Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.054108

ARTICLE

Network Traffic Synthesis and Simulation Framework for Cybersecurity Exercise Systems

Dong-Wook Kim[1], Gun-Yoon Sin[2], Kwangsoo Kim[3], Jaesik Kang[3], Sun-Young Im[3] and Myung-Mook Han[1,*]

[1]Department of AI Software, Gachon University, Seongnam-Si, 13120, Republic of Korea

[2]School of Computer Engineering & Applied Mathematics, Hankyong National University, Pyeongtaek-Si, 17738, Republic of Korea

[3]Cyber Electronic Warfare Research and Development, LIG Nex1, Seongnam-Si, 13488, Republic of Korea

*Corresponding Author: Myung-Mook Han. Email: mmhan@gachon.ac.kr

Received: 19 May 2024 Accepted: 26 July 2024 Published: 12 September 2024

ABSTRACT

In the rapidly evolving field of cybersecurity, the challenge of providing realistic exercise scenarios that accurately mimic real-world threats has become increasingly critical. Traditional methods often fall short in capturing the dynamic and complex nature of modern cyber threats. To address this gap, we propose a comprehensive framework designed to create authentic network environments tailored for cybersecurity exercise systems. Our framework leverages advanced simulation techniques to generate scenarios that mirror actual network conditions faced by professionals in the field. The cornerstone of our approach is the use of a conditional tabular generative adversarial network (CTGAN), a sophisticated tool that synthesizes realistic synthetic network traffic by learning from real data patterns. This technology allows us to handle technical components and sensitive information with high fidelity, ensuring that the synthetic data maintains statistical characteristics similar to those observed in real network environments. By meticulously analyzing the data collected from various network layers and translating these into structured tabular formats, our framework can generate network traffic that closely resembles that found in actual scenarios. An integral part of our process involves deploying this synthetic data within a simulated network environment, structured on software-defined networking (SDN) principles, to test and refine the traffic patterns. This simulation not only facilitates a direct comparison between the synthetic and real traffic but also enables us to identify discrepancies and refine the accuracy of our simulations. Our initial findings indicate an error rate of approximately 29.28% between the synthetic and real traffic data, highlighting areas for further improvement and adjustment. By providing a diverse array of network scenarios through our framework, we aim to enhance the exercise systems used by cybersecurity professionals. This not only improves their ability to respond to actual cyber threats but also ensures that the exercise is cost-effective and efficient.

KEYWORDS

Cybersecurity exercise; synthetic network traffic; generative adversarial network; traffic generation; software-defined networking

1 Introduction

In cybersecurity, providing network professionals with realistic threat scenarios and appropriate training environments that mimic real-life situations has recently emerged as a major challenge. To address this challenge, competitive mock training systems have been developed, and cybersecurity experts have established training regimes tailored to various tasks according to their roles in attack and defense. Industries, universities, research centers, and national security agencies worldwide are developing and advancing environments for cybersecurity training. Among these technological platforms, an early instance of supporting research and development, training, and exercises was developed by the Defense Advanced Research Projects Agency (DARPA), which established the National Cyber Range (NCR) [1]. The NCR provides technologies related to exercise simulations that allow cybersecurity personnel to learn proper responses to cyberattacks. However, the usage of a cyber range is limited by specific requirements, and it has command and control systems functionalities tailored to specific exercise scenarios.

Cybersecurity exercise systems must support highly realistic simulations by providing trainees with a realistic network environment that mimics actual scenarios. To establish such an environment, the characteristics, operations, and usage patterns of real networks should be replicated to resemble actual scenarios, thus requiring communication settings for each host and switch as well as techniques for generating and monitoring related network traffic. This process demands significant resources and incurs high costs in computing hardware and management software, resulting in substantial financial burdens for the construction and maintenance of systems like cyber ranges [2,3]. Additionally, postprocessing is necessary to exclude personal information, often limiting the use of the testbed environment for administrative efficiency [4]. In this context, while technologies such as schedulers and behavior modeling using Markov models [5,6] can be used for automatic and realistic network traffic generation, the use of generative adversarial networks (GANs) for synthesizing data that mimic network traffic flows using deep learning is gaining interest. Research on network traffic generation using GANs is becoming increasingly active across various network levels, from byte-level packet composition to scheduling patterns. The corresponding techniques allow to replicate of complex network patterns, including normal and malicious traffic, thus contributing to research on data imbalance and intrusion detection systems. Generative models may substantially enhance the precision and diversity of cybersecurity exercise simulations and facilitate sophisticated modeling of diverse scenarios that may occur in real network environments.

To deploy generated network traffic in a cybersecurity exercise environment, real-time simulation requires a model that reflects real network conditions [7]. Hence, network emulators based on software-defined networking (SDN) are being increasingly used to integrate management, response, and monitoring techniques for cyberattacks. SDN virtualization allows to implement and execute various cyber threat scenarios in complex network environments, such as smart grids, at a reduced cost, thus facilitating system vulnerability analysis, identification of communication environment impacts from cyberattacks, and enhanced response capabilities [8]. Moreover, machine learning and artificial intelligence can be integrated into an SDN environment, facilitating their use in anomaly detection of network traffic, real-time network status estimation, and intrusion detection. In cybersecurity exercises, generating realistic network traffic based on specific scenarios is essential, and this process involves complex procedures that depend on the network topology.

We propose a framework to synthesize network traffic for cybersecurity exercises. The framework establishes strategies to evaluate network adaptability by creating and deploying realistic scenarios, thus contributing to the generation of diverse cybersecurity exercise scenarios. Additionally, this framework includes the integration of AI (Artificial Intelligence) modules for seamless integration with existing cybersecurity exercise systems and incorporates synthetic data generation methods utilizing existing similar traffic generation technologies. A key component in the proposed framework is the conditional tabular GAN (CTGAN) [9], which generates synthetic tabular data with similar statistical characteristics to real data. Statistical characteristics in a network environment can include features such as packet size, transmission time, protocol type, and source and destination IP (Internet Protocol) addresses. By learning the distribution of real data, it is possible to generate similar synthetic data. This allows the construction of a network environment based on SDN for simulating the flow and patterns of network traffic in real situations and comparing them to real traffic. For comparison, a network traffic matrix can be constructed and analyzed to measure the traffic volume originating from hosts, thus providing important insights into adjustments and improvements to obtain realistic cybersecurity exercise scenarios. Therefore, the proposed approach has been validated for its effectiveness in optimizing network performance and generating cybersecurity exercise scenarios, verifying the effectiveness of the traffic circulated in the field of cybersecurity research and training systems.

2 Related Work

2.1 Cybersecurity Exercise and Scenarios

Cybersecurity training consists of interactive simulations targeting local networks, systems, tools, and applications of organizations equipped with traditional cyber-range environments and internet-level connectivity [10]. This research area focuses on building and assessing capabilities aligned with organizational goals, promoting cybersecurity awareness, and facilitating information sharing. Specifically, cybersecurity training involves defense training in customized arenas with various active scenario engines implemented as simulations to assess the performance of trainees. It requires the integration of computing platforms, infrastructure, and software provisioning technologies [11].

