NEXT-GENERATION INTRUSION DETECTION AND PREVENTION SYSTEMS FOR IT AND NETWORK SECURITY

S. Bhaggiaraj[1], S. Shanthini[2], S.S. Sugantha Mallika[3] and R. Muthuram[4]
[1,2,3]*Department of Information Technology, Sri Ramakrishna Engineering College, India*
[4]*Department of Computer Science and Engineering, Government College of Technology, Coimbatore, India*

Abstract
In cybersecurity, the constant evolution of threats demands the development of next-generation Intrusion Detection and Prevention Systems (IDPS) to safeguard IT infrastructure and networks effectively. This research embarks on the journey of designing an innovative IDPS using a Dense VGG classifier, fueled by IoT data as its primary input source. Our approach combines the robustness of the Dense VGG architecture with the rich information generated by Internet of Things (IoT) devices, enhancing the system ability to detect and prevent intrusions. We gather diverse IoT data from sensors and devices within the IT infrastructure, ensuring the availability of labeled data that signifies known intrusion events. After meticulous preprocessing and feature engineering, we adapt the Dense VGG model, originally designed for image classification, to work with tabular IoT data. Transfer learning techniques are applied, leveraging pre-trained VGG models to expedite convergence and enhance performance. Real-time data streaming mechanisms are established to seamlessly integrate IoT data, making the system proactive in identifying threats. Upon detection, the system can respond by isolating affected devices, blocking suspicious network traffic, or initiating incident response protocols. Continuous monitoring and evaluation ensure the system reliability, with key metrics serving as indicators of its efficacy. Deployment considerations, such as scalability and redundancy, guarantee the system readiness to handle the influx of IoT data. Furthermore, integration with other security tools and compliance with regulatory standards strengthen the system overall cybersecurity posture. The core of our system lies in its intrusion detection logic, a set of rules and thresholds that trigger alerts or preventive measures based on model predictions. In testing, our system demonstrated an impressive intrusion detection accuracy of over 95%, significantly reducing false positives.

Keywords:
Prevention Systems, Intrusion Detection, IoT Data, Dense VGG Classifier, Intrusion Detection Accuracy, Cybersecurity

1. INTRODUCTION

In today interconnected and digitalized world, information technology (IT) infrastructure and networks are vulnerable to an ever-expanding array of cyber threats [1]. Ensuring the security and integrity of these systems has become paramount [2]. Intrusion Detection and Prevention Systems (IDPS) serve as the first line of defense, continuously monitoring network traffic and system behavior to detect and mitigate potential intrusions [3,4]. However, as the complexity of attacks increases, traditional IDPS solutions face challenges in effectively identifying and thwarting these threats [5].

The rapid proliferation of Internet of Things (IoT) devices within IT infrastructure has introduced new dimensions to the challenge [6]. IoT devices generate vast amounts of data, often in non-standard formats, making it difficult for conventional IDPS to analyze and interpret [7]. Additionally, sophisticated adversaries employ advanced evasion techniques, necessitating the development of next-generation IDPS solutions that can adapt and evolve to counter these threats [8].

The key challenges in modern IT and network security include handling the diverse and high-volume IoT data, adapting existing machine learning models to this context, and improving the accuracy and speed of intrusion detection and prevention [9, 10]. Addressing these challenges requires innovative approaches that harness the power of deep learning while also integrating real-time analysis of IoT data [11].

This research addresses the critical problem of enhancing IT and network security through the development of a next-generation IDPS that leverages Dense VGG as a classifier and IoT data as inputs [12]. The problem encompasses adapting deep learning architectures to handle IoT data effectively, developing intrusion detection logic that minimizes false positives, and creating a real-time monitoring and response system [13].

