With taking into consideration the following: (1): srcip (2): sport (3): dstip (4): dport (6): state (10): sttl (11): dttl (14): service (26): res_bdy_len 2.5 Functional requirements 2.5.1 Data collection The UNSWNB15 dataset is used to train the algorithm. It has 49 features divided into six categories, including flow, time, content, and others. The output was divided into binary values as "0" and "1" for attack and normal data. The output was divided into 9 different categories of contemporary and frequent attacks such as Dos, fuzzers, backdoors, dormitories, etc. Feature selection is the process of reducing the number of input variables when developing a predictive model, or, in other words, it is the process where you automatically identify those features in your data that contribute more to the prediction variable or output you care about. After the data is collected, we need to choose the features which has the highest correlation with the output In some circumstances, reducing the number of input variables can increase the performance of the model and decrease the cost of computation for modelling. Many models, especially linear ones like logistics and linear regression, might become less accurate when you have irrelevant variables in your data. Performing feature selection prior to modelling your data has three benefits: 1. Less frequent data means less opportunities for decisions to be based on noise, which reduces overfitting. 2. Increases accuracy: More accurate modelling is made possible by less misleading data. 3. Shortens training time: Algorithms train more quickly with less data. As we will see in the correlation section, we choose which features to be removed by determining whether they are unrelated to our output. However, for the time being and based on our initial analysis of the data, we can see that two features can be dropped, namely: 1. ID 2. Attack_cat Because our model is based on binary classification and the first one is an index rather than a descriptive feature, utilizing it will result in 100% accurate predictions but will not produce a generalizable model. Additionally, and for upcoming feature selection work, we can delete any characteristics that have a correlation between virtually zero and the output (in our case, "label") by analysing the connection between input features and the output (other than the ID and attack cat) For more detailed correlation views, we can check every feature and its correlation to other features by values from rages [−1,1], when the value is becomes much higher up to 1 the correlation is high and the opposite is true. 2.5.3 Data Preprocessing Row data must be transformed into the proper format for deep learning models as part of pre-processing. In terms of data pre-processing, categorical features will be translated into numerical form using data encoding method, and the data will be normalized. From our data, we can see that some integer features have a range of value (let's assume it as X), and other integers features have a range of value of 10X. dur feature has a max value of 59, spkts feature has a max value of 10646 … etc (This means big difference in distribution ranges). This means that when we want to train our model, the high distribution features will eliminate the small one and that is completely wrong because the small ones can have useful information which will help our model to be trained successfully. Therefore, we need to apply normalization, the extreme values should be pruned to reduce the skewness of some distributions. The logic applied here is that features with a maximum value of more than ten times the average value is trimmed to 95 percent. If the 95 percentage is close to the maximum, the tail contains more interesting information than we want to get rid of. The clamping is also applied to features up to a maximum of more than 10 times the average. This prevents the trimming of bi-media distributions and excessively small value. 2.5.4 Data Splitting

When data is divided into two or more subgroups, this is known as data splitting. A two-part split often involves testing or evaluating the data in one part and training the model in the other. Data splitting is frequently used in deep learning to prevent overfitting. In that situation, a deep learning model fits its training data too well and is unable to consistently fit new data. The dataset will be split into training and testing data after pre-processing. The subset of the data used to train the model is known as training data. In order to improve any of its parameters, the model must observe and learn from the training set. The data that is examined in the final model and contrasted with the earlier data sets is referred to as test data. The testing set serves as an assessment of the chosen algorithm and mode. The dataset was divided into 20% for validation and 80% for training. It took 200 epochs to complete the training phase. 