Network attack classification framework based on Autoencoder model and online stream analysis technology

Ngo Thanh Tung, Dang Thi Mai, Nguyen Viet Hung

***Abstract*— To deal with diverse and constantly changing forms of cyberattacks, machine learning methods have been researched and applied extensively in network data processing for positive results in network attack detection. However, machine learning models require extensive computational resources and their application to handle significant real-time data flow monitoring problems still needs improvement. In this paper, we research and propose a network attack detection framework using a 2-stage classification algorithm with an Autoencoder model, integrating online stream processing technology based on Apache Kafka and Spark technology. The results show that the proposed framework has high efficiency in detecting network attacks and faster processing time than traditional data processing technology.**

***Tóm tắt*— Để đối phó với các hình thức tấn công mạng đa dạng và thay đổi liên tục, các phương pháp học máy đã được nghiên cứu và áp dụng rộng rãi trong xử lý dữ liệu mạng để đạt được hiệu quả cao trong việc phát hiện tấn công mạng. Tuy nhiên, các mô hình học máy đòi hỏi tài nguyên tính toán lớn và việc áp dụng chúng để xử lý các vấn đề giám sát luồng dữ liệu thời gian thực vẫn cần được cải thiện. Trong bài báo này, nhóm tác giả nghiên cứu và đề xuất mô hình phát hiện tấn công mạng sử dụng thuật toán phân lớp 2 giai đoạn kết hợp với mạng Autoencoder, tích hợp công nghệ xử lý luồng dữ liệu trực tuyến dựa trên Apache Kafka và Apache Spark. Kết quả cho thấy rằng mô hình được đề xuất có hiệu suất cao trong việc phát hiện tấn công mạng và thời gian xử lý nhanh hơn so với công nghệ xử lý dữ liệu truyền thống.**

***Keywords*— network attack detection; online stream processing; Autoencoder; network data representation.**

***Từ khóa*— phát hiện tấn công mạng; xử lý luồng trực tuyến; Autoencoder; biểu diễn dữ liệu mạng.**

I. INTRODUCTION

To timely detect and respond to increasingly complex attacks nowadays [1], network monitoring solutions have been developed and deployed in recent years. Network monitoring systems are typically designed to collect logs and security events from endpoints (servers, workstations, IoT devices, mobile phones...), network security devices (switches, routers...) or security appliances (Firewalls, IDS/IPS, DLP…) for analysis and centralized storage. Centralized event monitoring and management systems allow for analysis and reporting of an organization's network security events through correlation rules that help detect attacks not detected by single solutions (IDS/IPS, Firewall…).

Cyber attacks are becoming increasingly sophisticated, with a growing number of new variants of attacks. Traditional network intrusion detection algorithms that use signatures and correlation rules are no longer effective against new attack techniques. Machine learning methods have been applied and integrated into many monitoring systems to overcome this weakness. Some popular algorithms used in network intrusion detection include Naive Bayes, Decision Tree, Support Vector Machine (SVM), Random Forest, Neural Networks and Deep Learning [2].

However, using machine learning algorithms in network attack classification also faces challenges, including the effectiveness of traditional machine learning models with complex network data, insufficient training data, and the continuous evolution of network attack

This manuscript is received on April 04, 2023. It is commented on April 14, 2023 and is accepted on April 18, 2023 by the first reviewer. It is commented on April 14, 2023 and is accepted on April 21, 2023 by the second reviewer.

forms. Therefore, developing suitable network attack detection algorithms and building good classification models is crucial in network monitoring systems [3]. The problem is the need for a solution combining advanced machine learning methods and big data processing technology to accelerate the detection of network attacks and the continuous processing of large amounts of data from multiple sources.

Current surveillance systems mostly use centralized data analysis technology. The data is sent from many sources, stored and processed at the center. The increasingly large amount of data leads to higher latency in attack detection and detection of unusual events. Network data is collected from many different distributed sources, comes in different formats, and has many different log systems, causing specific difficulties in the analysis process. The amount of information generated is increasing, so the amount of data generated is increasing. Moderately fast networks also generate large amounts of data. For example, monitoring a 1 Gigabit Ethernet LAN card running at 50% capacity generated a Tetabyte of data in 2 hours. In particular, with the rapid development of IoT and mobile devices, the number of connected devices and the amount of data generated has exploded. Some experts estimate that there will be about 80 billion network sensors worldwide by 2025. According to Clay [4], centralized network monitoring systems rely on technology to collect and traditional data processing – that is, collecting data, storing it in a database, and then processing it. Therefore, it is not effective for networks with the current traffic volume. About 85% of network intrusions are detected after they have been active for several weeks. In addition, a study conducted by IBM [5] with data collected from 350 companies showed that the average time to detect a data leak is about 277 days, ranging from 20 to 582 days. A massive amount of data must be managed, processed, transformed and stored as soon as possible. Therefore, processing big data is a challenge for traditional network monitoring systems. Our contributions are the following:

- Developing an efficient two-stage learning algorithm with the Autoencoder model to represent network data as effective feature vectors for classification models.
- Proposing the architecture of a big data processing system with real-time stream processing technologies in network monitoring systems on the Apache Kafka and Apache Spark technology platforms.

The next section of the article is presented in the following structure: Section II introduces related research on online data processing methods and machine learning models for network intrusion detection. Section III presents the two-stage classification algorithm using Autoencoder and the proposed system architecture for network monitoring on the online stream processing platform, which we suggest building. Section IV presents the evaluation results of artificial intelligence algorithms for detecting network attacks with the standard datasets CIC-IDS2017 and CSE-CIC-IDS-2018 in the extracted data format on the stream. The results of the real-world data testing will also be presented in this section. Section V is the conclusions and directions for future research.

II. Related work

A. Data processing method

1. Data processing

Data processing is divided into three main methods: batch data processing, small batch data processing, and stream data processing. Big data analysis in static form is done using batch data processing methods. In which data is collected at times before the time of analysis, then indexed and stored for later querying. An example of a batch data processing technique is MapReduce, implemented by the open source tool Hadoop [6]. The limitation of batch data processing is latency when executing queries or making decisions, which is unsuitable for applications requiring real-time processing of constantly updated data [8]. The response time of a real-time system requires a response in microseconds. A small batch processing technique is proposed to

overcome this weakness. Then the system will divide the data stream into small bundles and process them. The third approach is stream data processing, which ensures the processing of extensive and continuously updated stream data with real-time response time [9].

To solve the challenge of big data processing, in the world, many research groups and technology companies have proposed solutions. The research focuses on improving the data processing model with online data stream analysis technology (Stream Analytics) to help the system process and analyze large amounts of data in real-time. We will analyze stream data processing technology's overview and applicability to network security monitoring systems [17].