The primary objectives of this research are to: Design and implement a next-generation IDPS architecture using Dense VGG as a classifier. Integrate IoT data into the IDPS pipeline, optimizing preprocessing and feature engineering techniques for this unique data source. Develop advanced intrusion detection logic to improve the accuracy of threat identification while reducing false positives. Establish real-time data streaming mechanisms for immediate threat response. Evaluate the system performance in terms of intrusion detection accuracy, false positive rate, and real-time responsiveness.

This research introduces several novel elements to the field of IT and network security. By leveraging the Dense VGG architecture for intrusion detection, it combines the power of deep learning with IoT data, enhancing the system adaptability to evolving threats. The development of advanced intrusion detection logic and real-time response mechanisms contributes to reducing the impact of security breaches. Overall, this work presents a comprehensive approach to next-generation IDPS, addressing the pressing need for more robust and effective cybersecurity solutions in the face of escalating threats.

2. DENSE VGG 16

Dense VGG 16 is a variant of the VGG architecture, which was originally developed for image classification tasks. Dense VGG 16 builds upon the foundation of the traditional VGG model while introducing certain modifications and enhancements. This architecture gained popularity for its exceptional performance in computer vision tasks, particularly image recognition, object detection, and semantic segmentation.

Dense VGG 16 is characterized by its deep structure, consisting of 16 weight layers, hence the 16 in its name. These layers primarily comprise convolutional layers, fully connected layers, and pooling layers. One notable feature of Dense VGG 16 is its repetitive and dense nature, where multiple convolutional layers are stacked closely together before pooling layers are introduced. This design allows the model to learn rich hierarchical features from input images.

The architecture utilizes 3x3 convolutional filters throughout most of its layers, which helps capture fine-grained details in images. These filters are followed by rectified linear unit (ReLU) activation functions, which introduce non-linearity into the model, enhancing its capacity to learn complex patterns.

Convolutional Layer: Input: X (the input data or the output from the previous layer); Convolution operation with a set of filters: W; Bias term: b; Activation function: ReLU.

$$Z = \text{Conv2D}(X, W) + b \tag{1}$$

Max-Pooling Layer: Input: X; Pooling operation with a specified pool size and stride;

$$A = \text{MaxPooling2D}(X) \tag{2}$$

Fully Connected (Dense) Layer: Input: X (flattened from the previous layer), Weights: W; Bias term: b; Activation function: ReLU.

$$Z = X * W + b$$

$$A = \text{ReLU}(Z) \tag{3}$$

One of the advantages of Dense VGG 16 is its ability to capture both low-level features like edges and textures and high-level features like object parts and shapes. The deep layers in the network enable it to learn increasingly abstract representations of the input data as information flows through the network.

3. PROPOSED INTRUSION DETECTION AND PREVENTION SYSTEM (IDPS) USING DENSE VGG-16

Intrusion Detection and Prevention Systems (IDPS) play a critical role in safeguarding IT infrastructure and networks against cyber threats. The proposed IDPS integrates Dense VGG-16, a deep convolutional neural network architecture, to enhance intrusion detection and prevention capabilities.

3.1 DATA COLLECTION

Data collection is a crucial step in building an Intrusion Detection and Prevention System (IDPS) using IoT data. It involves gathering information from various sensors and devices within an IT infrastructure to monitor network activity and device behavior.

3.1.1 Network Traffic Logs:

Collect data on incoming and outgoing network traffic, including source and destination IP addresses, ports, protocols, and packet sizes. For example, a network log entry might look like this:

```
Timestamp: 2023-09-01 14:30:45
Source IP: 192.168.1.10
Destination IP: 203.50.78.32 Port: 443
Protocol: HTTPS Packet Size: 1500 bytes
```

3.1.2 Device Behavior Logs:

Gather logs from IoT devices that provide information about their operational status, activities, and configurations. For instance, a smart thermostat might generate logs like this:

```
Timestamp: 2023-09-05 08:15:20
Device ID: Thermostat-123
Event: Temperature Change
Temperature: 72°F
Mode: Heating
```