2.5.5 Model Training A training dataset is a starting set of data that teaches DL models how to recognize specific patterns or carry out a specific task. Data will be separated into training and testing data after pre-processing. Applying the deep learning model to the training data will provide a trained model that will enable us to arrive at the desired result (output). The training process is as follows: ● Data is fed to the LSTM neural network to give us labelled data. ● Productive labelled data is compared with the original output labelled data. ● If not match, an error fed our neural network with a feedback called backpropagation in order to enhance the model's output in the next epoch. 2.6 Compared and evaluated method It isn't easy to compare our proposed method to all available Machine Learning techniques. We will only compare it to the most effective and well-liked ones, such as Logistic Regression, KNN, and Decision Tree. 3.1 Introduction In this chapter, we will discuss the obtained results from our LSTM deep learning model which task is to classify the packets based on binary classification to normal packets and attack packets (malware packets sent to the victim machine or attempt to gain unauthorized access to a computer) using the UNSWNB15 dataset, we will evaluate the model and check the accuracy and performance metrics, compare it to most common ML algorithms, visualize the results and summarize the work with a future action plan. 3.2 Work environment The GPU from the Google Colab was used. Two Intel(R) Xeon(R) CPU cores running at 2.20 GHz, 10 GB of RAM, and 72 GB of hard drive space made up the specifications. Tensorflow was version 2.8.0, and Python was version 3.8. 3.3 Implementation and evaluation metrics Model evaluation is the process of using various evaluation measures to understand the performance and strengths and weaknesses of a machine and deep learning model. It is critical to evaluate a model's effectiveness during the early stages of research. Model evaluation can also help with model monitoring. To evaluate the deep learning model's accuracy, we will compare the trained model with the test data. In order to determine algorithm performance, evaluation metrics will be examined. Popular metrics for DL models, such as MSE (Mean Squared Error), RMSE (Root Mean Squared Error), and MAE (Mean Absolute Error) were used in this instance. ● The Mean absolute error is the average of the absolute difference between the dataset's actual and predicted values. It computes the average of the residuals in the dataset. ● The mean squared error is defined as the mean of the squared difference between the original and forecasted values of a data set. It computes the variance of the residuals. ● Root Mean Squared Error is the square root of Mean Squared Error. It computes the standard deviation of the residuals. 3.4 performance indicators The effectiveness of a given model cannot be determined by a single metric. As a result, we evaluated the

models using a variety of metrics: Accuracy: The accuracy of the model is a measure of its overall performance. It is useful when all of the categories are equally important. It is calculated by dividing the number of correct forecasts by the total number of predictions. Precision: Precision is calculated as the proportion of correctly diagnosed positive samples to all positive samples (either correctly or incorrectly). Precision is a metric that expresses how accurately the model classified the sample as positive. Precision decreases and the denominator grows when the model assigns many false positives or few true positives. However, the precision is high when: ● The model consistently assigns many positive classifications (maximize True Positive). ● Less incorrect Positive classifications are made by the model (minimize False Positive). Recall: The recall is determined by dividing the total number of positive samples by the proportion of positive samples that were accurately identified as positive. The recall gauges how well the model can identify positive samples. More positive samples are found the higher the recall. F1 Score: The F1 Score is the harmonic average of recall and precision. Range of the F1 Score is [0, 1]. It informs you of the strength of your classification as well as how accurate it is (the number of cases it correctly categorizes) (not missing a large number of cases). Since Precision and Recall are average in the F1 score, Precision and Recall are given equal weight in the F1 score: If both Precision and Recall are high, a model will receive a high F1 score. If both Precision and Recall are low, a model will receive a low F1 score. If one of Precision and Recall is low and the other is high, a model will receive a medium F1 score. Confusion Matrix: A Confusion matrix is a N X N matrix used to assess the effectiveness of a classification model, where N is the total number of target classes. The matrix contrasts the actual target values with those predicted by the model. High TP and TN rates and low FP and FN rates are indicators of a strong model. Confusion matrices are frequently used because they provide a more accurate picture of a model's performance than classification accuracy provide. For example, there is no information on the number of cases that were incorrectly identified in the classification accuracy. False positive rate: percent of negative cases in the