The configuration for cybersecurity training involves the simulation of threat actions and attack scenarios as well as the division of training participants into attack, defense, and monitoring teams to respond to cyberattacks. Additionally, the scalability of the training system must ensure support for numerous simultaneous connections and realistic traffic loads, integrate diverse and complex attack vectors to accurately mirror real threats, and provide robust monitoring and analysis tools to evaluate the effectiveness of defense strategies in real environments [12].

Furthermore, the framework should facilitate seamless integration with existing cybersecurity tools and platforms, offer flexibility for scenario customization, and be able to scale both horizontally and vertically to accommodate future growth and technological advancements. This approach is adjusted according to the specific requirements of every scenario, and continuous research, development, verification, testing, and evaluation constitute a distinctive research challenge [13].

Russo et al. [14] presented a method for designing and validating cybersecurity exercise scenarios for next-generation cyber ranges by developing models for the design, validation, automatic generation, and testing of such scenarios. They introduced the scenario description language for scenario modeling, thereby contributing to the precise definition and reproducibility of cybersecurity scenarios and enhancing the efficiency and effectiveness of cybersecurity exercises within cyber ranges.

Yamin et al. [15] proposed a platform for simulating cybersecurity exercise scenarios to obtain environments using a domain-specific language and infrastructure orchestration. This platform served as a dynamic cybersecurity exercise scenario generator. The scenarios were easily designed through the interconnectivity of computers, routers, and switches while abstracting rules and vulnerabilities between attack and defense to present realistic environments.

Wen et al. [16] used an ontology for managing and standardizing knowledge to facilitate its sharing and management for scenario search and reuse in cybersecurity exercises. By identifying elements and relationships within a cybersecurity scenario, knowledge was divided into three sub-models: scenario information, scenario operation, and security knowledge models. This division helped structure a scenario instance library, thus obtaining a systematic method for managing scenarios.

The abovementioned studies emphasize the need for flexible creation and realistic reproducibility testing mechanisms in the development of cybersecurity exercise scenarios, indicating that a systematic approach is essential for maximizing the efficiency of cybersecurity exercises and corresponding scenarios.

2.2 Techniques for Network Traffic Generation Using GAN

Once cybersecurity exercise scenarios are prepared, deploying traffic that includes background traffic generation and user simulation to mimic a realistic internet environment in the exercise communication network is essential. Traffic generation techniques are being extensively studied based on technologies such as SDN and network function virtualization. Generally, as collected network traffic in real environments is scarcely available for researchers owing to privacy policies, traffic generators are required [4].

Adeleke et al. [17] classified network traffic generators into constant/maximum throughput generators, application-level synthetic workload generators, trace file replay systems, model-based generators, and script-driven traffic generators depending on the technique adopted for pushing packets into the network. On the other hand, the advent of GANs in 2014 brought innovation to generative machine learning. Thereafter, generative models, which estimate the probability distribution of data to then extract samples, have been adopted for network traffic generation [18].

Cheng [19] proposed Packet Generation Adversarial Networks (PAC-GAN), which combines a convolutional neural network (CNN) with a GAN to generate network packets. PAC-GAN implements an inverse CNN for the generator, while the discriminator uses a conventional CNN for supervised classification. This method was used to evaluate the feasibility of using GANs to generate realistic traffic flows, such as ICMP (Internet Control Message Protocol) pings, DNS (Domain Name System) queries, and HTTP (Hypertext Transfer Protocol) web requests.

Manocchio et al. [20] proposed Flow Generative Adversarial Networks (FlowGAN) for synthesizing network flow data for training a network intrusion detection system. They combined a supervised component that guided the training data distribution to learn all modes based on a manifold-guided GAN, producing synthetic traffic closely resembling real traffic.

Dowoo et al. [21] proposed Packet Capture Generative Adversarial Networks (PcapGAN) to augment data for cybersecurity fields that face data scarcity. PcapGAN consists of an encoder, generator, and decoder, and it uses a style-based encoder to extract information in various formats. The formats include graph data for source and destination IP (Internet Protocol) addresses, image data for time intervals, and hierarchical sequences for sequences, and they are used as inputs for a hybrid generator. Finally, the decoder combines the generated information and reconstructs valid pcap (packet capture) files, obtaining data similar to the original.

Ring et al. [22] presented three approaches using a Wasserstein GAN with gradient penalty to preprocess network data and replay it based on new flows. To address the GAN limitation of only being applicable to continuous data properties, they introduced a method using the IP2Vec method for learning vector representations of network flow data that include continuous, numerical, categorical, and binary attributes. Hence, they demonstrated an expanded application scope for GANs in processing network data.

Research on packet-level data generation using GANs is being conducted to demonstrate its efficacy and practicality. However, challenges arise in simulating the continuity and temporal flow of network packets due to data imbalance issues. Additionally, there are limitations in creating customized scenarios based on specific conditions and in finely adjusting individual packet levels when generating traffic based on network flows. To address these issues, we aim to reconstruct structured packet characteristics to improve data imbalance, create customized scenarios, achieve fine adjustments at the packet level, and enhance the evaluation of generated traffic.

3 Proposed Framework

We propose an exercise network topology and traffic analysis framework for generating network traffic distributions applicable to cybersecurity exercise scenarios. The framework consists of four main stages, namely, data collection, network topology generation, network traffic distribution, and network performance evaluation, as detailed in this section.

3.1 General Concept

A diagram of the proposed framework is shown in Fig. 1, which indicates the procedure for generating network traffic. Initially, network packets in pcap files are decoded. These network packets are typically stored in binary form and can be converted into hexadecimal representation for analysis. From the converted data, various properties of network layers such as Ethernet headers, IP headers, and TCP (Transmission Control Protocol) headers are extracted. The extracted information includes destination and source MAC (medium access control) addresses, IP versions, source and destination IP addresses, and port numbers. These attributes are transformed into features for input into the proposed CTGAN and presented in tabular form. These data are then used by the CTGAN generator to create synthetic data based on condition z. The generated data are assessed by the discriminator to distinguish between real and synthetic data and evaluate their similarity to real data. Once synthetic data are created, they are reencoded in the pcap format based on TCP streams. The encoded data are deployed through SDN to compare their similarity to real network traffic data.

To compare and evaluate real packets with synthetic packets, we managed and transmitted traffic across the network using an SDN (Software-Defined Networking) controller. The SDN controller communicated with network switches to regulate traffic, enabling real-time monitoring and evaluation of network traffic volumes. Based on this evaluation, we determined whether the synthetic data accurately reflected the operation of the real network and assessed the accuracy and effectiveness of training scenarios. Through this process, our framework provided a comprehensive methodology for generating and deploying network training traffic, allowing for the creation of realistic cybersecurity training environments. By utilizing CTGAN, we were able to reconstruct valid packets expressed as structured data, effectively addressing the imbalance issue and balancing normal and malicious traffic. This enabled the flexible creation of various scenarios by combining specific traffic segments, thereby enhancing the ability of trainees to respond to diverse patterns of simulated traffic, contributing to improved response capabilities.

