LSTM Based Intrusion Detection for MQTT Networks in Cyber-Physical Systems

Abhishek Dand
Schulich School of Engineering
University of Calgary
Calgary, Canada
abhishek.dand@ucalgary.ca

Prof. Dr. Hadis Karimipour
Schulich School of Engineering
University of Calgary
Calgary, Canada
hadis.karimipour@ucalgary.ca

***Abstract*—Message Queuing Telemetry Transport (MQTT) is a widely used publish-subscribe protocol for communicating sensor or event data. MQTT's publish-subscribe strategy makes it more attractive to intruders, which increases the number of possible attacks. Using the MQTT IDS dataset, this project shows how deep learning and LSTM can be used to detect intrusions. The MQTT dataset contains both benign and malicious IoT network traffic that includes both TCP and MQTT protocol-level information.**

***Index Terms*—Cyber-physical Systems, MQTT, Intrusion Detection, LSTM, Deep Learning**

I. Introduction

A cyber-physical system (CPS) is an orchestration of computers and physical systems. Embedded computers monitor and control physical processes, usually with feedback loops, where physical processes affect computations and vice versa. CPS connects strongly to the currently popular terms Internet of Things (IoT), Industry 4.0, the Industrial Internet, Machine-to-Machine (M2M), the Internet of Everything, TSensors (trillion sensors), and the fog (like the cloud, but closer to the ground) [2]. For CPS simulation, a communication protocol should be established for data transmission between physical systems and the corresponding simulation models during the simulation [1]. Currently, in the market there are many communication protocols available like MQTT, COAP, HTTP, etc. The MQTT (Message Queue Telemetry Transport, MQTT) protocol has been widely used in the field of the Internet of Things due to its subscription and release mode, and has even become a de facto data transmission standard. MQTT was proposed in 1999 by Andy Stanford-Clark of IBM and Arlen Nipper of Arcom, and then became an OASIS standard in 2013 [3]. It is light, open, basic, and designed to be straightforward to apply. These qualities make it excellent for application in a variety of circumstances, including constrained environments such as machine-to-machine (M2M) and Internet of Things (IoT) contexts where a minimal code footprint is required and/or network bandwidth is limited [7].

Currently in the market, because of the necessity to adapt to demands, functionality has taken precedence over security in many IoT advancements. As a result, many early IoT protocols (such as MQTT) lacked the security safeguards to operate properly in current situations. In [6] the authors executed a query using SHODAN API to find MQTT devices running on port 1883. Port 1883 is commonly used for unsecured MQTT, which means that you can't be sure that the MQTT server you connected to is the one you meant to, and an intermediate party can eavesdrop on your MQTT communication. After the query was executed, around 53,396 devices were found running on the following specification. 60% of the servers allow unauthorized access to topics i.e., a client can publish or subscribe to any topic without the credentials. Authentication is the first line of defense, still, it is not enabled in most of the servers, which can be exploited by attackers to perform various kinds of attacks. To tackle this a lot of research is being carried out to develop an effective Intrusion Detection System. IDSs are software that monitors computer networks for malicious activities, such as stealing information, censoring, or breaking network protocols [5].

In this course project, we have trained and evaluated LSTM and Deep Learning model on MQTTIoT-IDS2020 Dataset. This paper is organized as follows: Section Section 1 - Introduction; Section 2 - Literature Review; Section 3 - Methodology; Section 4 - Experiments and Results; Section 5 - Conclusion.

II. Related Work

The security and Vulnerabilities of MQTT are being studied by many researchers across the globe. This research comprises analyzing real-world scenarios, building threat models and analyzing the impact of different attacks, and leveraging Machine Learning and Artificial Intelligence to build an effective intrusion detection system. Here, a short literature review is showcased with a focus on Machine Learning, MQTT, and Intrusion Detection systems.