2. Stream data processing

Stream data processing is a method that allows features to be extracted from data streams in real-time. The difference between the stream data processing method and the traditional data processing method (batch data processing) is that the stream data processing method allows extracting features from data in dynamic form, i.e. the data changes continuously over time and is updated continuously, while the batch data processing method can only extract the feature of the data in the static form, i.e. the data does not change over time. Stream data processing aims to make real-time decisions by examining and analyzing the data flow through the processing system (Figure 1). Some examples of stream data processing systems are: Financial analysis system which monitors the flow of stock price data on stock exchanges; a system that detects the use of fake credit cards in which monitors the flow of customer transaction data; a health monitoring system in healthcare which monitors and analyzes the data stream sent back from sensors mounted on the patient such as heart rate sensors, blood pressure sensors, location; and many other areas of life use stream data processing [17].

Stream processing is modelled through a Directed Acyclic Graph (DAG). The graph consists of continuous source data nodes giving patterns and interconnected processing nodes. A data stream is an unconstrained data set, $\tilde{\omega} = \{Dt | t > 0\}$ where point Dt is a set of attributes with a datum. Formally, a data point is $Dt = (V, \tau t)$,

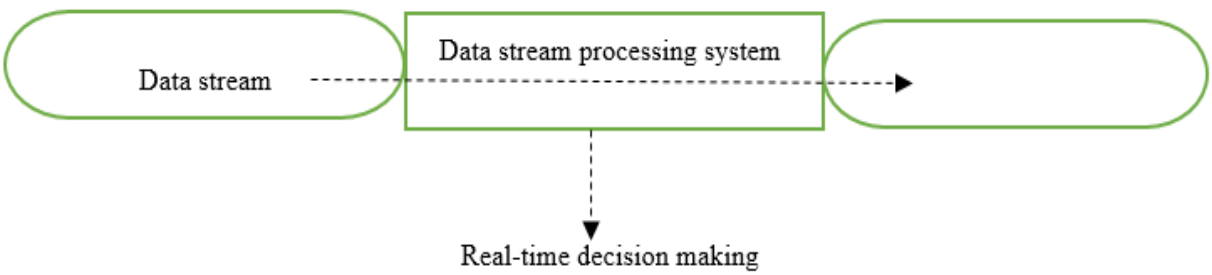

Figure 1. Streamed data processing overview

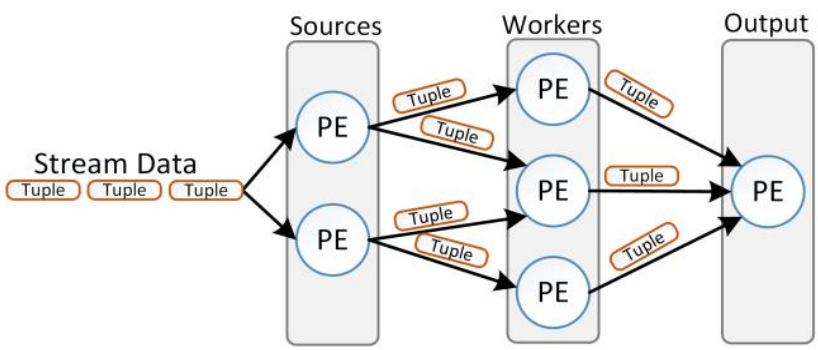

Figure 2. Stream processing architecture [17]

where V is an n-tuple, where each value corresponds to an attribute, and τt is the datum for the t-th sample.

Source Nodes issue tuples or messages from a Processing Element (PE). Each PE receives data on the input queue, performs computations using the local state, and produces a result for its output queue.

The Processor Elements (PE) are connected to create a directed ac graph. PE sources receive data streams, which are processed immediately and aggregated in the output. The output is a Processing Element that performs a specific function on the data, such as for data visualization [18].

Stream processing is widely applied in network monitoring systems, allowing direct data analysis from input data sources, such as logs and information from network devices or network queries.

Applications of stream data processing in network monitoring systems include:

(1) Network attack detection, enabling detection of suspicious behaviour or network attacks. Network monitoring systems can use stream data processing to analyze malicious files, suspicious queries, and unusual activities of network devices.

(2) Detect network performance problems: Network monitoring systems can use stream data processing to detect network performance problems, such as a slow network connection or overloading network equipment.

(3) Network data analysis and evaluation: Stream data processing can be used to analyze and evaluate network data, which enhances network security and management.

(4) Monitor and detect problems: Network monitoring systems can use stream data processing to monitor and detect network-related problems, helping to identify and remediate problems quickly.

Stream data processing is a beneficial technology in network monitoring systems, helping to detect network attacks and network performance issues and helping to identify and fix problems quickly.

B. Overview of network attack detection algorithms

Cyberattack detection algorithms are used in network monitoring systems to identify and analyze suspicious behaviour or network attacks. Here are some standard algorithms used in network attack detection:

(1) Rule-based Detection: This is the simplest and most common method in detecting network attacks. It is based on specific rules set in place to detect suspicious behaviour. For example, if there are many requests from the same IP address in a short time, it can be considered an attack from that IP.

(2) Anomaly Detection: This method uses machine learning data to identify unusual behaviours compared to previously learned data patterns. It uses statistical models and machine learning to analyze suspicious behaviour and detect anomalous behaviour. For example, if a specific IP address accesses accounts that do not match its activity, it can be considered anomalous behaviour.

(3) Signature-based Detection: This method uses signatures or detailed descriptions of known attacks to detect suspicious behaviour. It compares data patterns with known patterns to look for similar behaviours. For example, if an IP tries to access an account with the wrong password repeatedly, it can be seen as a brute-force attack.

(4) Hybrid Detection: This method combines the above methods to detect network attacks. It uses machine learning methods, specific rules, and attack signatures to detect suspicious behaviour and network attacks.

Network attack detection algorithms also need to be integrated into the network monitoring system to detect and warn network administrators as soon as suspicious behaviours or network attacks occur. This procedure allows administrators to react promptly to resolve problems and prevent attacks from spreading. Combining network attack detection algorithms

and other security methods like firewalls, user behaviour analysis, and login management can enhance detection and prevention effectiveness.

Several network attack detection techniques have been researched and achieved high results recently. These studies emphasize the importance of machine learning and deep learning techniques in detecting network attacks. In addition, the studies also mentioned the integration of different security methods to enhance the effectiveness of detecting and preventing network attacks.

The authors in [19] present an SVM-based IDS that uses evolutionary algorithms to optimize SVM parameters and improve intrusion detection accuracy.

