N. S. Ghumman, "Network programmability using pox controller", Proc. Int. Conf. Commun. Comput. Syst. (ICCCS), vol. 138, pp. 134-138, 2014.
[28] Tcpreplay. AppNeta. [Online]. Available: https://tcpreplay.appneta.com/. Accessed on: July 31, 2023.
[29] Bit-Twist. Addy Yeow. [Online]. Available: http://bittwist.sourceforge.net/. Accessed on July 31, 2023.
[30] Y. Tian, W. Chen and C. -T. Lea, "An SDN-Based traffic matrix estimation framework," *in IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1435-1445, Dec. 2018.
[31] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *in IEEE Access*, vol. 5, pp. 18042-18050, 2017.
[32] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network", *Physica D Nonlinear Phenom.*, vol. 404, Mar. 2020.
[33] Z. Lv, J. Guo, A. K. Singh and H. Lv, "Digital twins Based VR simulation for accident prevention of intelligent vehicle," *in IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 3414-3428, April 2022.
[34] K. Cho et al., Learning phrase representations using RNN encoder-decoder for statistical machine translation, 2014, [online] Available: https://www.arXiv:1406.1078.

DONG-WOOK KIM received the bachelor's degree in computer software and the master's degree in computer engineering from Gachon University, South Korea, in 2015 and 2017, respectively, where he is currently pursuing the Ph.D. degree with the Department of Computer Engineering. His research interests include insider threats, information security, data mining, and machine learning.

GUN-YOON SHIN received the M.S. and Ph.D. degrees in computer engineering from Gachon University, Republic of Korea, in 2018 and 2023, respectively. Currently, he is a Postdoctoral Researcher at Gachon University, Korea. His research interests include authorship attribution, unknown attack detection, network anomaly detection, information security, machine learning, and artificial intelligence.

YOUNG-HOAN JANG earned his Master's degree in Computer Science from Gachon University in 2017, and obtained his Ph.D. in Computer Science from the same university in 2021. His research interests include network traffic, anomaly detection, IoT, and EMS.

SEUNGJAE CHO received the Master's degree in Modeling and Simulation Engineering from Hannam University, Republic of Korea, in 2011. Since May 2001, he is currently affiliated with LIG Nex1, a leading defense industry in Korea, as a researcher for cyber security. His research interests include network security and system engineering.

KWANGSOO KIM received a B.S. degree in Information and Computer Engineering from Ajou University, Republic of Korea, in 2009, and a Ph.D. degree in Computer Engineering from Ajou University, Republic of Korea in 2017. Since January 2017, he is currently affiliated with LIG Nex1, a leading defense industry in Korea, as a researcher for cyber security. His research interests include network security and cyber warfare, especially cyber training system.

JAESIK KANG received the bachelor's degree in computer engineering and the master's degree in computer engineering from ChungNam National University, Korea, in 2015 and 2020. Since July 2022, he is currently affiliated with LIG Nex1, a leading defense industry in Korea, as a researcher for cyber security. His research interests include AI, Cyber Security, and Software Engineering.

MYUNG-MOOK HAN received the M.S. degree in computer science from the New York Institute of Technology, in 1987, and the Ph.D. degree in information engineering from Osaka City University, in 1997. From 2004 to 2005, he was a Visiting Professor at the Georgia Tech Information Security Center (GTISC), Georgia Institute of Technology. He is currently a Professor with the Department of Software, Gachon University, South Korea. His research interests include information security, intelligent systems, data mining, and big data.