Ripple20 Vulnerabilities Detection using Featureless Deep Learning Model

Sarah Binhulayyil*†, Shancang Li*
* *School of Computer Science and Informatics*
Cardiff University
Cardiff, UK
lis117@cardiff.ac.uk
† *College of Applied Studies and Community Service*
King Saud University
Riyadh, KSA
binhulayyilsh1@cardiff.ac.uk

***Abstract*—Inherent vulnerabilities create new security risks and challenges that leave Internet of Things (IoT) systems open to cyber attacks. Featureless deep learning shows great potential in vulnerabilities detection without relying on explicit feature engineering in the IoT. Featureless deep learning models provide a low-cost and memory time-series analysis of network traffic. This paper proposes a featureless deep learning procedure in a 1D CNN model to carry out Rippl20 detection. The experimental results demonstrate the effectiveness of proposed solution with it is beneficial for decreasing the time spent on feature engineering. Specifically, this proposed featureless model achieved 99% of accuracy and a F_1 score as 0.9991 with less time than traditional methods.**

***Index Terms*—IoT, vulnerability detection, featureless model, deep learning**

I. Introduction

The Internet of Things (IoT) devices are associated with many vulnerabilities that facilitate to exploitation these vulnerabilities to conduct attacks, which cause to steal sensitive data or misuse these devices to achieve their purposes. The common vulnerabilities in IoT devices lead to different types of attacks such as the "Top 10 vulnerability" that released by NIST *poor passwords, insecure services, insecure interfaces, lack of update mechanisms, insecure/outdated components, inadequate privacy protection, insecure data transfer/storage, lack of device management, insecure default setting, and lack of physical protection.*[1]

The majority of the data being transferred and stored by IoT devices should be secured while doing so, whether it is done wirelessly or through wires. According to a survey by Aruba, 84% of all recorded security incidents involving data breaches in 2019 occurred on IoT devices, and these assaults may discourage people from using IoT technologies [1]. In addition, the most recent vulnerability in IoT devices that discovered in 2020 by Joint Special Operations Forces (JSOF), which consists of 19 zero-day vulnerabilities which lead to multiple attacks such as remote code execution additionally the attacker can hide an embedded code within a device for many years [2]. This vulnerability could be in IoT industrial and daily uses and also in the Health care as NHS reported [3].

Ripple20 is a set of 19 zero-day vulnerabilities targeting IoT devices, which could allow malicious attackers to launch remote attacks, such as DDoS, take control of IoT devices, steal sensitive information, *etc.* The report that issued by JSOF shows that Ripple20 could cause many attacks such as remote execution, DDoS attacks, remote controlling of a device, MITM attack and DNS cache poisoning [2]. In addition, it had been reported by [4] that Ripple20 is affected millions of IoT devices from different fields. The majority of IoT network devices are equipped with intrusion prevention systems (IPS) or intrusion detection systems (IDS), which scan the traffic and classify it as secured or unsecured based on their rule sets and recognition algorithms. However, these networks need to have more capability to detect the vulnerabilities before exploiting them with faster rates, more processing power and memory.

This paper focuses on detecting Ripple20 vulnerability in the IoT environment by using Deep Learning (DL) with consideration of the limitations of IoT devices. Due to the impacts of exploitation of this vulnerability, it is crucial to develop techniques to detect this vulnerability to keep these devices secure and ensure the confidentiality, integrity, availability, authentication and authorisation are verified. The main contribution of this work can be summarised as

- This work proposes an featureless IoT vulnerabilities detection model without requiring explicitly defined features as input but instead learns to extract vulnerability features from the provided Ripple20 dataset.
- As a use case, the proposed featureless model was used to detect the critical vulnerability Ripple20 in IoT devices that are connected to a home network, which leads to identifying home networks that pose a risk to other connected devices.
- The experimental results show that the model can successfully differentiate between IoT devices that are vul-

[1]https://www.cardinalpeak.com/blog/top-10-iot-security-vulnerabilities

nerable to the Ripple20 vulnerability and those that are secure and free of the causes of this issue.

This paper is organized as follows: A review of related works and background are described in Section 2. The methodology which contains the approach, the experiment details and the proposed model architecture are presented in Section 3. The results and the evaluation of the proposed model are shown in Section 4. Finally, conclusions and future works are covered in Section 5.

II. Background and Related works

More recent attention has focused on the provision of securing IoT ecosystems. As known, IoT devices have many vulnerabilities that have been detected in each layer. The most critical one is Ripple 20 vulnerabilities which consist of 19 zero-day vulnerabilities, and it had been released by the JSOF team in 2020. Of these vulnerabilities, four of them had been ranked critical which have Common Vulnerability Scoring System (CVSS) 9 and they lead to remote execution attacks, whereas four of them have high severity 7 in CVSS rank and the rest of them have been ranked low severity but they could lead to execute DoS attacks and data leaks. These critical vulnerabilities are located in Treck TCP/IP stack which has been used since 20 years ago. It allows an attacker to execute remote code and get gain on an IoT device, DoS attacks and steal data [2].