The authors of [20] provided an AOCD (An Adaptive Outlier Based Coordinated Scan Detection Approach) solution for the early detection of network reconnaissance attacks with high accuracy when testing with the KDD99 dataset. The author presented a solution to convert network traffic data into a format that filters and classifiers can handle. It is worth noting that the authors have selected random samples in the database, thereby determining the set of attributes for clustering detection used in the previous network scan detection technique.

The authors of [21] focuses on detecting anomalous activities in industrial control systems using deep learning techniques. The author finds that logic controllers (PLCs) and end devices often control industrial systems. Values often measure the activities in these systems read from sensors and stored in memory as logs. Specifically, the author uses the Autoencoder model - a deep learning model used to learn efficient data representations - to detect anomalous activities in the logs of industrial control systems. The author tested an actual data set of industrial control systems and showed that the proposed method could detect abnormal activities with high accuracy.

C. Autoencoder network

Autoencoder is a deep neural network designed to encode the input data (Encoder) into latent representations and then decode (Decoder) to reconstruct the input using the newly created encoded data. The idea of the proposal: Instead of putting all dimensions of the original data into classical classification algorithms, we will use the AE network to reduce the data dimensionality (feature selection) and then put it into classification algorithms. The purpose is to improve the performance of classical data classification models.

The Autoencoder network architecture is divided into three parts: Encoder, Decoder, and Latent Space, also known as the "Bottleneck" node, as shown in Figure 3. The network is trained using unsupervised (unlabeled) data to learn data representations. The input data is transformed (encoded) into a lower dimensional (latent) space through neural layers. The representations in this latent space will then be decoded to reconstruct the input.

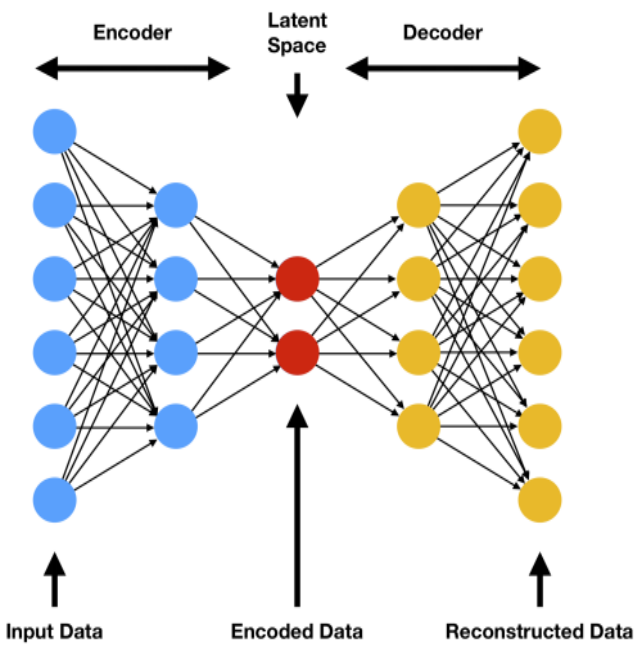

Figure 3. Model Autoencoder

Encoder: Consists of a regular feedforward neural network designed to predict the latent representation of the input data. In which f_θ is the encoder, and $X = \{x_1, x_2, \ldots, x_n\}$ is a dataset. The encoder f_θ aims to map the input data $x_i \in X$ to a hidden representation (latent representation) [22].

$$z_i = f_\theta(x_i) \quad (1)$$

Decoder: Consists of a feedforward neural network, but the size of the layers increases in reverse order (symmetrical) to the layers of the Encoder. The decoder g_ϕ aims to map the latent representation z_i back to the input space [23], therefore, the reconstruction is calculated by the formula:

$$\hat{x}_i = g_\phi(z_i) \quad (2)$$

Latent Space: This represents the input data or low-dimensional compressed representation of the model's input. The structure of the Decoder uses this low-dimensional data to reconstruct the input (represented by z_i).

The encoder and decoder are represented as activation functions of linear functions related to weights and biases [23] as follows:

$$f_\theta(x) = \psi_f(Wx + b) \quad (3)$$

$$g_\phi(x) = \psi_g(W'z + b') \quad (4)$$

In which θ = (W, b) and ϕ = (W ', b') are the (weights and biases) for the encoder and decoder, respectively trained ψ_f and ψ_g are the activation functions of the encoder and decoder.

The reconstruction loss on the training samples can be written as follows [23]:

$$\mathcal{L}_{AE}(\theta; \phi; x) = \frac{1}{n}\sum_{i=0}^{n} l(x_i, \hat{x}_i) \quad (5)$$

In which $l(x_i; \hat{x}_i)$ is the difference between the input x_i and its reconstruction $\hat{x}_i$; n is the number of data samples in the dataset Autoencoder learns to optimize the objective function in (5) related to parameters θ = (W, b) and ϕ = (W', b') using a training algorithm such as stochastic gradient descent with backpropagation.

Autoencoder networks are widely applied in data science, such as dimensionality reduction, anomaly detection, image denoising, image compression, image generation, feature extraction, and recommendation systems.

D. Technology

Our research used two leading platforms in big data collection, storage and processing, Apache Kafka and Apache Spark.

Kafka is one of the industry's most popular data stream processing platforms today, used by over 80% of Fortune 100 companies. Kafka is used by thousands of the world's top organizations for high-performance data pipelines, stream analytics, data integration, and many other mission-critical applications. Kafka is an open source distributed publish/subscribe message system built for real-time streaming data processing [8].

It is a fully packaged project, highly fault tolerant and a fast data transmission system. Because of its reliability, Kafka is gradually being substituted for the traditional data transmission system. It is used for standard data transmission systems in different contexts due to its horizontal scalability and reliable data transfer.

Next, Apache Spark is used to process large volumes of data in the system. It is an open source large-scale data processing framework. Spark provides an interface for programming parallel computing clusters with fault tolerance. Spark was initially developed at the University of California Berkeley's AMPLab, then the source code was donated to the Apache Software Foundation in 2013, and the organization has maintained it to this day [9].

Apache Spark's distributed computing capabilities make it well-suited to big data and machine learning, which require massive computing power to work on large data warehouses. Spark also helps take some of the programming burdens off developers' shoulders with an easy-to-use API that takes care of much of the hard work of distributed computing and big data processing. Apache Spark consists of two main components: drivers and executables. Drivers are used to converting user code into multiple tasks that can be distributed across worker nodes. The executor runs on the processing nodes and performs the assigned tasks. Spark can also run in standalone cluster mode requiring only the Apache Spark framework and JVM on each machine in the cluster. However, using cluster management tools as an intermediary between the two components improves

resource utilization and allows on-demand allocation. Apache Spark builds commands that process user data into a DAG. DAG is Apache Spark's scheduling layer, it determines which tasks are executed on which nodes and in what sequence [8].