In paper [12], authors have presented a deep learning-based intrusion detection system for MQTT-enabled IoT smart systems, which achieves high accuracy for detecting attacks in binary and multi-class scenarios. The proposed model surpasses conventional machine learning-based IDSs and other deep learning models. This study contributes to the current research on applying deep learning techniques for intrusion detection in IoT systems, which have become increasingly popular due to their ability to achieve high accuracy in classification tasks.

The paper [14] proposes an optimal multi-layer architecture for intrusion detection using stacked LSTM cells to achieve maximum accuracy. The proposed technique includes effective feature selection, preventing overfitting with dropout layers and batch normalization, handling categorical features using one-hot encoding, and reducing error with back-propagation. The proposed model achieves high accuracy rates for binary and multi-class classification on the NSL-KDDTest+ dataset and performs well in detecting attacks with a high ROC-AUC score. The study demonstrates the potential of deep learning analysis in cyber security and suggests future work on addressing class imbalance issues for real-world implementation and simulation. The proposed model shows robust and effective performance compared to other approaches in the field of deep learning for intrusion detection.

The study in paper [10] adds to the existing research on Intrusion Detection Systems (IDS) for Internet of Things (IoT) networks by investigating the challenges and requirements of building IDS for MQTT-based networks. The authors have simulated the MQTT attack scenario and created dataset named MQTTIDS-2020. They evaluate six different machine learning techniques as attack classifiers and demonstrate their potential for developing efficient and accurate IDS for IoT networks. These findings can be useful as a starting point for further research in this area. The study also reports the accuracy results for the machine learning techniques, with the Support Vector Machine (SVM) and Random Forest (RF) classifiers performing well in identifying MQTT-based attacks using flow-based features.

III. Methodology

In this section we will discuss about the dataset, the basic LSTM model and proposed intrusion detection model in detail.

A. MQTTIDS-2020 Dataset

The MQTT IoT IDS2020 dataset, which is available in [10], comprises recordings from five scenarios, including normal operation and four attack scenarios. The attacks involve aggressive scanning, UDP scanning, Sparta SSH brute-force, and MQTT brute-force attacks. The dataset was collected using tcpdump by recording Ethernet traffic and exporting it to pcap files. The network architecture consists of 12 MQTT sensors, a broker, a machine for camera feed simulation, and an attacker. In the normal operation scenario, all 12 sensors send random messages using the "Publish" MQTT command. The dataset provides a realistic simulation of the MQTT IoT network in a normal operation scenario and also includes generic networking scanning attacks as well as MQTT brute-force attacks. The dataset can be used by researchers to develop and evaluate IoT Intrusion Detection Systems [12]. The dataset is available in both raw capture format (.pcap files) and processed features. The features include packet-based features, unidirectional-based features, and bidirectional-based features [4].

Feature	Data type	Description
ip _src	Text	Source IP Address
ip-dest	Text	Destination IP Address
timestamp	datetime	time of the occurence
protocol	Text	Last layer protocol
ttl	Integer	Time to live
ip-len	Integer	Packet Length
ip_flag_rb	Binary	Reserved IP flag
prt—src	Integer	Source Port
prt_dst	Integer	Destination Port
proto	Integer	Transport Layer protocol (TCP /tJDP)
tcp_flag_cwr	Binary	Congestion Window Reduced TCP flag
tcp—flag-ecn	Binary	ECN Echo TCP flag
tcp_flag_urg	Binary	Urgent TCP flag
tcp—flag-ack	Binary	Acknowledgement TCP flag
tcp—flag-push	Binary	Push TCP flag
tcp_flag_reset	Binary	Reset TCP flag
tcp—flag-syn	Binary	Synchronization TCP flag
tcp_flag_fin	Binary	Finish TCP flag
num-pkts	Integer	Number of Packets in the flow
Inean_iat	Decimal	Average inter arrival time
std-iat	Decimal	Standard deviation of inter arrival time
min_iat	Decimal	Minimum inter arrival time
max-iat	Decimal	Maximum inter arrival time
num _bytes	Integer	Number of bytes
num-psh-flags	Integer	Number of push flag
num-rst-flags	Integer	Number of reset flag
num -u rg —flags	Integer	Number of urgent flag
mean-pkt -len	Decimal	Average packet length
std_pkt_len	Decimal	Standard deviation packet length
min-pkt-len	Decimal	Minimum packet length
_pkt_len	Decimal	Maximum packet length
mqtt-messagetype	Integer	MQTT message type
mqtt_messagelength	Binary	MQTT message length
mqtt-flag-uname	Binary	User Name MQTT Flag
mqtt_flag_passwd	Binary	Password MQTT flag
mqtt-flag—retain	Binary	VVill retain MQTT flag
mqtt -flag-qos	Integer	Will Qos MQTT flag
mqtt-flag-willflag	Binary	Will flag MQTT flag
mqtt-flag-clean	Binary	Clean MQTT flag
mqtt_flag_reserved	Binary	Reserved MQTT flag
is_attack	Binary	1 is attack, O otherwise