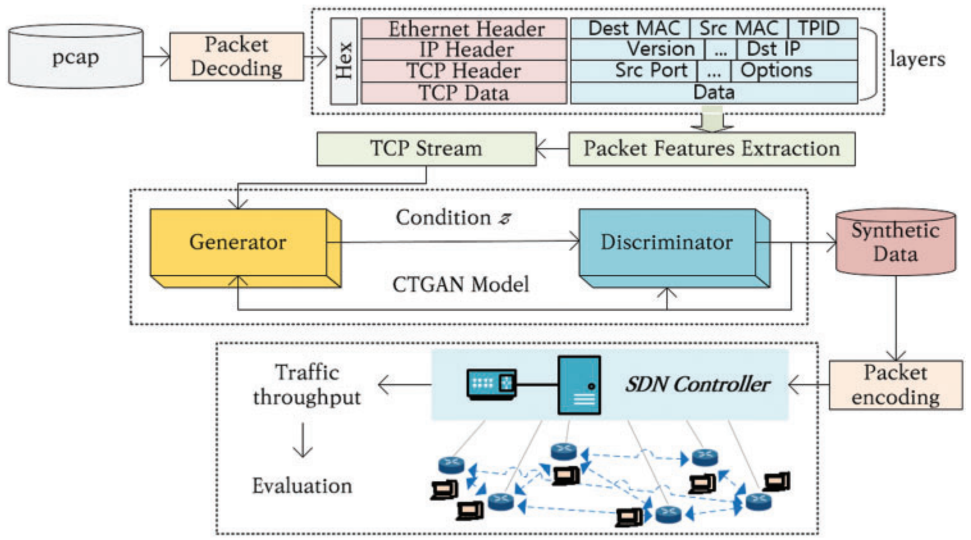

Figure 1: Framework for generating network traffic for cybersecurity exercise (Dst, destination; Src, source; TPID (Tag Protocol Identifier))

3.2 Network Packet Decoding

Network packet decoding involves analyzing binary data packets transmitted over a network to extract their information. Packets are primarily composed of headers and payloads, with the header containing control information such as source and destination IP addresses, port numbers, data length, and protocol type.

For packet decoding, captured packet data from pcap files can be extracted using tools like Wireshark and TCPDUMP, which provide hexadecimal information representing the structure and content of data packets in network communications [23]. The hexadecimal format represents binary data in base 16, conveniently displaying each byte as two hexadecimal digits. Packets can be analyzed in various ways according to the OSI (Open Systems Interconnection) model of network layers. Hexadecimal information includes data such as the source and destination IP addresses, version, header length, time to live, and protocol type, which help determine the packet path, priority, and fragmentation status. Additionally, at the TCP/UDP (User Datagram Protocol) layer, information such as port numbers and application layer protocols like HTTP and HTTP Secure can be verified, allowing the identification of the service or application for which the packet is intended and the type of transmitted data.

As shown in Fig. 2, from captured packets, we extract and classify the flow of consecutive data packets according to TCPStream, identifying the path, sender and receiver, and amount and type of transmitted data. Hence, data are structured to analyze patterns and characteristics of specific communication sessions. Such conversion is applied to produce data in an easily analyzable form for input into the CTGAN and assign unique identifiers to each packet based on data structuring, which aids in tracking relations between packets during data analysis and identifying packets of interest.

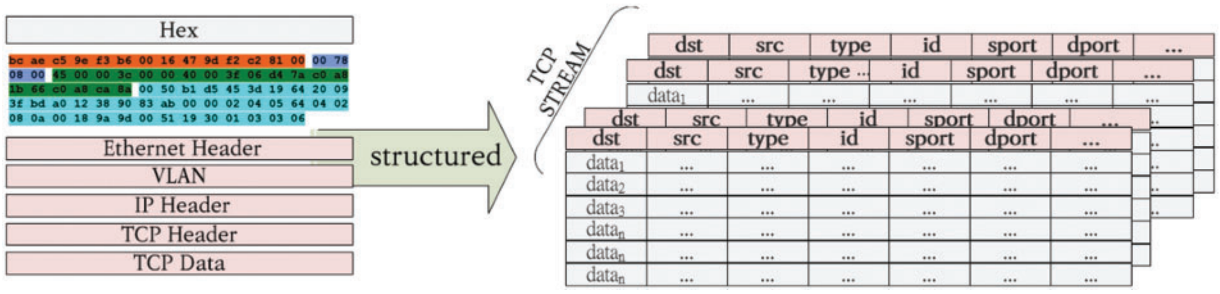

Figure 2: Data structure by network layer (dst, destination; dport, destination port; sport, source port; src, source; VLAN (virtual local area network))

3.3 Traffic Synthesis Using CTGAN

CTGAN generates synthetic tabular data by modeling the statistical characteristics and structure of real datasets [9]. CTGAN can handle various data types, such as continuous, discrete, and categorical variables, by creating new instances while preserving the statistical characteristics of the original real data. In addition, it can include information not present in the original data. Due to these capabilities, it can generate similar synthetic data while preserving the distribution of the real data. The generated synthetic data can be evaluated for statistical similarity and scenario suitability through traffic patterns, packet loss rates, and latency.

CTGAN generates synthetic data through adversarial learning between the generator and discriminator by learning conditional probability distributions. This approach allows a GAN to replicate real data distributions and generate data based on given conditions by using labels or other metadata corresponding to each data point. In CTGAN, discrete values are represented as one-hot vectors and continuous values with arbitrary distributions are normalized in [−1, 1] by applying min-max normalization to represent information about data structures. CTGAN designs and applies specific normalization operations over columns to handle various types of data, especially continuous and discrete categorical data with complex distributions. Moreover, CTGAN ensures that all categories in discrete attributes are evenly sampled during training through conditional generation and training sampling to correctly reflect the real data distribution during testing.

To replicate a real data distribution, the generator learns the conditional distribution of rows that match specific values of various columns for given conditions $D_{i*} = k^*$, expressed as $\widetilde{r} \sim P_G(\text{row} \mid D_{i*} = k^*)$. Thus, the conditional GAN can effectively replicate the real data distribution as follows:

$$P(\text{row}) = \sum\nolimits_{k \in D_{i*}} P_G(\text{row} \mid D_{i*} = k^*)\, P(D_{i*} = k) \quad (1)$$

Eq. (1) describes the process of calculating the overall distribution of the generated rows P(row) across possible conditions k. Each term $P_G(\text{row} \mid D_{i*} = k^*)$ represents the conditional distribution of rows generated by generator G given condition k^*, and $P(D_{i*} = k)$ represents the probability in the real data for condition k^*. Hence, the generator learns the conditional distribution and reproduces the overall distribution of real data considering the probability of each condition. The conditional generation of CTGAN requires a clear definition of conditions to ensure that the synthesized data accurately reflects the given conditions and precisely learns the conditional distributions of real data. To support this process, conditional vectors, a generator loss, and sampling-based training are used.