With many advantages in real-time data processing and compatibility with many programming languages, we have applied Spark effectively in processing and analyzing network packets.

The current streaming data application systems are diverse and are used in many different fields. Below are some examples of streaming data application systems:

Log processing: Streaming data is used to process logs in web and mobile applications, helping to quickly detect and resolve issues. Technologies such as Apache Kafka, Apache Flink, Elasticsearch are commonly used in log processing.

IoT data analysis: Streaming data is used to analyze IoT data in fields such as productivity measurement, device health monitoring, security monitoring,... Technologies such as Apache Kafka, Apache Spark Streaming, AWS Kinesis are commonly used in IoT data analysis.

Real-time processing in social media platforms: Streaming data is used to process real-time activities in social media platforms such as Twitter, Facebook, Instagram,... Technologies such as Apache Flink, Apache Kafka, Apache Spark Streaming are commonly used in real-time processing in social media platforms.

E-commerce data processing: Streaming data is used to process purchasing activities in e-commerce sites such as Amazon, eBay,... Technologies such as Apache Kafka, Apache Flink, Apache Spark Streaming are commonly used in streaming data processing in e-commerce sites.

Financial data processing: Streaming data is used to process financial data in applications such as risk management, stock trading, financial forecasting,... Technologies such as Apache Flink, Apache Kafka, Apache Spark Streaming are commonly used in streaming data processing in financial applications.

We all acknowledge that Apache Spark is highly regarded and widely used in many systems across all fields. The role of Apache Spark in processing streaming data is to provide a distributed, efficient, and capable system to process large amounts of data, making it easy and fast for users to process streaming data. It provides features such as processing batch and streaming data in the same system, enabling simultaneous computation on multiple threads, providing integrated libraries for data analysis, and supporting distributed data computation.

Currently, streaming data systems all include main components such as data sources, data ingestion, data processing, data storage, and data visualization. Our system implements a simulation of a small-scale system with similar tools as current streaming data systems, however, the algorithms used to process network data are more optimized, helping to increase the speed in terms of time to timely detect network attacks.

III. PROPOSAL FOR CLASSIFICATION ALGORITHM AND FRAMEWORK MODEL FOR NETWORK ATTACK CLASSIFICATION

We propose to build a model to process distributed data streams to monitor network traffic and detect attacks accurately. The model consists of 3 layers: processing raw data, distributed processing and data visualization. We used open source tools Apache Kafka and Apache Spark to test and optimize the model.

A. Network attack detection algorithm

AE networks such as dimensionality reduction, anomaly detection, image denoising, image compression, image generation, feature extraction, and recommendation systems are widely applied in data science.

We propose a two-stage model for network attack detection and classification, which utilizes AE networks combined with traditional machine learning algorithms such as AE_Decision Tree,

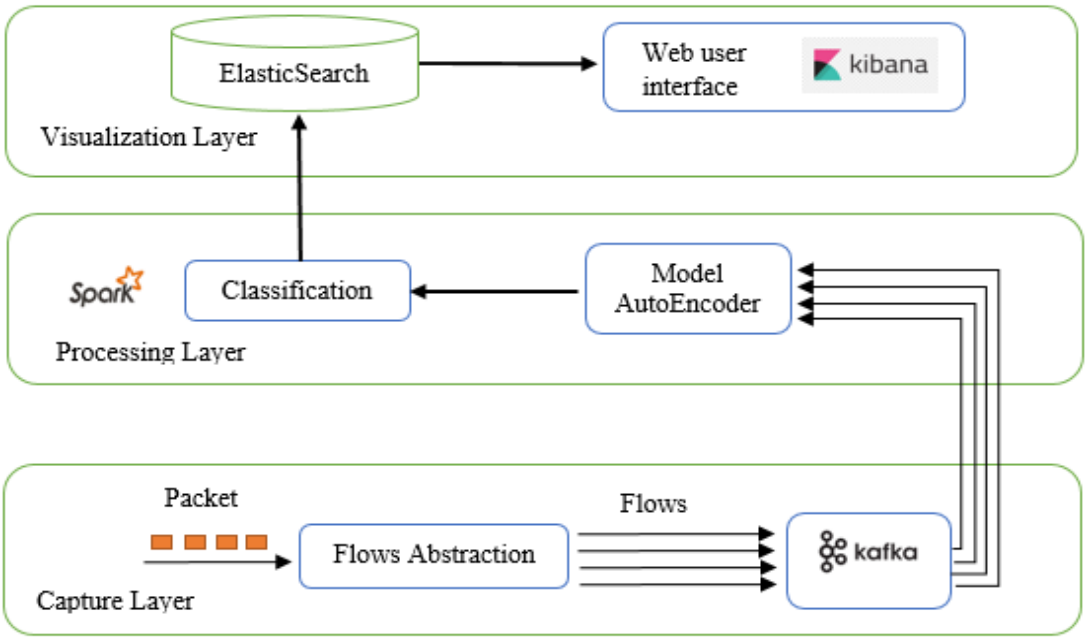

Figure 4. System Architecture

AE_RandomForest, AE_kNN, AE_Naive Bayes, and AE_MLP, in there:

- Stage 1 uses the AE network to train unsupervised data to extract features.
- Stage 2 is supervised learning, using data classification algorithms to detect and classify network reconnaissance attacks.

B. System Architecture

We designed the system with three main layers: the data collection layer, the data processing layer, and the data visualization layer. In particular, after the packets are collected and extracted from the network, the system will store, process, and classify them to give statistics to evaluate normal or abnormal packet properties. The data collection layer groups packet into flows with the same characteristics. Each flow is a sequence of packets with the same source IP, destination IP, source port, destination port and protocol within a timeframe. Flows have 41 features generated by TCP/IP headers data such as TCP, UDP and ICMP packet rates, source and destination port numbers, and the number of each TCP flag. Apache Kafka is important in converting flows into streams, acting as buffers or queues, throttling, scaling, and data receive/push speed.

The data processing layer uses Apache Spark. Spark is implemented in a cluster in a master/slave model, where the slaves are scalable and share resources, making the system scalable. In this layer, we use a classification model trained with standard datasets (CICIDS-2017 and CSE-CIC-IDS-2018) and traditional machine learning algorithms to process information streams. Streams are taken from Kafka and passed through a previously trained model to classify normal or abnormal flows in the network.

The visualization layer uses Elastic Search and Kibana to visualise the classification results. The data is stored in a structured manner, which is convenient for statistics, search, and evaluation over time.

V. Results and Evaluation

As analyzed, artificial intelligence models have been studied and used to improve the ability to detect increasingly numerous new attacks. In this study, we will evaluate machine learning algorithms before and after combining with Autoencoder regarding attack detection effectiveness and execution time to select an appropriate model for application in a monitoring system on an online stream analysis platform.