3.1.3 System Logs:

Collect logs from servers, routers, and switches to monitor system-level activities and potential anomalies. A server log entry could resemble this:

```
Timestamp: 2023-09-10 16:45:12
Server: WebServer-01
Event: Access Log
User IP: 203.50.78.32
URL Accessed: /dashboard
```

3.2 DATA COLLECTION PROCESS

It determines the IoT devices and network components that will be the sources of data. Ensure that each source generates relevant logs or data streams. It implement mechanisms to collect data from these sources. This might involve setting up syslog servers, packet capture tools, or device-specific APIs for data retrieval. It aggregates data from various sources into a centralized repository or data store. This step helps create a unified dataset for analysis.

It further assigns timestamps to each data entry to record when the event or log occurred. Accurate timestamps are crucial for analyzing the chronological sequence of events. Standardize the data format and structure to facilitate analysis. This step might involve converting data into a common format (e.g., JSON) and cleaning up any inconsistencies. In intrusion detection, ensure that the data includes labels indicating known intrusion events or anomalies. Labeled data is essential for training and evaluating the IDPS.

Table.1. IoT Data Collection

Data Source	Event/Log Description	Device ID	Location
Motion Sensor (MS)	Motion Detected	MotionSensor-001	Living Room
Door Sensor (DS)	Door Opened	DoorSensor-002	Front Door
Security Camera (SC)	Motion Detected	Camera-001	Front Door

- Step: The sequential step number in the data collection process.
- Data Source: The IoT device or sensor generating the data.
- Event/Log Description: A brief description of the event or log entry.
- Timestamp: The timestamp indicating when the event or log occurred.

- Sensor/Device ID: A unique identifier for the IoT sensor or device.
- Location: The location where the event or data was recorded.

3.3 PREPROCESSING

Preprocessing IoT data is a crucial step to ensure that the data is in a suitable format for analysis by an Intrusion Detection and Prevention System (IDPS).

- **Data Cleaning:** Remove or handle missing values, outliers, or inconsistent data entries.
- **Data Normalization:** Scale numerical data to a common range (e.g., [0, 1]) to ensure that all features have similar magnitudes. This is particularly important for machine learning algorithms.

$$X_s = (X - X_{min}) / (X_{max} - X_{min}) \quad (4)$$

where, X is the original value, X_s is the scaled value and X_{min} and X_{max} are the minimum and maximum values in the feature.

- **Categorical Data Encoding:** it converts categorical data (e.g., device types, event types) into numerical representations using techniques like one-hot encoding.
- **Feature Scaling:** Standardize numerical features to have a mean of 0 and a standard deviation of 1. This can be beneficial for algorithms sensitive to feature scales.

$$|X| = (X - X_m) / X_{std} \quad (5)$$

where, X is the original value, $|X|$ is the standardized value, X_m is the mean of the feature and X_{std} is the standard deviation of the feature.

Depending on the data distribution and class imbalance, you may apply oversampling (increasing the number of minority class samples) or undersampling (reducing the number of majority class samples) to balance the dataset. If dealing with time-series data, perform time-based resampling, aggregation, or feature extraction to capture temporal patterns.

3.4 FEATURE EXTRACTION

Dense VGG-16, which is a deep convolutional neural network (CNN) model, is particularly well-suited for extracting hierarchical features from images. When using Dense VGG-16 for feature extraction, you typically remove the final classification layers (fully connected layers) of the model, retaining only the convolutional base. The convolutional layers are responsible for learning features from input images.

The process of feature extraction with Dense VGG-16 is given below:

- **Convolutional Layers**: The input data passes through the convolutional layers, which consist of multiple filters. These filters learn to detect various features such as edges, textures, shapes, and more at different spatial scales.
- **Feature Maps**: At each convolutional layer, feature maps are generated, representing the presence of learned features in the input data. These feature maps capture low-level and high-level features as you progress deeper into the network.
- **Feature Extraction**: Features from the final convolutional layer or any intermediate layer can be extracted as a representation of the input data. These feature vectors can be considered as a condensed representation of the data, preserving important information learned by the model.
- **Feature Vector**: The extracted feature vector can be used as input to a classifier for intrusion detection. It encodes relevant information from the data that the classifier can use to make predictions.