These components allow CTGAN to properly model complex distributions in continuous and discrete categorical data using conditional information to generate diverse and accurate synthetic data.

CTGAN can handle structured network traffic data and reconstruct the correct sequence of TCP traffic flows as shown in Fig. 3. First, multiple traffic records are collected and arranged according to specific TCP streams. "Real packet order" reflects the order in which data are transmitted in the actual network. These data are provided as input to CTGAN, which learns their sequence considering the actual packet order, which is crucial for accurately modeling network protocol operations like TCP handshake. Next, "seq reassemble" ensures that the sequence reconstructed by the model correctly reflects specific stages of TCP communication, such as handshaking. As a result, the model can generate new traffic based on learned patterns, maintaining structural characteristics and sequences of TCP streams that reflect actual network operations.

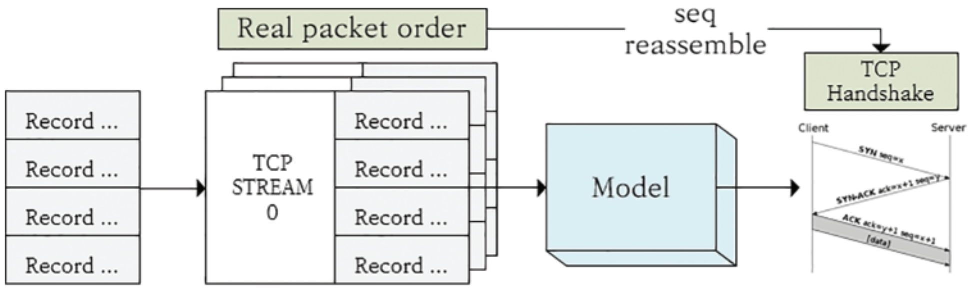

Figure 3: TCP traffic sequence reconstruction and learning using CTGAN (ACK, acknowledge; seq, sequence; SYN (synchronize))

3.4 Synthetic Data Encoding

We encode the tabular data generated using CTGAN into packets according to specific attributes. Encoding is necessary because original pcap files are needed to deploy packets on the actual exercise network, and the patterns and transmission volumes should be compared between real and generated traffic to assess the plausibility of synthetic traffic.

Fig. 4 illustrates the encoding of network packet layers for data synthesized using CTGAN. Typically, network packets are composed of multiple layers, each responsible for specific information according to its header positions. First, the Ethernet layer includes information such as MAC addresses and frame types, enabling packets to reach their destinations through the physical network. Second, the IP layer contains data such as source and destination IP addresses and packet length, which are necessary for correctly routing packets across the internet. Particularly, along with IP addresses, the field "len" must include the total length of the IP packet and the precise payload or length of the data field. Third, the TCP layer is responsible for reliable data transmission and includes port numbers, sequence numbers, and other information. Finally, the data segment contains the actual application data being transmitted.

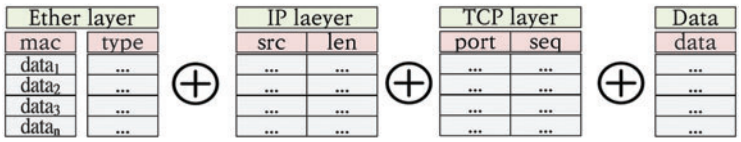

Figure 4: Network packet layer encoding (len, length; seq, sequence; src, source)

For configuration, we use the Scapy library [24] to automatically calculate specific fields such as the IP packet length, aiming to reduce complexity and prevent errors in assembling network packets to ensure that synthetic data accurately reflects real network operations. When data generated by CTGAN are encoded using Scapy, the fields of every layer can be adjusted according to precise network communication standards, ensuring operation in environments built to resemble real network traffic or in simulations for network security and performance evaluation.

3.5 Network Emulator and Traffic Replay

For traffic data deployment, we implement a cybersecurity exercise traffic deployment simulation by combining the Mininet emulator and Tcpreplay suite to configure the network topology and replay pcap files. Mininet allows to create and manage virtual routers, switches, and hosts in a single system using SDN and enables testing network protocols and applications without requiring hardware [25]. During simulation, the network behavior is monitored using the POX controller [26], a centralized function that manages and controls network flows.

To generate an arbitrary network topology and traffic from hosts, we use the Tcpreplay suite [27] with the Mininet emulator. This suite uses actual network traffic data contained in pcap files to distribute traffic and realistically reproduce network traffic situations for assessing the system performance. The simulated environment allows the configuration of the network topology environments by replaying pcap data in the network topology created through Mininet.

3.6 Traffic Monitoring

Through traffic monitoring, we store captured network data in the form of a traffic matrix. The traffic matrix comprehensively captures the overall traffic flow information from the network [28]. To deploy packets and observe traffic volumes in the configured network topology, we obtain the network traffic matrix through the POX controller of the Mininet emulator. Typically, the traffic matrix represents the amount of traffic exchanged between every pair of nodes within the network over a specific time interval by transmitting packets contained in a common pcap file. This allows for the analysis of network performance, or the prediction of traffic patterns based on the amount of traffic generated during the transmission period.

To calculate the change in network traffic, we determine the time difference between two measurements and base our calculations on the difference between the current and previous traffic states. This difference represents the "change in traffic," and dividing it by the elapsed time gives the "rate of traffic change per hour." Hence, the change in traffic is calculated by subtracting the previous traffic state from the latest traffic state and dividing by the duration as follows:

$$\text{Traffic change rate } \Delta\text{T} = \frac{\text{latest traffic} - \text{previous traffic}}{\text{duration}} \tag{2}$$

where the latest traffic and previous traffic are matrices that represent the traffic states, with each matrix element indicating the traffic volume at a specific time. Therefore, by calculating the difference between these two matrices, the change in traffic per element can be determined. The calculated traffic matrix can then be viewed as the hourly traffic change between every pair of nodes given by the traffic matrix $\text{T} = \Delta\text{T}$. This method allows for real-time understanding and responding to changes in network conditions to determine and adjust the dynamic network performance.

4 Evaluation Results

4.1 Dataset

For generating network traffic for cybersecurity exercises, we used the MACCDC (Mid-Atlantic Collegiate Cybersecurity Defense Competition) 2012 dataset constructed at MIT (Massachusetts Institute of Technology) Lincoln Laboratory [29]. This dataset includes various stages of a cybersecurity attack, from scanning and reconnaissance to exploitation. The dataset consists of 17 individual files, mostly capturing packets reflecting the information gathering and reconnaissance phases. Instead of generating from the entire dataset, we generated synthetic traffic from the last file (maccdc2012_00016.pcap), which contains 3,816,907 traffic records. From this file, we selected the five most active hosts in the network to synthesize data.

4.2 Experimental Environment

We configured the experimental environment in a computer running Python to generate random network topologies, operate learning models, and simulate network traffic in a WSL2 (Windows Subsystem for Linux 2) environment with Ubuntu 20.04. In this environment (see Table 1), we installed the Mininet emulator, configured the network to replay pcap files from the MACCDC 2012 dataset, and calculated and monitored the traffic matrix.