data that were mistakenly classified as positive cases True positive rate: proportion of correct predictions in predictions of positive class. Where: ● True Positive (TP): The proportion of attack samples that were correctly foreseen. ● False Positive (FP): The quantity of attack samples that were incorrectly expected to occur. ● True Negative (TN): Quantity of normal samples that were accurately predicted. ● False Negative (FN): Number of mistakenly predicted normal samples 3.5 Experimental results A deep learning model is used in the proposed study to enhance the intrusion detection system. In this instance, the UNSW NB15 dataset's IDS accuracy is determined using an LSTM deep learning model. The suggested work is carried out in Python 3.8 using the Keras, TensorFlow, Matplotlib, and other required files libraries. Here, we use LSTM and evaluation metrics to calculate the algorithm performance and obtain an algorithm accuracy of 96.70%. The outcome demonstrates that the LSTM algorithm is effective for intrusion detection. The LSTM structure can be seen in Figure 2. Figure 3 displays the model's accuracy utilizing training and validation data with a 200–epoch LSTM method. The model loss accuracy of the training and validation sets using the LSTM algorithm with 200 epochs is shown in Figure 4. The performance indicators to assess the robustness of our model are shown in Figure 5. Now, we will compare our suggested model with logistic Regression, KNN and Decision tree. Table 6 displays the model accuracy

of training and validation data using the Logistic Regression algorithm. Table 7 and Table 8 display the model accuracy of training and validation data using the KNN algorithm and the Decision tree algorithm, respectively. The efficiency of each model (chosen and comparative models) is finally summarized in Table 9 by displaying accuracy along with performance indicators and evaluation metrics. Our model performance is the best model among other models in all evaluation metrics. ● Regarding the evaluation metrics (MSE, MAE, RMSE), we can notice that our proposed model achieved the least error values among all other models. Finally, we have to compare our proposed model with models from previous studies which all in common by using the UNSWNB15 dataset with a binary classification type. Conclusion and future work The study presented an effective deep learning-based network attack detection approach. Furthermore, the UNSW NB15 network intrusion public Dataset was used as a classifier, and the classification procedure was carried out using the LSTM algorithm which gives us an accuracy of 96.70%. On the same data and criteria, we compared our model to others. From the results, we can see that our LSTM model is more efficient than the other ML algorithms which means that using DL for UNSWNB15 dataset is better than ML to build the IDS system for smart cities and IoT devices. Our results can be applied to detect cyberattacks in smart city systems and IoT devices due to the efficient and robustness we achieved. For future research, we will first apply more LSTM variations and evaluate the effectiveness of complex LSTMs with dimensional reduction methods before applying the model to the actual and integrated network capture module to create the online intrusion detection model. References: [1] The concept of 'smart cities'. Towards community development?, https://journals.openedition.org/netcom/1105 [2] Jeffin J 2016 Smart City, Structuring a Smarter India, https://www.cronj.com/blog/smart-city- structuring-a-smarter-india/amp/ [3] Top 7 IoT applications for Smart Cities, https://www.baseapp.com/iot/iot-applications-for-smart-cities/ [4] IoT Architecture Models, https://bernhardwenzel.com/2019/iot-architecture-models/ [5] Giovanni Apruzzese, "On the Effectiveness of Machine and Deep Learning for Cyber Security," 2018 10th International Conference on Cyber Conflict, 2018, pp. 20, https://ieeexplore.ieee.org/document/8405026 [6] Dua, M. (2019, June). Machine learning approach to IDS: A comprehensive review. In 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA) (pp. 117-121). IEEE. [7] Liu, H., & Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. applied sciences, 9(20), 4396. [8] Musa, U. S., Chhabra, M., Ali, A., & Kaur, M. (2020, September). Intrusion detection system using machine learning techniques: A review. In 2020 international conference on smart electronics and communication (ICOSEC) (pp. 149-155). IEEE. [9] Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Transactions on Emerging Telecommunications Technologies, 32(1), e4150. [10] Kilincer, I. F., Ertam, F., & Sengur, A. (2021). Machine learning methods for cyber security intrusion detection: Datasets and comparative study. Computer Networks, 188, 107840. [11] Karatas, G., Demir, O., & Sahingoz, O. K. (2020). Increasing the performance of machine learning-based IDSs on an imbalanced and up-to-date dataset. IEEE Access, 8, 32150-32162. [12] bhai Gupta, A. R., & Agrawal, J. (2020, April). A comprehensive survey on various machine learning methods used for intrusion detection system. In 2020 IEEE 9th International