A. Experimental data set

In the experiment, we use two datasets, CIC-IDS-2017 and CSE-CIC-IDS-2018, which are data sets provided by the Canadian Institute for Cybersecurity. This dataset consists of benign network traffic and network traffic of common attacks, provided in the form of PCAP containing network packets during crawling and CSV files containing statistical information. Both datasets use CICFlowMeter (version 1 for 2017 and version 3 for 2018). CITFlowMeter is a network traffic generator and analyzer. It can be used to create bidirectional flows, where the first packet defines the forward direction (source to destination) and reverse direction (destination to source), so there are more than 80 statistical network traffic properties such as Duration, Number of Packets, Number of Bytes, Length of packets,... can be calculated separately in the forward and backward direction. The output of this tool is a CSV-formatted file with six labelled columns for each flow (FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort, and Protocol) with more than 80 network traffic analysis properties.

CIC-IDS-2017 consists of 8 files (Table 1), with more than 80 features analyzed from 225,745 packets collected in 5 days with different network attack scenarios and normal network traffic.

TABLE 1. DESCRIPTION OF THE CIC-IDS-2017 DATASET

Attack Type	Data set	Label	Total number of records	Normal	Attack
DOS	Wednesday-workingHours.pcap_ISCX_KQ.csv	BENIGN, DoS slowloris, DoS Slowhttptest, DoS Hulk, DoS GoldenEye	692703	440031	252672
DDOS	Friday-WorkingHours-Afternoon-DDos.pcap_ISCX_KQ.csv	BENIGN, DDOS	225745	97718	128027
Bruteforce	Tuesday-WorkingHours.pcap_ISCX_KQ.csv	BENIGN, FTP-Patator, SSH-Patator	445909	432074	13835
Infiltration	Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX_KQ.csv	BENIGN, Infiltration	288602	288566	36
Port Scan	Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX_KQ.csv	BENIGN, PortScan	286467	127537	158930
Web Attack	Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX_KQ.csv	BENIGN, Web Attack – Brute Force, Web Attack – XSS, Web Attack – Sql Injection	170366	168186	2180

TABLE 2. DESCRIPTION OF DATASET CSE-CIC-IDS-2018

Attack Type	Data set	Label	Total number of records	Normal	Attack
Bruteforce	14-02-2018.csv	Benign,FTP-BruteForce, SSH-Bruteforce	1048575	676626	371949
DoS	16-02-2018.csv	Benign, DoS-SlowHTTPTest, DoS-Hulk	1048575	446772	601803
Bot	02-03-2018.csv	Benign, Bot	1048575	762384	286191

During preprocessing, we remove some attribute columns that do not affect the training process (attribute columns contain only one value). In addition, the attribute columns belonging to the traffic flow details representing the system information used to build the dataset are also removed, namely Flow ID, Source IP, Source Port, Timestamp, and Protocol. Next, we remove incomplete records that are rows containing the value NaN (Not a Number – undefined). For the attack detection task, during training, the labels bearing the value BENIGN will be changed to 0, and the remaining labels will show the details of the attack type for the multi-type attack (DDoS, Brute-force,…) will be changed to 1.

The CSE-CIC-IDS-2018 dataset was developed to replace older datasets used for NIDS (Network-based Intrusion Detection System). This dataset is an updated CIC-IDS-2017 dataset with similar primary attributes. The authors performed seven common types of attacks, including Brute-force attacks, DoS attacks, Web attacks, Local network infiltration attacks, Botnet attacks, and Port Scan. The simulation environment was built with 50 network nodes, 30 servers, and 420 end devices. The authors extracted 80 basic features from the collected network flow data with the help of the CICFlowMeter-V3 (version 3) tool. We preprocessed the data similarly to the previous dataset to ensure that the selected attributes were the best in the data format extracted from the flow (including 41 features).

B. Experiment

Experiments are performed to compare the effectiveness of the proposed model and other machine learning models in network attack detection and classification.

1. Data processing

The input datasets will be preprocessed on both the training and test sets. The data will be normalized using one of several normalization methods (Scale), such as *StandardScaler* and *MaxAbsScaler*, provided in the *sklearn* library.

2. Architecture of Autoencoder

The Autoencoder network training program is built in Python with the PyTorch library. The test was conducted on an Intel Core i7-2.8GHz PC with 16GB of RAM. The AE network architecture is built with different layers and corresponding nodes for each input data set (depending on the input data vector's values). The input vector of the network has a size equal to the number of attributes of the tuple described above. The final configuration of the AE model was selected through testing processes, and the architecture of AE model applied to each dataset, shown Table 3.

The parameters included in training the network are: *leaning_rate = 0.01, batch_size = 100, display_step = 10, epoch = 200*.

The weight of the AE is initialized with the Xavier initialization method [23]. The optimization algorithm is Adadelta, and the activation function is the TANH function.

TABLE 3. AE NETWORK ARCHITECTURE TABLE WITH DATASETS

Datasets	**Network Architecture AE (Encoder _Decoder)**			
	Input	**Hidden layer 1**	**Hidden layer 2**	**Hidden space**
CIC-IDS-2017	41	36	26	12
CIC-IDS-2018	41	36	26	12

3. Machine learning algorithms

Some algorithms are used to classify network reconnaissance attack data such as Decision Tree, Random Forest, Naive Bayes, K-Nearest Neighbor (k-NN), Multi-layer Perceptron (MLP), Support Vector Machine (SVM) [10-16].

C. Classification results

The classifier's results are usually evaluated through the Confusion Matrix and some measurements based on the confusion matrix. The confusion matrix is described in Table 4, where Class 0 is normal, and Class 1 is attack.

TABLE 4. CONFUSION MATRIX

		Prediction class	
		Class 0	Class 1
Practical class	Class 0	TN	FP
	Class 1	FN	TP

The following metrics will be used to evaluate the performance of the models.

$$acc = \frac{TP+TN}{TP+FP+TN+FN} \quad (6)$$

$$Precision(\pi) = \frac{TP}{TP+FP} \quad (7)$$

$$Recall(\rho) = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}} \quad (8)$$

$$F1 = \frac{\mathbf{2}}{\frac{\mathbf{1}}{\boldsymbol{precision}}+\frac{\mathbf{1}}{\boldsymbol{recall}}} \quad (9)$$

1. Experiment with the CIC-IDS-2017 dataset

The dataset CIC-IDS-2017 will be tested with two classifiers: the first is a binary classifier to determine whether the received network traffic record is attack or normal. The second classifier is trained to classify network attack types (DDoS, portscan, Bot, web attack).

We use 70% of the data for training and 30% for model evaluation.