Table 1: Specifications of the experimental environment

Component	Specification
Processor	Intel Xeon E5-2620 v4
Memory	48 GB
Operating system	Windows 10, WSL2 (Ubuntu 20.04)
Language	Python 3.8
Libraries	Networkx, tensorflow, sklearn

4.3 Network Environment and Topology

The network topology was constructed by selecting five random hosts from file maccdc2012_00016 .pcap in the MACCDC 2012 dataset. Using a subset of traffic flow scenarios for synthetic data evaluation is likely more intuitive and suitable for developing cybersecurity exercise scenarios, where the synthetic data can be added to the original traffic flows, than using all the available hosts. Additionally, we extracted representative traffic types corresponding to TCP, SSH (secure shell), and HTTP, which are commonly used protocols at the IPv4 layer, based on the TCP handshake (ACK (Acknowledge character), SYN) sequence. The system was configured to analyze and simulate traffic patterns in various network environments according to scenarios involving communication in the training system. However, we avoided using payloads with encrypted packets as much as possible. This was to minimize the string errors that could occur during the decoding process and to ensure proper encoding. Fig. 5 shows the connections and communication directions among the network IP addresses. Overall, five hosts were designated for sending communications, while some hosts were exclusively set as receivers. Overall, 18 hosts interacted within the network. For each of the five sending hosts, Table 2 lists the IP address, the total number of packets sent or received, and the total size of the packets in bytes.

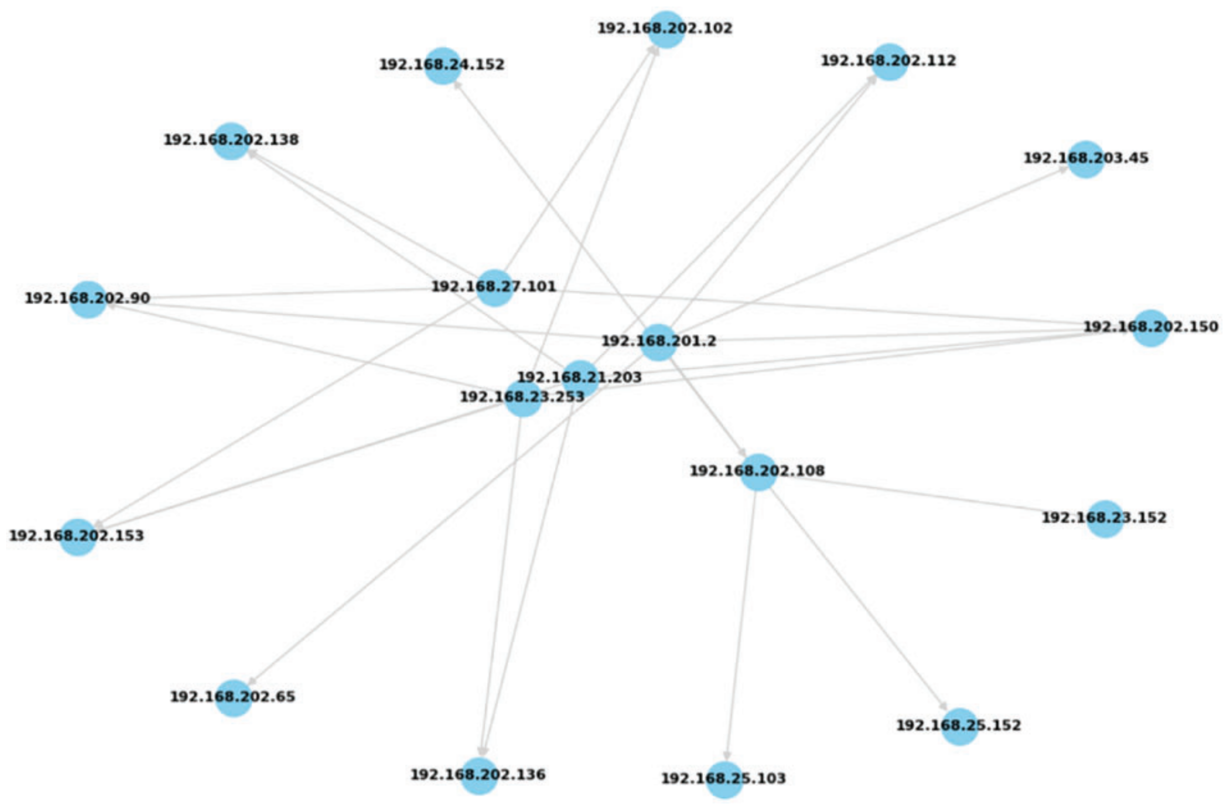

Figure 5: Communication network topology used for evaluation

Table 2: Number of packets and their size during network communication

Host	No. packets	Size (bytes)
192.168.21.203	3598	1,639,598
192.168.202.2	7025	3,176,795
192.168.23.253	1471	134,094
192.168.27.101	3872	293,001
192.168.202.108	254	38,551

The host with IP address 192.168.21.203 sent/received 3598 packets or approximately 1.64 MB of data, while host 192.168.202.2 processed most data, approximately 3.18 MB from 7025 packets. Host 192.168.23.253 handled 1471 packets or 134,094 bytes of data, showing a relatively low data usage. Host 192.168.27.101 transmitted approximately 293 KB of data from 3872 packets, amounting to 293,001 bytes. Finally, host 192.168.202.108 processed the least amount, with 254 packets or 38,551 bytes of data.

The host environment in the network virtual simulation using the Mininet topology configuration environment was set as shown in Fig. 6.

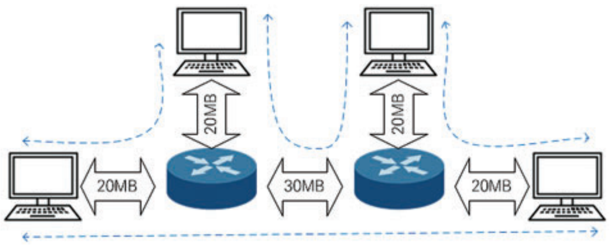

Figure 6: Mininet network topology for evaluation scenario

Mininet can be configured to handle large-scale traffic for hosts and switches in network simulations. Each host and router had a maximum transmission bandwidth of 20 MB, and the bandwidth between routers was set to 30 MB. These bandwidth settings provided an environment with sufficient bandwidth while effectively transmitting information among various network components. This setup contributed to simulating various traffic situations that could occur in a real network and played a crucial role in constructing diverse scenarios for cybersecurity exercises. Through these settings, the framework effectively managed the scalability and computational resources required for large-scale network simulations. By integrating these three tools, we replicated and analyzed a network environment with realistic traffic distribution for cybersecurity exercises and designed and implemented an effective simulation system.