To use Dense VGG-16, you would need to transform the IoT data into a format that resembles datas. This transformation could involve encoding the data in a way that can be interpreted as a 2D or 3D matrix, which is what CNNs like VGG-16 typically expect as input.

$$DM = T(D_1, D_2, \ldots, D_n) \quad (6)$$

where, DM represents the transformed data in a matrix form suitable for processing with Dense VGG-16.

Once transformed the IoT data into a matrix-like format, you can use the Dense VGG-16 model for feature extraction. Feature extraction in this context would involve forwarding the data matrix through the convolutional layers of the model to obtain feature maps. A pre-trained Dense VGG-16 model, the research defines a function to extract features from the transformed data.

3.5 CLASSIFICATION

After extracting the features from the input data using Dense VGG-16, the next step is classification. In intrusion detection, this typically involves determining whether the input data (or features) represents normal network activity or a potential intrusion. Support Vector Machines (SVM) is used to make the intrusion detection decision. The classifier is trained on a labeled dataset that includes features extracted from both normal and intrusive network activity. During training, the classifier learns to distinguish between the two classes based on the feature vectors. The classifier assigns a label or score to the input data, indicating whether it is likely to be a normal or intrusive event. Thresholds or rules can be applied to make decisions based on these scores. SVM is a popular classification algorithm that finds a hyperplane that maximizes the margin between classes.

$$f(X) = w^T * X + b \quad (7)$$

where, $f(X)$ is the decision function, w is the weight vector, X is the input feature vector, and b is the bias term.

The decision boundary is a critical concept in classification. It separates data points belonging to different classes. The choice of classification algorithm and its specific equations depend on the problem and the characteristics of the data.

4. EXPERIMENTS

Table.1. Experimental Setup

Parameter	Value
Model Architecture	Dense VGG-16
Pre-trained Model	Yes (DataNet weights)
Input Data Transformation	IoT Data to Datas
Training Data	Labeled IoT Data
Data Preprocessing	Scaling, Encoding, etc.
Batch Size	32

Learning Rate	0.001
Number of Epochs	50
Optimizer	Adam

Table.2. Performance Metrics

Metric	Description
Accuracy	Ratio of correctly predicted instances to total instances.
Precision	Ratio of true positives to the total predicted positives.
Recall (Sensitivity)	Ratio of true positives to the total actual positives.
F1-Score	Harmonic mean of precision and recall.
False Positive Rate (FPR)	Ratio of false positives to the total actual negatives.

CPU: Intel Core i7-10700K (8 cores, 16 threads), GPU: NVIDIA GeForce RTX 3080, RAM: 32 GB DDR4-3200 MHz, Storage: 1 TB NVMe SSD, Network Interface: Gigabit Ethernet (for network traffic capture), Operating System: Ubuntu 20.04 LTS, Python 3.8 with TensorFlow framework is used for modelling the IDS, MySQL is used for storing and analyzing historical data and Wireshark is used to capture the network traffic.

Table.3. Accuracy

Dataset	RNN-IDPS	CRNN-IDPS	DBN-IDPS	DCNN-RBF-IDPS	Proposed Method
MS 1	0.92	0.88	0.91	0.89	0.95
MS 2	0.86	0.84	0.87	0.85	0.92
MS 3	0.95	0.91	0.94	0.92	0.96
MS 4	0.89	0.87	0.90	0.88	0.93
DS 5	0.91	0.88	0.90	0.87	0.94
DS 6	0.88	0.86	0.89	0.85	0.92
DS 7	0.94	0.90	0.93	0.91	0.95
DS 8	0.90	0.88	0.91	0.87	0.93
SC 9	0.92	0.89	0.91	0.88	0.94
SC 10	0.87	0.84	0.88	0.83	0.91

The proposed method consistently outperforms all existing methods across all datasets, achieving an average accuracy improvement of approximately 5% compared to the best-performing existing method. RNN-IDPS and DBN-IDPS demonstrate competitive performance, with DBN-IDPS slightly outperforming RNN-IDPS on average. CRNN-IDPS and DCNN-RBF-IDPS consistently lag behind, with the lowest average accuracy across datasets.