Binary classification: Table 5 and Table 6 describe the experimental results of the models with the Friday and Tuesday&Wednesday datasets in CIC-IDS-2017. The results show that basic machine learning algorithms perform better regarding classification effectiveness on the Friday scenarios (Friday-WorkingHours-Afternoon-DDoS, Friday-WorkingHours-Afternoon-PortScan, Friday-WorkingHours-Morning) and Tuesday&Wednesday (Tuesday-WorkingHours, Wednesday workingHours) datasets. The KNN algorithm performs the highest in most evaluation criteria on the CIC-IDS-2017 dataset, especially when combined with Autoencoder.

TABLE 5. ATTACK DETECTION RESULTS WITH FRIDAY DATASET

Method	**ML (%)**				**AE_ML(%)**			
	acc	π	*Recall*	*F1*	*acc*	π	*Recall*	*F1*
DT	0.99986	0.99983	0.99986	0.99985	99.505	0.998	0.989	0.993
RF	0.99989	0.99999	0.99976	0.99987	99.24	0.9977	0.9837	0.9907
Naive Bayes	0.60378	0.53439	0.53439	0.69468	81.4493	0.7493	0.825	0.7855
K_NN	0.98969	0.98969	0.99196	0.98869	99.7833	0.9966	0.998	0.9973
MLP	0.93333	0.98423	0.86833	0.91923	98.505	0.9847	0.9788	0.9817
SVM	0.9870	0.9889	0.9690	0.9880	0.9945	0.9951	0.9812	0.9920

TABLE 6. ATTACK DETECTION RESULTS FOR TUESDAY&WEDNESDAY DATASET

Method	**ML (%)**				**AE_ML(%)**			
	acc	π	*Recall*	*F1*	*acc*	π	*Recall*	*F1*
DT	0.9860	0.9883	0.9896	0.9880	99.4625	0.9961	0.9807	0.9884
RF	0.9940	0.9990	0.9975	0.9987	99.6431	0.9983	0.9863	0.9923
Naive Bayes	0.6520	0.5945	0.5840	0.7160	90.4146	0.7948	0.7946	0.7947
K_NN	0.9969	0.9969	0.9916	0.9889	99.8776	0.9971	0.9975	0.9973
MLP	0.9535	0.9845	0.9839	0.9532	95.9210	0.9759	0.8462	0.9064
SVM	0.9855	0.9899	0.9430	0.9654	0.9946	0.9989	0.9625	0.9820

TABLE 7. ATTACK DETECTION RESULTS FOR DATASET FRIDAY

Method	(%)	Attack Type		
		DDoS	Portscan	bots
DT	acc	73.3		
	π	0.9997	0.84	0.92396
	recall	0.77473	0.00747	0.71735
	F1	0.87295	0.01481	0.80765
AE_ DT	acc	97.88		
	π	0.99844	0.99975	0.90476
	recall	0.89567	0.99841	0.50984
	F1	0.94426	0.99908	0.65217
RandomForest	acc	88,473		
	π	first	first	first
	recall	0.97952	0.51477	0.38104
	F1	0.98966	0.67966	0.55181
AE_ RandomForest	acc	99.56		
	π	0.99968	0.99998	0.97484
	recall	0.98472	0.99881	0.55456
	F1	0.99215	0.99939	0.70696
Naive Bayes	acc	47.4		
	π	0.38162	0.97206	0.00806
	recall	0.95312	0.50252	0.73703
	F1	0.54502	0.66254	0.01595
AE_NaiveBayes	acc	83.95		
	π	0.70892	0.90683	0.01007
	recall	0.98571	0.98234	0.2254
	F1	0.82471	0.94308	0.01927
kNN	acc	99.1		
	π	0.99714	0.98234	0.62949
	recall	0.99797	0.99282	0.59571
	F1	0.99756	0.98755	0.61213
AE_kNN	acc	99.8		
	π	0.99803	0.99946	0.85526
	recall	0.99945	0.99948	0.69767
	F1	0.99874	0.99947	0.76847
MLP	acc	96.49		
	π	0.97097	0.97643	0.07533
	recall	0.99181	0.94855	0.46869
	F1	0.98128	0.96229	0.1298
AE_MLP	acc	98.65		
	π	0.99431	0.97772	0
	recall	0.97204	0.99483	0
	F1	0.98305	0.9862	0

2. Classification of network attacks

The results from Table 9 show that the proposed model performs better with higher classification efficiency for classifying network attack types than conventional machine learning algorithms, including AE_DT, AE_Naive Bayes, and AE_RandomForest with performance metrics such as acc, F1, recall, precision mostly higher.

TABLE 8. ATTACK DETECTION RESULTS FOR DATASET TUESDAY&WEDNESDAY

Method	**(%)**	**Attack Type**		
		Dos	**SSH/FTP-Patator**	**Heartbleed**
DT	acc	99.93		
	π	0.99901	0.9983	0.83333
	recall	0.99902	0.99879	1
	F1	0.99902	0.99854	0.90909
AE_ DT	acc	99.37		
	π	0.99734	0.96945	0.83333
	recall	0.97655	0.99393	1
	F1	0.98683	0.98153	0.90909
RandomForest	acc	99.88		
	π	0.9986	1	1
	recall	0.9986	0.99854	1
	F1	0.99864	0.99927	1
AE_ RandomForest	acc	99.58		
	π	0.99864	0.9983	1
	recall	0.98273	0.99563	0.8
	F1	0.99062	0.99696	0.88889
Naive Bayes	acc	54.34		
	π	0.81098	0.02818	1
	recall	0.67174	0.99757	0.8
	F1	0.73482	0.05481	0.8888
AE_NaiveBayes	acc	81.96		
	π	0.89754	0.10745	0.66667
	recall	0.59893	0.79359	0.8
	F1	0.71844	0.18927	0.72727
kNN	acc	99.49		
	π	0.98654	0.97239	1
	recall	0.99245	0.98349	0.8
	F1	0.98949	0.97791	0.88889
AE_kNN	acc	99.88		
	π	0.99755	0.9966	1
	recall	0.99758	0.99733	0.8
	F1	0.99757	0.99697	0.88889
MLP	acc	96		
	π	0.89594	0.94714	0
	recall	**0.93281**	0.6831	0
	F1	**0.91401**	0.79374	0
AE_MLP	acc	96.98		
	π	0.91399	0.9666	1
	recall	0.98419	0.50607	0.6
	F1	0.94779	0.66433	0.75

3. Experiments with the CSE-CIC-IDS-2018 dataset

For this dataset, we experimented with binary classification. The dataset consists of 10 scenarios corresponding to 10 days of data collection in 2018, with a size of nearly 6.5 GB. Due to computational limitations, we selected 03 data scenarios with a smaller size (about 300MB per file) for the experiments. The tables below describe the experimental results of traditional classification algorithm models and compare with the combination of using Autoencoder to evaluate based on the metrics of the three selected files.