4.4 Experimental Results and Evaluation

Based on the configured dataset and environment, we applied CTGAN to synthesize data based on the original pcap file. The input data for CTGAN comprised 22 features that constitute the network layer. Each feature was composed of categorical and discrete values. The main parameters related to the configuration of the generator and discriminator of the learning model were set as follows. The generator had three layers of dimensions 256, 256, and 1 and a learning rate of 0.0001. The discriminator had three layers of dimensions 128, 128, and 1, having a smaller capacity than the generator and enabling effective differentiation between real and synthetic data. The smaller discriminator enhanced the computational efficiency and prevented overfitting by reducing excessive training. The discriminator learning rate was set slightly higher than that of the generator to 0.0002, allowing the discriminator to learn faster than the generator and efficiently identify differences between real and synthetic data. For these settings, training proceeded with a batch size of 100, and the model was trained over 500 epochs to adequately learn the diverse dataset features.

The synthetic network layer data were reencoded into packet data and used in network simulations to compare the volumes of original and synthetic traffic data. Fig. 7 shows the network traffic throughput over time for real and synthetic data. The x axis shows time intervals of 10 s, while the y axis represents the traffic throughput in bytes calculated as a traffic matrix across all hosts. The graph of real traffic through the network shows high variability, as expected in a normal real environment. Such variability is due to real user activity or system processes, as network usage is not constant and fluctuates greatly. The synthetic traffic data generated by CTGAN are shown on the graph for comparison. The synthetic traffic data show fluctuations similar to those of the original traffic data, indicating that they resemble real network usage.

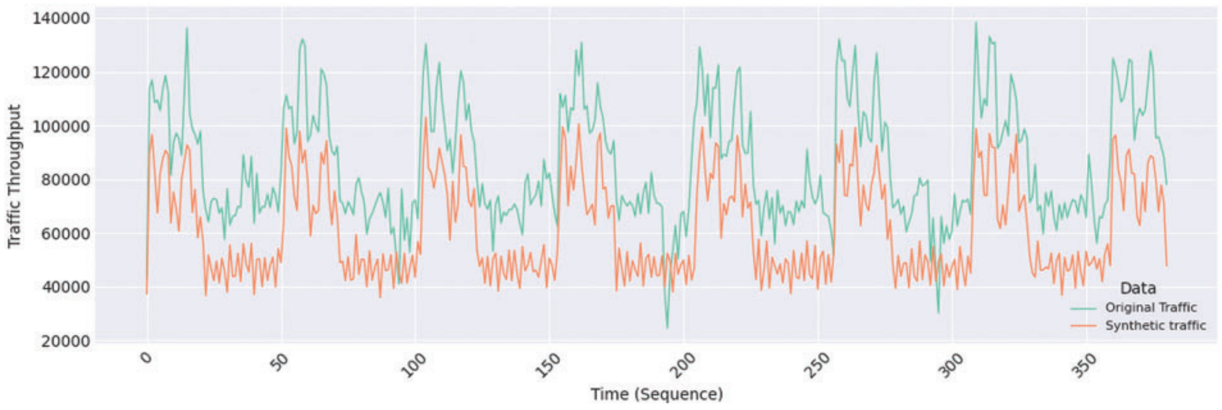

Figure 7: Traffic volumes from real and synthetic data

Fig. 7 shows peaks of high traffic volumes at specific times in the real and synthetic traffic curves. These peaks occur under heavy data transmission across the network. The similarities in the fluctuation patterns of both real and synthetic traffic data indicate that synthetic traffic faithfully describes a real environment. Both traffic patterns generally follow a similar trend, although the synthetic traffic consistently shows slightly lower volumes than the real traffic. Thus, sudden changes in the real traffic are not replicated in the synthetic traffic, indicating that synthetic data may not capture all the volatility of real data. The failure of certain sequences in the synthetic traffic to match the peaks of the original traffic suggests that synthesis may not fully reflect exceptional traffic conditions. This discrepancy may lead to issues with traffic loss or inaccuracies in packet encoding.

To compare traffic volume changes between real and synthetic data, we analyzed the change rates of the two curves. We calculated the difference in total traffic volume between real and synthetic data and then divided this difference by the total traffic volume of real data to compute the percentage difference. This calculation is given by:

$$P = \left| \frac{T_1 - T_2}{T_1} \right| \times 100 \quad (3)$$

where T_1 and T_2 represent the total volume of real and synthetic traffic, respectively. The calculation shows the percentage difference in total traffic volume between real and synthetic data, enabling an intuitive interpretation of the changes over time.

Calculating the average of the percentage differences indicates the average relative difference between real and synthetic traffic data, thus quantifying the overall difference between these data. The average percentage difference allows us to understand the consistency and average magnitude of changes over time, thereby indicating if a consistent difference between data occurs or if the difference has large fluctuations over time.

Fig. 8 shows the percentage differences between the real and synthetic traffic data over time. The average difference is 29.28%, indicating that the data exhibit high variability. The negative values at each time point occur when the percentage difference is lower than the average. Overall, approximately 200 data points are below the average. The highest difference occurs in segment 196, which appears to coincide with the largest drop in the traffic volume shown in Fig. 7. Generally, percentage differences cluster around the average, but at certain times, the difference between the real and synthetic data is much greater than the average, suggesting that synthetic traffic patterns may not suitably describe real data in those periods. As an average percentage difference of 29.28% indicates a substantial difference between real and synthetic data, the remaining 70.72% of synthetic data is assumed to show similar patterns to real data.

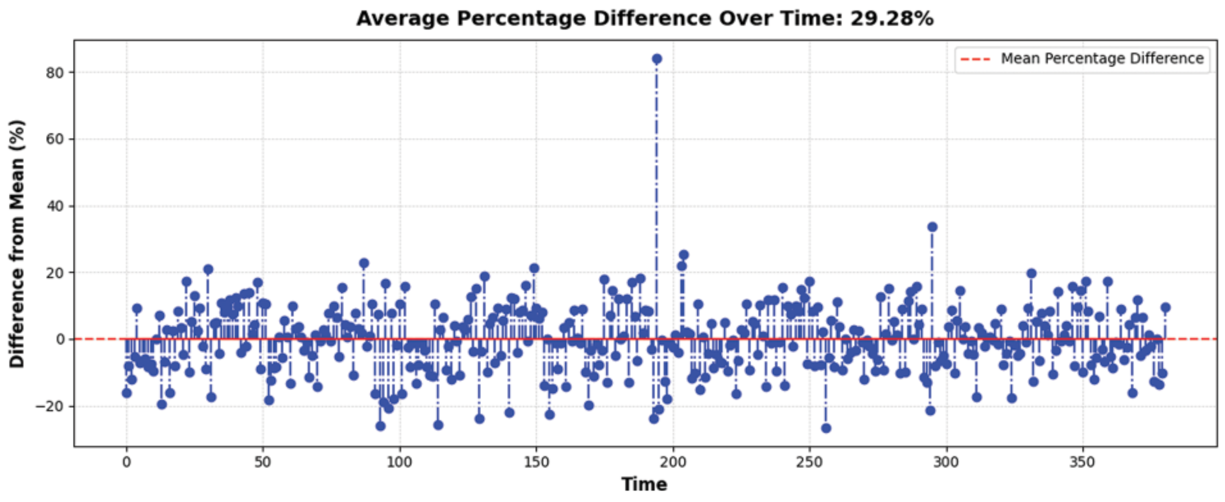

Figure 8: Average traffic rate differences between real and synthetic data

Next, the differences in the number of packets and volume of real and synthetic packets were analyzed. Table 3 shows that host 192.168.202.2 has the highest byte loss rate of 36.77%. This suggests that a large portion of data is not transmitted, greatly impacting the volume shown in Fig. 7. This issue seems to arise from improperly applied encrypted payloads during packet layer encoding. Despite the data loss, the traffic volume patterns of real and synthetic data are quite similar. This indicates that while the number of original and synthetic packets is the same, both data sets maintain similar traffic patterns in terms of the amount of concentrated traffic. In contrast, a notable difference occurs in the data size, with an average difference of 29.28%. This may be due to issues with the encoding of encrypted payloads or from other network issues. Despite these differences, the overall volume and patterns of real traffic are maintained in synthetic traffic, suggesting consistency in the occurrence of other errors.