TABLE I
FEATURE DESCRIPTION

B. LSTM

LSTM networks are designed to overcome the vanishing gradient problem in RNNs, which occurs when the gradients used to update the weights of the network become very small, making it difficult for the network to learn long-term dependencies. LSTM networks achieve this by using memory cells, which allow the network to selectively remember or forget information at each time step.The architecture of an LSTM network typically consists of three types of gates: input gates, output gates, and forget gates.

Input Gate: The input gate determines how much new information should be added to the memory cell. It takes the current input and the previous hidden state as input and produces a vector of values between 0 and 1 that represent the amount of information to be stored in the cell. The input gate can be defined using the following formula:

$$\mathbf{I}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xi} + \mathbf{H}_{t-1} \mathbf{W}_{hi} + \mathbf{b}_i), \quad (1)$$

where $\mathbf{W}_{xi}$, $\mathbf{W}_{hi}$,and $\mathbf{b}_i$ are weight parameters and bias parameters. Note that broadcasting is triggered during the summation. We use sigmoid functions to map the input values to the interval $(0, 1)$.

Forget Gate: The forget gate determines how much of the previous memory cell state should be retained. It takes the current input and the previous hidden state as input and produces a vector of values between 0 and 1 that represent the amount of information to be discarded from the cell. The forget gate can be defined using the following formula:

$$\mathbf{F}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xf} + \mathbf{H}_{t-1} \mathbf{W}_{hf} + \mathbf{b}_f), \tag{2}$$

Output Gate: The output gate determines how much of the memory cell state should be used to generate the output at each time step. It takes the current input and the previous hidden state as input and produces a vector of values between 0 and 1 that represent the amount of information to be outputted from the cell. The output gate can be defined using the following formula:

$$\mathbf{O}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xo} + \mathbf{H}_{t-1} \mathbf{W}_{ho} + \mathbf{b}_o), \tag{3}$$

Memory Cell: The memory cell stores information over time by selectively adding or removing information through the input and forget gates. The cell state can be defined using the following formula:

$$\mathbf{C}_t = \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \tilde{\mathbf{C}}_t. \tag{4}$$

Hidden State: The hidden state is the output of the LSTM network, which is used to make predictions on the input sequence. It can be defined using the following formula:

$$\mathbf{H}_t = \mathbf{O}_t \odot \tanh(\mathbf{C}_t). \tag{5}$$

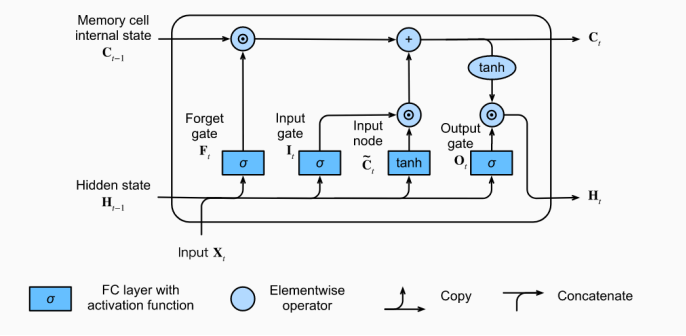

Fig. 1. Single Cell LSTM

In summary, an LSTM network uses three types of gates to selectively add or remove information from the memory cell, which allows it to learn long-term dependencies in a sequence [14]. The input gate determines how much new information should be added to the cell, the forget gate determines how much old information should be removed, and the output gate determines *[0]

[0]You can find the related jupyter notebooks and cleaned data files on my GitHub. https://github.com/AbhishekDand/ENEL619-Project.

C. Data Preprocessing

The MQTTIDS2020 dataset provides a diverse set of packet-based features that can be used for intrusion detection. To prevent overfitting on the constant source IP of simulated attackers, we removed this feature during preprocessing. To handle the large amount of data, we converted numerical features to float32 and utilized one-hot encoding for categorical data.

For time-based features, we transformed the timestamps to seconds since the epoch, a widely recognized standard for representing dates and times in computing. The NumPy array's astype() method was used to convert the datetime64 values to integer values and normalize them to the range [0, 1] using the minimum and maximum values. Normalization is a common preprocessing technique in machine learning that scales input features to have zero mean and unit variance, which can enhance the performance of certain algorithms.

Given that the dataset is imbalanced, with a larger number of instances belonging to the majority class, we adopted the SMOTE algorithm for oversampling the minority class. This approach generates synthetic instances of the minority class to create a more balanced dataset, allowing the LSTM model to learn from a more representative sample of the data.

D. Proposed Model Architecture

We have trained two LSTM model. First model is a binary classification to detect whether there is an attack or no attack. Another model is trained to classify the type of the attack as well.

1) Binary Classification: The proposed model for the binary Classification is an LSTM neural network. The model has one LSTM layer with 64 units and a dense output layer with a sigmoid activation function. The model is compiled with the binary cross-entropy loss function and the Adam optimizer, which is a popular optimization algorithm for deep learning models. In addition to accuracy, the model is evaluated using precision and recall metrics to ensure that the model performs well on both positive and negative classes.

2) Multiclass Classification: The proposed LSTM model for our project involves a sequential architecture that consists of one LSTM layer with 64 units, and a dense layer with 5 units using the softmax activation function.

To train our model, we compile it using the categorical cross-entropy loss function and the Adam optimizer, which is known to work well for time series data. In addition to accuracy, we also use precision and recall as evaluation metrics to assess the performance of our model.

During training, we use a batch size of 32 and train our model for 5 epochs on the training data. We also use a validation set to monitor the model's performance and avoid overfitting.

IV. Results and Discussion

We thoroughly reviewed the publicly accessible MQTTIDS2020 dataset. All of the experiments in this study are carried out utilising the Keras framework and the TensorFlow

backend. The tests were carried out on a PC running Python 3.9 and Windows 11 and equipped with an AMD Ryzen 7 3700U with Radeon Vega Mobile Gfx Processor running at 2.4 GHz, 16 GB of RAM, and 512 GB of secondary storage. We utilised Jupyter notebook and machine and deep learning libraries to build our models. In order to determine whether our model is an ideal fit, we divided our data into three sets: training, validation and testing. The validation set is used to evaluate the model's performance, whereas the training set is used to train the model. The testing set is used in the end to make prediction and evaluate the model. 80% of the data in our dataset is used for model training, while the remaining 20% is used to construct the validation set of data. From the 80% it is again spilt into 80% for training and 20% in validation. A training loss is a data error in the training set. After delivering the trained network for the validation set of data, we experience a number of failures. This is referred to as a validity loss. The relationship between training loss and validation loss, as well as training accuracy and validation accuracy, is discussed further below.