TABLE 9. TEST RESULTS FOR DATA SET 02-14-2018

Method	**ML (%)**				**AE_ML(%)**			
	acc	*π*	*Recall*	*F1*	*acc*	*π*	*Recall*	*F1*
DT	0.9785	0.9695	0.9770	0.9760	0.9820	0.9855	0.9840	0.9865
RF	0.9882	0.9830	0.9820	0.9850	**0.9940**	0.9920	0.9950	0.9930
Naive Bayes	0.7252	0.7154	0.6952	0.7030	0.8450	0.7520	0.7190	0.7650
K_NN	0.9730	0.9782	0.9900	0.9876	0.9885	0.9825	0.9620	0.9890
MLP	0.9791	0. 9952	0.9989	0.9890	0.9850	**0.9960**	**0.9999**	**0.9889**
SVM	0.9680	0.9325	0.9655	0.9570	0.9845	0.9510	0.9750	0.9640

TABLE 10. TEST RESULTS FOR DATASET 02-16-2018

Method	**ML (%)**				**AE_ML(%)**			
	acc	*π*	*Recall*	*F1*	*acc*	*π*	*Recall*	*F1*
DT	0.9889	0.9794	0.9982	0.9880	0.9920	0.9950	0.9990	0.9948
RF	0.9985	0.9793	0.9882	0.9491	0.9990	0.9895	0.9996	0.9781
Naive Bayes	0.8192	0.9657	0.9450	0.9739	0.8960	0.9850	0.9890	0.9867
K_NN	0.9920	0.9780	0.9891	0.9562	0.9980	0.9840	0.9960	0.9789
MLP	0.9991	0.9850	0.9889	0.9879	**0.9998**	**0.9920**	**0.9980**	**0.9950**
SVM	0.9889	0.9210	0.9785	0.9474	0.9920	0.9635	0.9850	0.9660

TABLE 11. TEST RESULTS FOR DATASET 03-02-2018

Method	**ML (%)**				**AE_ML(%)**			
	acc	*π*	*Recall*	*F1*	*acc*	*π*	*Recall*	*F1*
DT	0.9989	0.9994	0.9986	0.9989	99.993	0.9999	0.9999	0.9998
RF	0.9983	0.9992	0.9988	0.9992	**99.997**	**1**	**0.999**	**0.9999**
Naive Bayes	0.75804	0.9929	0.9976	0.9953	87.065	0.6822	0.9844	0.8059
K_NN	0.9973	0.9980	0.9988	0.9984	99.995	0.999	0.9999	0.9999
MLP	0.99894	0.99656	0.99899	0.99788	99.91	0.9971	0.9996	0.9983
SVM	0.9980	0.9000	0.9988	0.9994	0.9972	0.9250	0.9860	0.9890

TABLE 12. TIME COMPARISON OF THE MODELS IN THE NORMAL ENVIRONMENT WITH THE PROPOSED SYSTEM

	Basic classification		**AE+ Subclass**		**Time delay (s)**
Method	**Normal System (s)**	**Proposed platform (s)**	**Normal System (s)**	**Proposed platform (s)**	
DT	64.58	49.66	100.42	76.54	14-23
RF	782.68	622.19	1437.5	1156.5	160-281
Naive Bayes	5.65	4.1796	3.1255	2.1	1-1.5
K NN	1005.4	608.23	48.39	15.65	32-400
MLP	254.04	128.16	102.31	51.25	51-125
SVM	420.55	261.12	210.5	89.77	120-160

The test results show that our 2-stage algorithm performs better for all machine learning models. MLP combined with Autoencoder model gives the best results in the data set 02-14-2018 and the set 02-16-2018. In the data set 03-02-2018, the MLP, DT, and RF models have nearly equivalent results in the evaluated measures. The test results show that the data set is very separable, so the obtained measures are very high. Therefore, applying machine learning models in attack detection in practice is possible. However, the model must have a fast processing time for application in the online stream processing system. In the next experiment, we will evaluate the performance of the algorithms on the standard computer and the proposed online stream processing platform.

D. Evaluation of system performance

We use 06 Ubuntu VMs on the Oracle VM VirtualBox tool to test the performance of the time between the proposed system and the implementation on single computers.

In the data processing layer, we design 03 virtual machines (01 machine to install and run classification algorithms on Spark, 02 machines to connect to share computing resources on Bigdata system) to perform the evaluation. The dataset used is the CIC-IDS-2017 set.

The results show that the data processed on the proposed system for processing big data on the online stream platform has a significantly lower time than in the normal environment in all the experimental algorithms.

V. Conclusion

In this paper, we have proposed a platform for network attack detection and classification with a novel algorithm combined with online stream data processing technology based on Kafka and Spark platforms. The test results showed that the proposed system has a high network attack detection rate while ensuring lower processing time than centralized systems. Most proposed models provided higher data classification results, showing that learning data features with AE are effective. However, the classification performance results are equivalent for small datasets with lower dimensions. In the future works, we will evaluate the quality of the model's attack classification and propose advanced algorithms to enhance the ability to detect attacks to serve network security monitoring in practice.

References

[1] A. Chidukwani, S. Zander and P. Koutsakis, "A Survey on the Cyber Security of Small-to-Medium Businesses: Challenges, Research Focus and Recommendations," in IEEE Access, vol. 10, pp. 85701-85719, 2022, doi: 10.1109/ACCESS.2022.3197899.

[2] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi and R. Ahmad, "Machine Learning and Deep Learning Approaches for CyberSecurity: A Review," in IEEE Access, vol. 10, pp. 19572-19585, 2022, doi: 10.1109/ACCESS.2022.3151248.

[3] R. P. Krupani, M. G. Aditya, C. S. Prithvi Raghavan and H. S. Gururaja, "Big Data Cybersecurity Monitoring System using Machine Learning," 2021 International Conference on Forensics, Analytics, Big Data, Security (FABS), Bengaluru, India, 2021, pp. 1-7, doi: 10.1109/FABS52071.2021.9702637.

[4] Clay. P. - "A modern threat response framework", Network Security, v.2015, n. 4, pp. 5(October 2015).

[5] IBM report - "Cost of a data breach 2022" - https://www.ibm.com/reports/data-breach - Last Visited: 7/3/2023.

[6] COLUCCIO, R., Ghidini, G., REALE, A., et al. "Online stream processing of machine-to-machine communications traffic: A platform comparison". In: IEEE Symposium on Computers and Communication (ISCC), pp. 1{7, 6 2014. doi: 10.1109/ISCC.2014.6912528.