Table 3: Number of packets and their size during network communication for real and synthetic data

Host	No. real packets	No. synthetic packets	Real data size (bytes)	Synthetic data size (bytes)	Byte loss rate
192.168.21.203	3598	3598	1,639,598	1,497,904	8.64%
192.168.202.2	7025	7025	3,176,795	2,008,681	36.77%
192.168.23.253	1471	1471	134,094	109,494	18.35%
192.168.27.101	3872	3872	293,001	292,400	0.21%
192.168.202.108	254	254	38,551	30,534	20.80%

5 Discussion

In this study, we proposed a CTGAN-based synthetic traffic generation model to create the network traffic required for cybersecurity training systems and evaluated it through network simulations.

Cybersecurity training systems must provide a variety of training scenarios, requiring repetitive and realistic network traffic. However, generating real network traffic is resource-intensive and costly. We found that synthetic traffic generated using the CTGAN model can closely replicate both normal and malicious traffic by adjusting a few attributes. By combining specific segments of traffic, it is possible to create more scenarios, providing flexibility to support various training scenarios.

We proposed a method to simulate the generated synthetic traffic alongside real traffic in an SDN-based network environment. The simulation results were verified using a Traffic Matrix calculated to measure traffic volume under different network conditions. Although some transmission loss occurred, a comparison with the original traffic patterns confirmed an error rate of 29.28%, indicating that specific segments of the generated traffic needed improvement. Nevertheless, the presence of new traffic patterns through error rates can be positive, allowing trainees to learn from unexpected scenarios. However, these new patterns require experts to re-interpret and validate them, potentially increasing the time and effort needed for training and affecting training efficiency.

To achieve near-perfect replication, first, issues arising from anomalous strings in encrypted packet payloads that affect packet length and cause errors must be resolved. To address this, a new approach is anticipated to address the imbalance issue of encrypted traffic [30]. Second, the issue of data generated during training being consistently distributed around the average must be addressed. If extreme values are not included, the model may fail to learn diverse data patterns, necessitating optimization of generator parameters to prevent underfitting.

6 Conclusion

The potential of generating realistic network traffic using a CTGAN-based synthetic traffic generation model has been confirmed in this research. Although an error rate of 29.28% was observed, further studies are required to reduce this rate to around 1%. Achieving this goal involves extracting and synthesizing fine segments for specific intervals, applying data augmentation, and adjusting learning parameters to balance the distribution of training data.

Lowering the error rate is crucial for providing realistic scenarios to cybersecurity trainees and enhancing their ability to detect and respond to realistic malicious packets. Therefore, improving the quality of synthetic data using models like CTGAN and increasing its statistical similarity to real data is essential. Future efforts will focus on strengthening monitoring technologies to adhere to additional network protocol standards and analyzing synthetic packet data. These improvements will be applied and validated in actual simulation training systems for a wider range of scenarios, ensuring that the iterative technology frameworks for cybersecurity training systems are efficiently enhanced and optimized.

Acknowledgement: The authors would like to express their gratitude to the National Research Foundation of Korea (NRF) and LIG Nex1 for their generous financial support, which was crucial for the successful completion of this research. We are also deeply appreciative of the invaluable feedback and insightful comments provided by our colleagues, whose expertise and encouragement significantly enhanced the quality of this paper. Their support in various aspects, from conceptual discussions to technical assistance, has been instrumental. Additionally, we would like to acknowledge the administrative and logistical support from our institution, which facilitated our research activities. Finally, we extend our sincere thanks to the anonymous reviewers for their thorough and constructive critiques, which helped us refine and improve the manuscript. For any technical inquiries related to this research, please contact the authors or the corresponding author via email.

Funding Statement: This work was supported in part by the Korea Research Institute for Defense Technology Planning and Advancement (KRIT) funded by the Korean Government's Defense Acquisition Program Administration (DAPA) under Grant KRIT-CT-21-037; in part by the Ministry of Education, Republic of Korea; and in part by the National Research Foundation of Korea under Grant RS-2023-00211871.

Author Contributions: Study conception and design: Dong-Wook Kim; data collection: Gun-Yoon Sin; analysis and interpretation of results: Dong-Wook Kim, Gun-Yoon Sin, Myung-Mook Han, Kwangsoo Kim, Jaesik Kang, Sun-Young Im; draft manuscript preparation: Dong-Wook Kim. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] B. Ferguson, A. Tall, and D. Olsen, "National cyber range overview," in *2014 IEEE Mil. Commun. Conf.*, Baltimore, MD, USA, 2014, pp. 123–128. doi: 10.1109/MILCOM.2014.27.

[2] R. Beuran, D. Tang, C. Pham, K. Chinen, Y. Tan and Y. Shinoda, "Integrated framework for hands-on cybersecurity training: CyTrONE," *Comput. Secur.*, vol. 78, pp. 43–59, 2018. doi: 10.1016/j.cose.2018.06.001.

[3] V. E. Urias, W. M. S. Stout, B. Van Leeuwen, and H. Lin, "Cyber range infrastructure limitations and needs of tomorrow: A position paper," in *2018 Int. Carnahan Conf. Secur. Technol. (ICCST)*, Montreal, QC, Canada, 2018, pp. 1–5. doi: 10.1109/CCST.2018.8585460.

[4] S. Hui *et al.*, "Knowledge enhanced GAN for IoT traffic generation," in *Proc. ACM Web Conf.*, Lyon, France, 2022, pp. 3336–3346. doi: 10.1145/3485447.3511976.

[5] C. Javali and G. Revadigar, "Network web traffic generator for cyber range exercises," in *2019 IEEE 44th Conf. Local Comput. Netw. (LCN)*, Osnabrueck, Germany, 2019, pp. 308–315. doi: 10.1109/LCN44214.2019.8990880.

[6] R. Rouquette, S. Beau, M. M. Yamin, U. Mohib, and B. Katt, "Automatic and realistic traffic generation in a cyber range," in *10th Int. Conf. Future Internet Things Cloud (FiCloud)*, Marrakesh, Morocco, 2023, pp. 352–358. doi: 10.1109/FiCloud58648.2023.00058.