Conference on Communication Systems and Network Technologies (CSNT) (pp. 282–289). IEEE. [13] Handa, A., Sharma, A., & Shukla, S. K. (2019). Machine learning in cybersecurity: A review. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 9(4), e1306. [14] Thomas, R., & Pavithran, D. (2018). A survey of intrusion detection models based on NSL-KDD data set. 2018 Fifth HCT Information Technology Trends (ITT), 286–291. [15] Abd Jalil, K., Kamarudin, M. H., & Masrek, M. N. (2010, June). Comparison of machine learning algorithms performance in detecting network intrusion. In 2010 international conference on networking and information technology (pp. 221–226). IEEE. [16] Alzahrani, A. O., & Alenazi, M. J. (2021). Designing a network intrusion detection system based on machine learning for software defined networks. Future Internet, 13(5), 111. [17] Mohammadi, S., Mirvaziri, H., Ghazizadeh-Ahsaee, M., & Karimipour, H. (2019). Cyber intrusion detection by combined feature selection algorithm. Journal of information security and applications, 44, 80–88. [18] Roopak, M., Tian, G. Y., & Chambers, J. (2019, January). Deep learning models for cyber security in IoT networks. In 2019 IEEE 9th annual computing and communication workshop and conference (CCWC) (pp. 0452–0457). IEEE. [19] Dutta, V., Chora, M., Pawlicki, M., & Kozik, R. (2020). A deep learning ensemble for network anomaly and cyber-attack detection. Sensors, 20(16), 4583. [20] Surya, V. (2021). A Review on Deep Learning and Intrusion Detection System Technologies to Secure IoT. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(11), 3642–3656. [21] Vaiyapuri, T., Sbai, Z., Alaskar, H., & Alaseem, N. A. (2021). Deep learning approaches for intrusion detection in IIoT networks–opportunities and future directions. International Journal of Advanced Computer Science and Applications, 12(4). [22] Ibor, A. E., Oladeji, F. A., Okunoye, O. B., & Ekabua, O. O. (2020). Conceptualisation of Cyberattack prediction with deep learning. Cybersecurity, 3(1), 1–14. [23] Abu Al-Haija, Q., & Zein-Sabatto, S. (2020). An efficient deep-learning-based detection and classification system for cyber-attacks in IoT communication networks. Electronics, 9(12), 2152. [24] Zhong, M., Zhou, Y., & Chen, G. (2021). A security log analysis scheme using deep learning algorithm for IDSs in social network. Security and Communication Networks, 2021. [25] Thakkar, A., & Lohiya, R. (2021). A review on machine learning and deep learning perspectives of IDS for IoT: recent updates, security issues, and challenges. Archives of Computational Methods in Engineering, 28(4), 3211–3243. [26] Alzahrani, M. S., & Alsaade, F. W. (2022). Computational Intelligence Approaches in Developing Cyberattack Detection System. Computational Intelligence and Neuroscience, 2022. [28] Deep Learning | Introduction to Long Short Term Memory,
https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/ [29] LSTM — Introduction in simple words,
https://medium.com/nerd-for-tech/lstm-introduction-in-simple-words-fe544a45f1e7 [27] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735–1780. [28] Patel, A.; Qassim, Q.; Wills, C. A survey of intrusion detection and prevention systems. Inf. Manag. Comput. Secur. 2010, 18, 277–290. [29] Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. Cybersecurity 2019, 2, 20. [30] Yuan, L.; Chen, H.; Mai, J.; Chuah, C.N.; Su, Z.; Mohapatra, P. Fireman: A toolkit for firewall modeling and analysis. In Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06),

Berkeley/Oakland, CA, USA, 21–24 May 2006; IEEE: Manhattan, NY, USA, 2006; pp. 15–213. [31] KDD Cup; UCI KDD Archiv: Irvine, CA, USA, 28 October 1999. Available online: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html [32] Tavallaee, M.; Bagheri, E.; Lu,W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6. [33] Revathi, D.a.M.S.; Revathi, S.; Malathi, A. A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection. Int. J. Eng. Res. Technol. 2013, 2, 1848–1853. Available online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.680.6760&rep=rep1&type=pdf [34] Kayacik, H.G.; Zincir-Heywood, A.N.; Heywood, I.M. Selecting features for intrusion detection: A feature relevance analysis on KDD 99 intrusion detection datasets. In Proceedings of the Third Annual Conference on Privacy, Security and Trust, San Francisco, CA, USA, 20–22 October 2010; Volume 94, pp. 1722–1723. [35] P. Hick, E. Aben, K. Claffy, and J. Polterock, "the CAIDA DDoS attack 2007 dataset," ed, 2007 [36] Moustafa, N.; Jill, S. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the Military Communications and Information Systems Conference (MilCIS), Canberra, ACT, Australia, 10–12 November 2015; IEEE: Manhattan, NY, USA; pp. 1–6. [37] Moustafa, N.; Slay, J. The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Inf. Secur. J. 2016, 25, 18–31. [38] Feature Selection: Beyond feature importance?, https://medium.com/fiverr-engineering/feature-selection-beyond-feature-importance-9b97e5a842f [39] Feature Selection Tutorial, https://prwatech.in/blog/tag/types-of-feature-selection-methods/ [40] What is a confusion matrix? https://medium.com/analytics-vidhya/what-is-a-confusion-matrix-d1c0f8feda5