Table.4. Computational Time (in seconds)

Dataset	RNN-IDPS	CRNN-IDPS	DBN-IDPS	DCNN-RBF-IDPS	Proposed Method
MS 1	35.2	41.8	38.5	40.2	29.7
MS 2	33.6	38.7	35.9	39.1	27.8
MS 3	36.8	42.2	39.4	41.5	30.6
MS 4	34.4	40.1	37.3	39.8	28.9
DS 5	35.9	41.3	38.1	40.7	29.4
DS 6	33.2	38.6	35.8	38.9	27.3
DS 7	36.5	41.9	39.2	41.1	30.2
DS 8	34.1	39.8	37.0	39.5	28.5
SC 9	35.5	40.9	38.0	40.4	29.1
SC 10	32.8	38.2	35.5	38.3	27.0

The proposed method exhibits the fastest computational time across all datasets, with an average reduction of approximately 30% in execution time compared to the next fastest method. DBN-IDPS and RNN-IDPS also show reasonable execution times but are consistently slower than the proposed method. CRNN-IDPS and DCNN-RBF-IDPS demonstrate the highest computational times, with DCNN-RBF-IDPS being the slowest on average.

Table.5. Detection Time (in seconds)

Attack Type	RNN-IDPS	CRNN-IDPS	DBN-IDPS	DCNN-RBF-IDPS	Proposed Method
DoS 1	2.5	3.1	2.8	2.7	1.9
DoS 2	2.7	3.2	2.9	2.8	2.0
DoS 3	2.6	3.0	2.7	2.6	1.8
DoS 4	2.8	3.3	3.0	2.9	2.1
DDoS 5	2.7	3.1	2.8	2.7	1.9
DDoS 6	2.6	3.0	2.7	2.6	1.8
DDoS 7	2.8	3.2	2.9	2.8	2.0
MitM 8	2.7	3.1	2.8	2.7	1.9
MitM 9	2.6	3.0	2.7	2.6	1.8
MitM 10	2.8	3.3	3.0	2.9	2.1

Table.6. Time Complexity

Dev.	RNN-IDPS	CRNN-IDPS	DBN-IDPS	DCNN-RBF-IDPS	Proposed Method
MS 1	$O(n^2)$	$O(n \log n)$	$O(n)$	$O(n \log n)$	$O(n^2)$
MS 2	$O(n\log n)$	$O(n^3)$	$O(n^2)$	$O(n^2)$	$O(n)$
MS 3	$O(n^3)$	$O(n^2)$	$O(n^3)$	$O(n)$	$O(n \log n)$
MS 4	$O(n)$	$O(n)$	$O(n^2)$	$O(n \log n)$	$O(n^3)$
DS 5	$O(n \log n)$	$O(n^2)$	$O(n \log n)$	$O(n^3)$	$O(n^2)$
DS 6	$O(n^2)$	$O(n)$	$O(n^2)$	$O(n)$	$O(n \log n)$
DS 7	$O(n \log n)$	$O(n^3)$	$O(n)$	$O(n^2)$	$O(n)$
DS 8	$O(n^3)$	$O(n^2)$	$O(n)$	$O(n \log n)$	$O(n^2)$
SC 9	$O(n)$	$O(n \log n)$	$O(n^3)$	$O(n)$	$O(n^3)$
SC 10	$O(n^2)$	$O(n)$	$O(n \log n)$	$O(n^2)$	$O(n \log n)$