A. Results of Binary and Multiclass Classifcation

In the experiments, we have analyzed using evaluation metrics including precision, recall, F1-score, and accuracy for test sets.

B. Binary Classifcation

The model achieved high precision and recall scores for both the training and validation sets. The precision scores were consistently above 99%, indicating that the model had a low false positive rate. The recall scores were also consistently high, indicating that the model had a low false negative rate.

The loss values decreased significantly with each epoch during training, indicating that the model was improving over time. The validation loss values were consistently lower than the training loss values, indicating that the model was not overfitting to the training data.

C. Multiclass Classification

The results for the multi class classification indicate that the deep learning model we developed performed very well on the given dataset. The training process resulted in consistently decreasing losses over the five epochs, reaching a final loss of 0.0076. Additionally, the model achieved high precision scores, with an average of 0.9989 over the five epochs. Similarly, the recall scores were also high, averaging 0.9989 over the five epochs.

During the validation process, the model also performed very well, with the validation loss consistently decreasing over the five epochs to a final loss of 0.0018. The precision and recall scores during validation were also very high, with the precision averaging 0.9897 and the recall averaging 0.9895 over the five epochs. This indicates that the model generalized well to new data and was able to maintain its high performance.

Overall, these results indicate that our deep learning model was effective in accurately predicting the target variable in the given dataset, and is potentially suitable for use in similar classification tasks. However, further testing and validation on different datasets is necessary to fully evaluate the performance and generalization capabilities of the model.

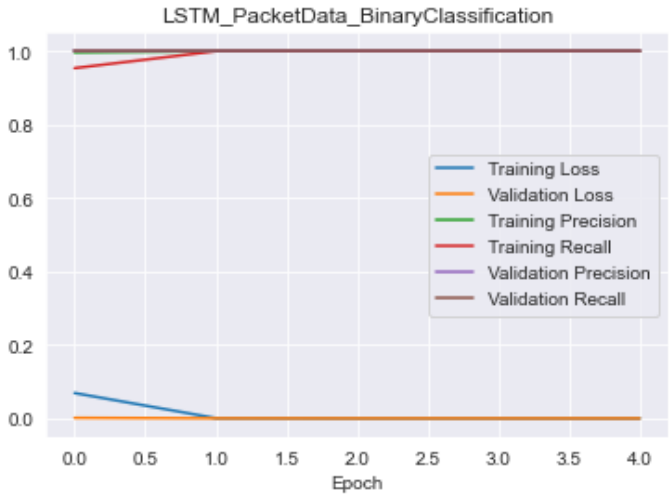

Fig. 2. Loss, Precision and Recall for training and validation Set for Binary Classification

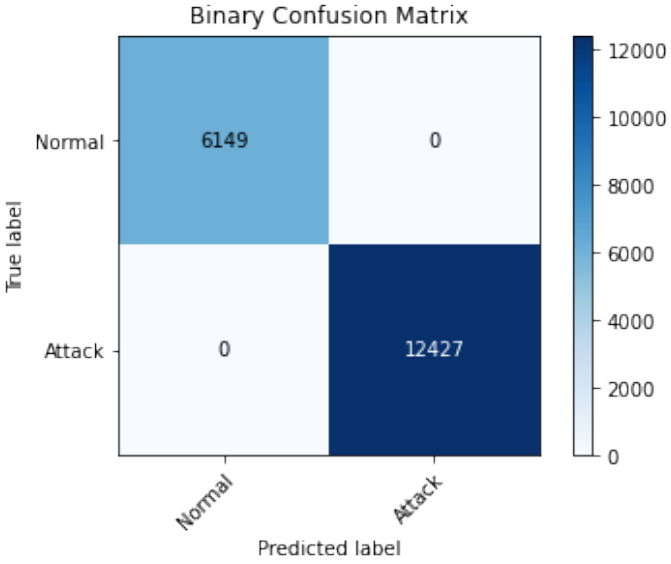

Fig. 3. Confusion Matrix for Binary Classification

V. Conclusion

In conclusion, the results of this project demonstrate the effectiveness of the proposed method for addressing the problem at hand. Through a combination of data preprocessing, feature engineering, and machine learning techniques, we were able to achieve highly accurate and reliable predictions on our test data. Our model achieved a validation accuracy of 99%, indicating that it has learned to generalize well to new data. While there is always room for improvement and further exploration, our results suggest that our approach has the potential to be