[7] Hu P., Li, H., Fu, H., et al. "Dynamic defense strategy against advanced persistent threat with insiders." In: 2015 IEEE Conference on Computer Communications (INFOCOM), pp. 747.

[8] Paxson V. "Bro: a system for detecting network intruders in real-time", Computer Networks, v. 31, n. 23-24, pp. 2435.

[9] Bar A., Finamore A., Casas, P., et al. "Large-scale network traffic monitoring with DBStream, a system for rolling big data analysis." In: 2014 IEEE International Conference on Big Data (Big Data), pp. 165{170. IEEE, 10 2014. ISBN: 978-1- 4799-5666-1. doi: 10.1109/BigData.2014.7004227.

[10] Stonebraker M.,Etintemel U., Zdonik, S. "The 8 requirements of real-time stream processing", ACM SIGMOD Record, v. 34, n. 4, pp. 42{ 47, 12 2005. ISSN: 01635808. doi : 10.1145/1107499.1107504.

[11] S plunk Buys Another Startup, Launches Mission Control - https://www.sdxcentral.com/articles/news/splunk-buys-another-startup-launches-mission-control/2019/10/ - Last Accessed 6/7 2020.

[12] Dimensionality reduction for machine learning based iot botnet detection, H. Bahsi, S. Nomm, and FBL Torre, in 15th International Conference on Control, Automation, Robotics and Vision, ICARCV 2018, Singapore, November 18-21 , 2018, pp . 1857-1862.

[13] Unsupervised anomaly based botnet detection in iot networks, S. Nomm and H. Bahsi, in 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018, Orlando, FL, USA, December 17-20, 2018, 2018, pp. 1048–1053.

[14] Nguyen Viet Hung, Nguyen Van Quan, Le Thi Trang Linh, Shone Nathan, (2018), "Using Deep Learning Model for Network Scanning Detection", ICFET '18: Proceedings of the 4th International Conference on Frontiers of Educational Technologies , [117-121].

[15] Huong, P. V., Van, L. T. H., & Nguyen, P. S. (2021). Detecting Web Attacks Based on Clustering Algorithm and Multi-branch CNN. Journal of Science and Technology on Information Security, 2(12), 31-37. https://doi.org/10.54654/isj.v2i12.120.

[16] Adnan Mohsin Abdulazeez, (2021), "Classification Based on Decision Tree Algorithm for Machine Learning", Journal of Applied Science and Technology Trends.

[17] Dianne SV Medeiros; Helio N. Cunha Neto; Martin Andreoni Lopez, (2020), "A survey on

data analysis on large-Scale wireless networks: online stream processing, trends, and challenges", Journal of Internet Services and Applications.

[18] Haruna Isah; Tariq Abughofa; Sazia Mahfuz; Dharmitha Ajerla; Farhana Zulkernine; Shahzad Khan, (2019), "A Survey of Distributed Data Stream Processing Frameworks", IEEE Access, 7, 154300 – 154316.

[19] Meijuan Gao, Jingwen Tian, Mingping Xia, Intrusion Detection Method Based on Classify Support Vector Machine , secondInternational Conference on Intelligent Computation Technology and Automation, ICICTA '09, vol. 2, pp. 391-394.

[20] BHUYAN, Monowar H.; BHATTACHARYYA, Dhruba K.; KALITA, Jugal K, AOCD: An Adaptive Outlier Based Coordinated Scan Detection Approach , IJ Network Security, 2012, 14.6: 339-351.

[21] Chao Wang, Bailing Wang, Hongri Liu, Haikuo, Anomaly Detection for Industrial Control System Based on Autoencoder Neural Network, 2020, Wireless Communications and Mobile Computing.

[22] Dung, N. T., Quân, N. V., & Hùng, N. V. (2023). Application of deep learning model in network reconnaissance attack detection. Journal of Science and Technology on Information Security, 2(16), 60-72. https://doi.org/10.54654/isj.v1i16.922.

[23] Xavier Glorot, Yoshua Bengio, (2010), "Understanding the difficulty of training deep feedforward neural networks", Journal of Machine Learning Research 9, [249-256].

ABOUT THE AUTHORS

Ngo Thanh Tung

Workplace: Faculty of information technology, Le Quy Don Technical University.

Email: tungmta@gmail.com

Education: Master student in Information System Security specification, Le Quy Don technical university.

Recent research detection: Information security; Intrusion detection.

Cơ quan làm việc: Khoa Công nghệ thông tin, Trường Đại học Kỹ thuật Lê Quý Đôn.

Email: tungmta@gmail.com

Quá trình đào tạo: Thạc sĩ chuyên ngành An toàn hệ thống thông tin, trường Đại học Kỹ thuật Lê Quý Đôn.

Hướng nghiên cứu hiện nay: Bảo mật thông tin; Phát hiện xâm nhập.

Dang Thi Mai

Workplace: Faculty of basic science, University of transport and communication.

Email: maidt@utc.edu.vn

Education: She received her BSc, MSc in Applied Mathematics and Informatics from Hanoi university of Natural Science, PhD degrees in Applied Mathematics from Moscow Institute of Physics and Technology in 2012.

Recent research detection: Applied Mathematics; Machine learning.

Cơ quan làm việc: Khoa Khoa học cơ bản, Trường Đại học Giao thông Vận tải.

Email: mait@utc.edu.vn

Quá trình đào tạo: Nhận bằng Cử nhân, Thạc sĩ Toán Ứng dụng và Tin học tại trường Đại học Khoa học Tự nhiên Hà Nội, bằng Tiến sĩ Toán Ứng dụng tại Viện Vật lý và Công nghệ Moscow năm 2012.

Hướng nghiên cứu hiện nay: Toán ứng dụng; Học máy.

Nguyen Viet Hung

Workplace: Faculty of information technology, Le Quy Don Technical University.

Email: hungnv@lqdtu.edu.vn

Education: He received his BSc, MSc and PhD degrees in Computer Science from Moscow Institute of Physics and Technology in 2006, 2008 and 2012 respectively.

Recent research detection: Information security; Malware detection; Intrusion detection.

Cơ quan làm việc: Khoa Công nghệ thông tin, Trường Đại học Kỹ thuật Lê Quý Đôn.

Email: hungnv@lqdtu.edu.vn

Quá trình đào tạo: Nhận bằng Cử nhân, Thạc sĩ và Tiến sĩ về Khoa học Máy tính tại Viện Vật lý và Công nghệ Moscow vào các năm 2006, 2008 và 2012.

Hướng nghiên cứu hiện nay: Bảo mật thông tin; Phát hiện phần mềm độc hại; Phát hiện xâm nhập.