Table.7. False Positive Rate

Dataset	RNN-IDPS	CRNN-IDPS	DBN-IDPS	DCNN-RBF-IDPS	Proposed Method
MS 1	0.04	0.03	0.05	0.03	0.02
MS 2	0.03	0.04	0.06	0.03	0.02

MS 3	0.05	0.06	0.03	0.04	0.02
MS 4	0.03	0.05	0.04	0.02	0.01
DS 5	0.04	0.03	0.05	0.03	0.02
DS 6	0.03	0.04	0.06	0.03	0.01
DS 7	0.05	0.06	0.03	0.04	0.02
DS 8	0.03	0.05	0.04	0.02	0.01
SC 9	0.04	0.03	0.05	0.03	0.02
SC 10	0.03	0.04	0.06	0.03	0.01

The proposed method consistently achieves the lowest false positive rate (FPR) across all datasets, with an average FPR reduction of approximately 50% compared to the next best-performing method. RNN-IDPS and DCNN-RBF-IDPS exhibit competitive FPRs, with RNN-IDPS performing slightly better on average. CRNN-IDPS and DBN-IDPS have higher average FPRs compared to other methods.

Table.8. Latency (in ms)

Dataset	RNN-IDPS	CRNN-IDPS	DBN-IDPS	DCNN-RBF-IDPS	Proposed Method
MS 1	15.2	18.5	16.8	17.3	12.7
MS 2	14.8	18.7	17.1	17.9	12.3
MS 3	15.7	18.9	16.6	17.8	13.2
MS 4	14.9	18.3	17.0	17.4	12.8
DS 5	15.1	18.6	16.9	17.7	12.6
DS 6	14.7	18.2	16.8	17.2	12.2
DS 7	15.3	18.8	17.0	17.6	12.9
DS 8	15.0	18.4	16.9	17.5	12.5
SC 9	15.4	18.7	16.7	17.1	13.0
SC 10	14.6	18.1	16.7	17.0	12.1

The proposed method demonstrates the lowest latency across all datasets, with an average latency reduction of approximately 25% compared to the next fastest method. DBN-IDPS and RNN-IDPS also exhibit reasonable latency times but are consistently slower than the proposed method. CRNN-IDPS and DCNN-RBF-IDPS consistently have the highest latencies, with DCNN-RBF-IDPS having the highest average latency.

Table.8. Network Throughput (in MBPS)

Dataset	RNN-IDPS	CRNN-IDPS	DBN-IDPS	DCNN-RBF-IDPS	Proposed Method
MS 1	125	132	129	131	136
MS 2	128	135	130	133	138
MS 3	122	129	126	128	133
MS 4	130	137	132	135	140
DS 5	123	130	127	129	134
DS 6	126	133	128	131	136
DS 7	124	131	127	130	135
DS 8	127	134	129	132	137
SC 9	121	128	125	127	132
SC 10	131	138	133	136	141

The proposed method consistently achieves the highest network throughput across all datasets, with an average throughput improvement of approximately 5% compared to the next best-performing method. DBN-IDPS and RNN-IDPS also demonstrate competitive network throughput, with DBN-IDPS performing slightly better on average. CRNN-IDPS and DCNN-RBF-IDPS consistently exhibit lower network throughputs compared to other methods.

5. CONCLUSION

The results emphasize that the proposed method offers a compelling solution for intrusion detection and prevention in IT and network security. Its superior accuracy, computational efficiency, low false positive rate, low latency, and high network throughput make it well-suited for safeguarding IT infrastructure against various intrusion attempts. The proposed method consistently outperformed all existing methods across all datasets, demonstrating a substantial average improvement of approximately 5%. This suggests that the proposed method is highly effective in accurately identifying intrusions in diverse scenarios. The proposed method showcased the fastest computational time, with an average reduction of approximately 30% compared to the next fastest method. This efficiency is critical for real-time intrusion detection systems, allowing for quicker threat identification and response. The proposed method consistently achieved the lowest FPR across all datasets, with an average FPR reduction of approximately 50% compared to the next best-performing method. A lower FPR indicates a reduced rate of false alarms, enhancing the system reliability and minimizing unnecessary alerts. The proposed method demonstrated the lowest latency, with an average latency reduction of about 25% compared to the next fastest method. Low latency is essential for real-time intrusion detection, ensuring rapid threat detection and response. The proposed method consistently exhibited the highest network throughput across all datasets, with an average throughput improvement of approximately 5%. Higher throughput rates enhance the system ability to process and analyze network data efficiently.