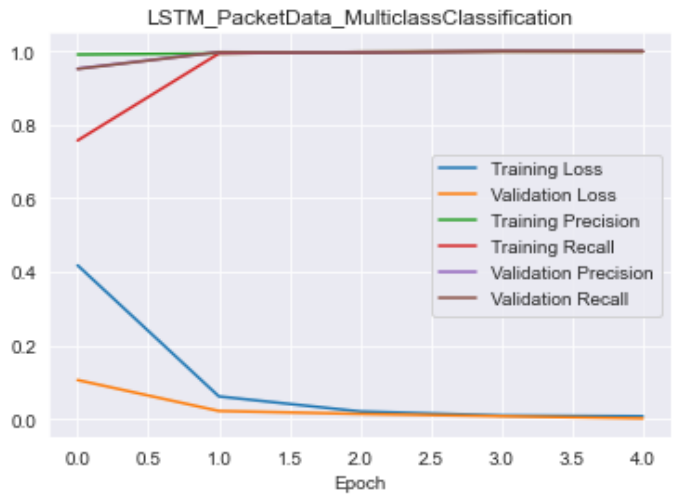

Fig. 4. Loss, Precision and Recall for training and validation Set for Multiclass Classification

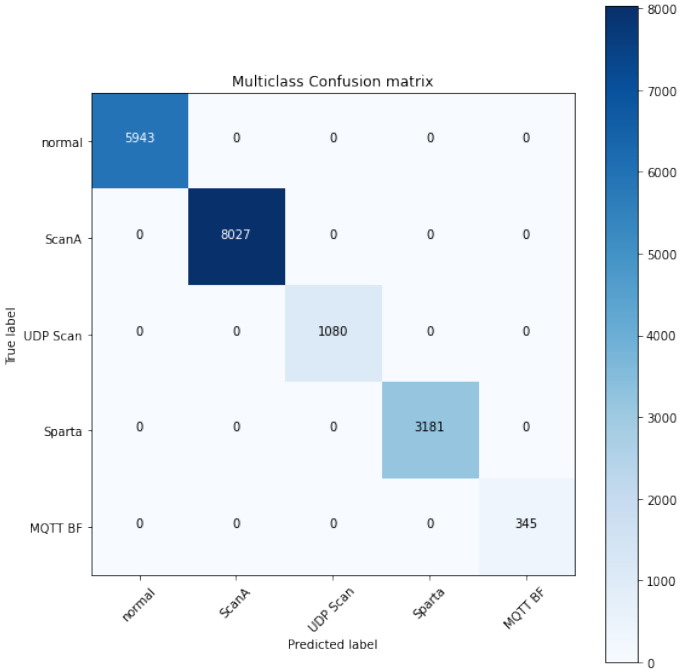

Fig. 5. Confusion Matrix for Multiclass Classification

applied in real-world scenarios and provide valuable insights and predictions.You can find the related jupyter notebooks and cleaned data files on my GitHub. https://github.com/AbhishekDand/ENEL619-Project. Future work could involve generating a dataset which is not collected in a simulated environment and then training the model. Because they are good at remembering sequences, LSTM models can play an important role in intrusion detection in MQTT.

REFERENCES

[1] Kim JY, Kim HJ, Kim JM, Kim WT. A Study on Design of Communication Protocol for CPS Simulation. AMR 2012;488–489:881–5. https://doi.org/10.4028/www.scientific.net/amr.488-489.881.