[7] S. K. Sharma and J. Sefchek, "Teaching information systems security courses: A hands-on approach," *Comput. Secur.*, vol. 26, no. 4, pp. 290–299, 2007. doi: 10.1016/j.cose.2006.11.005.

[8] U. Ghosh, P. Chatterjee, and S. Shetty, "A security framework for SDN-enabled smart power grids," in *2017 IEEE 37th Int. Conf. Distrib. Comput. Syst. Workshops (ICDCSW)*, Atlanta, GA, USA, 2017, pp. 113–118. doi: 10.1109/ICDCSW.2017.20.

[9] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN," in *Adv. Neural Inform. Process. Syst., NIPS'19: 33rd Int. Conf. Neural Inform. Process. Syst.*, Vancouver, BC, Canada, Dec. 8–14, 2019, vol. 32, pp. 7335–7345. doi: 10.1109/ICDCSW.2017.20.

[10] M. Glas, M. Vielberth, and G. Pernul, "Train as you fight: Evaluating authentic cybersecurity training in cyber ranges," in *Proc. 2023 CHI Conf. Human Factors Comput. Syst.*, Hamburg, Germany, 2023, pp. 1–19. doi: 10.1145/3544548.3581046.

[11] R. Ošlejšek, M. Macák, and K. D. Burská, "Hands-on cybersecurity training behavior data for process mining," *Data Brief*, vol. 52, 2024, Art. no. 109956. doi: 10.1016/j.dib.2023.109956.

[12] J. Kim, K. Kim, and M. Jang, "Cyber-physical battlefield platform for large-scale cybersecurity exercises," in *2019 11th Int. Conf. Cyber Conflict (CyCon)*, Tallinn, Estonia, 2019, pp. 1–19. doi: 10.23919/CY-CON.2019.8756901.

[13] M. M. Yamin and B. Katt, "Modeling and executing cyber security exercise scenarios in cyber ranges," *Comput. Secur.*, vol. 116, 2022, Art. no. 102635. doi: 10.1016/j.cose.2022.102635.

[14] E. Russo, G. Costa, and A. Armando, "Scenario design and validation for next generation cyber ranges," in *2018 IEEE 17th Int. Symp. Netw. Comput. Appl. (NCA)*, Cambridge, MA, USA, 2018, pp. 1–4. doi: 10.1109/NCA.2018.8548324.

[15] M. M. Yamin, B. Katt, and M. Nowostawski, "Serious games as a tool to model attack and defense scenarios for cyber-security exercises," *Comput. Secur.*, vol. 110, 2021, Art. no. 102450. doi: 10.1016/j.cose.2021.102450.

[16] S. -F. Wen, M. M. Yamin, and B. Katt, "Ontology-based scenario modeling for cyber security exercise," in *2021 IEEE Eur. Symp. Secur. Priv. Workshops (EuroS & PW)*, Vienna, Austria, 2021, pp. 249–258. doi: 10.1109/EuroSPW54576.2021.00032.

[17] O. A. Adeleke, N. Bastin, and D. Gurkan, "Network traffic generation: A survey and methodology," *ACM Comput. Surv.*, vol. 55, no. 2, pp. 1–23, 2022. doi: 10.1145/3488375.

[18] P. Suresh *et al.*, "Contemporary survey on effectiveness of machine and deep learning techniques for cyber security," in *Machine Learning for Biometrics*, Cambridge, MA, USA: Academic Press, 2022, pp. 177–200. 10.1016/B978-0-323-85209-8.00007-9.

[19] A. Cheng, "PAC-GAN: Packet generation of network traffic using generative adversarial networks," in *2019 IEEE 10th Annu. Inf. Technol. Electron. Mob. Commun. Conf. (IEMCON)*, Vancouver, Canada, 2019, pp. 728–734. doi: 10.1109/IEMCON.2019.8936224.

[20] L. D. Manocchio, S. Layeghy, and M. Portmann, "FlowGAN-synthetic network flow generation using generative adversarial networks," in *2021 IEEE 24th Int. Conf. Comput. Sci. Eng. (CSE)*, Shenyang, China, 2021, pp. 168–176. doi: 10.1109/CSE53436.2021.00033.

[21] B. Dowoo, Y. Jung, and C. Choi, "PcapGAN: Packet capture file generator by style-based generative adversarial networks," in *2019 18th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Boca Raton, FL, USA, 2019, pp. 1149–1154. doi: 10.1109/ICMLA.2019.00191.

[22] M. Ring, D. Schlör, D. Landes, and A. Hotho, "Flow-based network traffic generation using generative adversarial networks," *Comput. Secur.*, vol. 82, pp. 156–172, 2019. doi: 10.1016/j.cose.2018.12.012.

[23] J. Yang, L. Wang, A. Lesh, and B. Lockerbie, "Manipulating network traffic to evade stepping-stone intrusion detection," *Internet Things*, vol. 3, pp. 34–45, 2018. doi: 10.1016/j.iot.2018.08.011.

[24] P. Biondi, *Scapy: The Python-Based Interactive Packet Manipulation Program & Library*. San Francisco, CA, USA: GitHub. 2024. Accessed: Jul. 23, 2024. [Online]. Available: https://github.com/secdev/scapy

[25] R. L. de Oliveira, C. M. Schweitzer, A. A. Shinoda, and L. Rodrigues Prete, "Using Mininet for emulation and prototyping software-defined networks," in *2014 IEEE Colomb. Conf. Commun. Comput. (COLCOM)*, Bogota, Colombia, 2014, pp. 1–6. doi: 10.1109/ColComCon.2014.6860404.

[26] S. Kaur, J. Singh, and N. S. Ghumman, "Network programmability using POX controller," in *Proc. Int. Conf. Commun., Comput., and Syst. (ICCCS)*, Ferozepur, India. 2014, pp. 134–138. Accessed: Jul. 23, 2024. [Online]. Available: http://sbsstc.ac.in/icccs2014/Papers/Paper28.pdf

[27] F. Klassen and AppNeta, "Tcpreplay-Pcap editing and replaying utilities," *AppNeta*. Accessed: Jul. 23, 2024. [Online]. Available: https://tcpreplay.appneta.com

[28] Y. Tian, W. Chen, and C. -T. Lea, "An SDN-based traffic matrix estimation framework," *IEEE Trans. Netw. Serv. Manage.*, vol. 15, no. 4, pp. 1435–1445, 2018. doi: 10.1109/TNSM.2018.2867998.

[29] Netresec, "Capture files from Mid-Atlantic CCDC: MACCDC 2012," *Netresec*, 2012. Accessed: Jul. 23, 2024. [Online]. Available: https://www.netresec.com/?page=MACCDC

[30] J. Zhai, P. Lin, Y. Cui, L. Xu, and M. Liu, "GraphCWGAN-GP: A novel data augmenting approach for imbalanced encrypted traffic classification," *Comput. Model. Eng. Sci.*, vol. 136, no. 2, pp. 2069–2092, 2023. doi: 10.32604/cmes.2023.023764.