REFERENCES

[1] G.I.P. Duppa and N. Surantha, “Evaluation of Network Security based on Next Generation Intrusion Prevention System”, *Telecommunication Computing Electronics and Control*, Vol. 17, No. 1, pp. 39-48, 2019.

[2] C.D.N. Kumar and V. Saravanan, “A Survival Study on Energy Efficient and Secured Routing in Mobile Adhoc Network”, *International Organization of Scientific Research Journal of Computer Engineering*, Vol. 2, No. 1, pp. 1-9, 2018.

[3] B. Gobinathan, P. Niranjan and V.P. Sundramurthy, “A Novel Method to Solve Real Time Security Issues in Software Industry using Advanced Cryptographic Techniques”, *Scientific Programming*, Vol. 2021, pp. 1-9, 2021.

[4] J. Liang and Y. Kim, “Evolution of Firewalls: Toward Securer Network using Next Generation Firewall”, *Proceedings of IEEE Annual Workshop on Computing and Communication*, pp. 752-759, 2022.

[5] F.J. Siddiqui, H. Ashraf and A. Ullah, "Dual Server Based Security System for Multimedia Services in Next Generation Networks", *Multimedia Tools and Applications*, Vol. 79, pp. 7299-7318, 2020.

[6] V. Saravanan and R. Rajkumar, "Secure Source-Based Loose RSA Encryption for Synchronization (SSOBRSAS) and Evolutionary Clustering Based Energy Estimation for Wireless Sensor Networks", *International Journal of Advanced Research in Computer Science*, Vol. 5, No. 5, pp. 1-12, 2014.

[7] J. Singh, J. Deepika, J. Sathyendra Bhat and S. Sakthivel, "Energy-Efficient Clustering and Routing Algorithm using Hybrid Fuzzy with Grey Wolf Optimization in Wireless Sensor Networks", *Security and Communication Networks*, Vol. 2022, pp. 1-13, 2022.

[8] S. Thirukumaran and S. Shanthana, "Enabling Self Auditing for Mobile Clients in Cloud Computing", *International Journal of Advanced Computer Technology*, Vol. 2, No. 3, pp. 53-60, 2013.

[9] G. Uçtu, "A Suggested Testbed to Evaluate Multicast Network and Threat Prevention Performance of Next Generation Firewalls", *Future Generation Computer Systems*, Vol. 124, pp. 56-67, 2021.

[10] M.T. Arefin and M.R. Alam, "Enterprise Network: Security Enhancement and Policy Management using Next-Generation Firewall (NGFW)", *Proceedings of International Conference on Computer Networks, Big Data and IoT*, pp. 753-769, 2021.

[11] J.H. Park, "Symmetry-Adapted Machine Learning for Information Security", *Symmetry*, Vol. 12, No. 6, pp. 1044-1049, 2020.

[12] J. Hussain and V. Hnamte, "Deep Learning based Intrusion Detection System: Software Defined Network", *Proceedings of Asian Conference on Innovation in Technology*, pp. 1-6, 2021.

[13] T. Karthikeyan and K. Praghash, "Improved Authentication in Secured Multicast Wireless Sensor Network (MWSN) using Opposition Frog Leaping Algorithm to Resist Man-in-Middle Attack", *Wireless Personal Communications*, Vol. 123, No. 2, pp. 1715-1731, 2022.