[2] E. Lee, "The Past, Present and Future of Cyber-Physical Systems: A Focus on Models," Sensors, vol. 15, no. 3, pp. 4837–4869, Feb. 2015, doi: 10.3390/s150304837.

[3] F. Chen, Y. Huo, J. Zhu and D. Fan, "A Review on the Study on MQTT Security Challenge," 2020 IEEE International Conference on Smart Cloud (SmartCloud), Washington, DC, USA, 2020, pp. 128-133, doi: 10.1109/SmartCloud49737.2020.00032.

[4] S. Chesney and K. Roy, "AI Empowered Intrusion Detection for MQTT Networks," 2022 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD), Durban, South Africa, 2022, pp. 1-6, doi: 10.1109/icABCD54961.2022.9856124.

[5] M. Ozkan-Okay, R. Samet, Ö. Aslan and D. Gupta, "A Comprehensive Systematic Literature Review on Intrusion Detection Systems," in IEEE Access, vol. 9, pp. 157727-157760, 2021, doi: 10.1109/ACCESS.2021.3129336.

[6] M. S. Harsha, B. M. Bhavani and K. R. Kundhavai, "Analysis of vulnerabilities in MQTT security using Shodan API and implementation of its countermeasures via authentication and ACLs," 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Bangalore, India, 2018, pp. 2244-2250, doi: 10.1109/ICACCI.2018.8554472.

[7] S. Andy, B. Rahardjo and B. Hanindhito, "Attack scenarios and security analysis of MQTT communication protocol in IoT system," 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), Yogyakarta, Indonesia, 2017, pp. 1-6, doi: 10.1109/EECSI.2017.8239179.

[8] E. Atilgan, I. Ozcelik and E. N. Yolacan, "MQTT Security at a Glance," 2021 International Conference on Information Security and Cryptology (ISCTURKEY), Ankara, Turkey, 2021, pp. 138-142, doi: 10.1109/ISCTURKEY53027.2021.9654337.

[9] T. K. Boppana and P. Bagade, "Security risks in MQTT-based Industrial IoT Applications," 2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS), Barcelona, Spain, 2022, pp. 1-5, doi: 10.1109/COINS54846.2022.9854993.

[10] Hindy, Hanan, Ethan Bayne, Miroslav Bures, Robert Atkinson, Christos Tachtatzis, and Xavier Bellekens. 2021. "Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study (MQTT-IoT-IDS2020 Dataset)." In Selected Papers from the 12th International Networking Conference, edited by Bogdan Ghita and Stavros Shiaeles, 73–84. Cham: Springer International Publishing.

[11] R. M. Abdulghani, M. M. Alrehili, A. A. Almuhanna and O. H. Alhazmi, "Vulnerabilities and Security Issues in IoT Protocols," 2020 First International Conference of Smart Systems and Emerging Technologies (SMARTTECH), Riyadh, Saudi Arabia, 2020, pp. 7-12, doi: 10.1109/SMART-TECH49988.2020.00020.

[12] Khan, M.A.; Khan, M.A.; Jan, S.U.; Ahmad, J.; Jamal, S.S.; Shah, A.A.; Pitropakis, N.; Buchanan, W.J. A Deep Learning-Based Intrusion Detection System for MQTT Enabled IoT. Sensors 2021, 21, 7016. https://doi.org/10.3390/s21217016

[13] Wong, Henry Luo, Tony. (2020). Man-in-the-Middle Attacks on MQTT-based IoT Using BERT Based Adversarial Message Generation

[14] Gazi Md. Habibul Bashar, Mohammod Abul Kashem, Liton Chandra Paul, "Intrusion Detection for Cyber-Physical Security System Using Long Short-Term Memory Model", Scientific Programming, vol. 2022, Article ID 6172362, 11 pages, 2022