There are few researches that focus on detecting a vulnerability in IoT devices so detecting IoT vulnerabilities still required more research and proposed solutions [5]. Static analyses are one of the most methods that has been used to detect IoT vulnerabilities which are shown in [6] to detect five of the "Top 10 OWASP" by using an external industrial analyzer (Julia). Other research aims to detect the MQTT vulnerabilities in IoT environments by using IoT-Enabled IDS. It proposed an IDS engine that could be incorporated with the initial layer of the IDS to check extensively the validation for the packets to detect IoT protocols vulnerabilities [7].

The latest study used DL to detect vulnerable IoT applications which focused to detect the flaws in the applications code. They proposed a hybrid model that consist of *CodeBERT, Word2Vec*, and *FastText* and used IoT open-source APIs [8]. It could be noticed that there is no previous research that aims to detect Ripple20 vulnerabilities in IoT devices either using machine learning or IDS, so this paper is the first study that focuses to detect Ripple20 by using DL and featureless technique. As DL plays an important role in the maintenance security of IoT devices, it indicates a need to understand the various perceptions of using featureless that exist among IoT environments due to the advantages of using it to be more realistic with conducting DL models in real-time. There are a few previous studies that used featureless technique in IoT security but have only been carried out in a few areas.

The first study that proposed the featureless technique was in 1997 which aims to conduct featureless to make the models able to recognise the patterns in different types of classifier models [9] and it got good results in most of the classifiers. Then in 2004, a featureless model was used with Support Vector Machine (SVM) in the healthcare section specifically in digital mammograms to classify and detect lumps [5].

After that, it has been performed in many sectors in healthcare and industry, however, in IoT security it has been conducted a few series of studies recently to improve the device's security. The first study that proposed a featureless model to detect attacks in IoT devices and it used the IoT-23 dataset with a 1D-CNN model. It resulted in about 100% in accuracy whereas the F1 scores between 79% and 88% [10]. The latest study that using featureless to enhance the security in IoT environments by detecting Botnet attacks. It used the same dataset IoT-23 with GRU (Gated Recurrent Unit) models and resulted with 99.87% accuracy [11].

In summary, the key challenges in deep learning based IoT vulnerabilities detection including: limited labeled data, data imbalance, feature extraction and representation, interpretability and explainability, etc. This work proposes featureless models in IoT vulnerabilities detection that can well address the *feature extraction and representation, limited labeled data* issues.

III. Methodology

A. The Approach

Unlike traditional machine learning techniques, where features were identified as input of the machine learning model, the featureless technique can learn features from the raw input rather than expert manually engineered from the input. The featureless techniques use multiple layers of deep learning networks to learn complex representations (features) from the input and show great potential in automatically learning features from the raw input. The featureless learning techniques can achieve better performance, such as higher accuracy, than existing deep learning-based models [12], [13].

The RNN (Recurrent Neural Networks) is one of the powerful featureless tools that can be used for anomaly detection in IoT. Traditional machine learning-based anomaly detection techniques require the manual extraction of features from high-dimensional, time-series data, which usually are time-consuming and may not be able to capture all key features. The RNN can learn to represent the temporal dependencies in the time-series data and can capture all information from previous time steps. Using RNNs (e.g., the Long short-term memory, LSTM), it is possible to detect cyber attacks or abnormal behavior in real time without the need for manual feature engineering. This can help implement accurate and quick responses against cyber incidents.

A featureless model can be used in prediction, classification, etc., which does not have specific input features. A featureless model involves internal mechanisms f_I (such as algorithms, decision-making mechanisms, etc.), external factors f_E, dynamic factors like time t and environment factors E, and data X, as shown in Eq.(1)

$$Y = \mathcal{G}(f_I, f_E, t, E, X) \tag{1}$$

in which the $\mathcal{G}$ could be unsupervised learning techniques that can be used to discover patterns or structures within the data.

In cyber security scenarios, to detect vulnerabilities from a given input data x_i from a collection of data X, output $v \in V(x)$ gives the potential vulnerabilities of x_i. In this end, $V(x)$ is the set of detected vulnerabilities $|V(x)| = n$. For a labelled data to learn a function $f : V(x) \rightarrow \{0, 1\}$,

$$f(x) = \begin{cases} 1, & \text{if } v \notin \emptyset \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

To learn function f, we can use Ripple20 which includes both benign and malicious data. Let $M \in V(x)$ denote true vulnerabilities in $V(x)$. In the packets stream, since there are many more benign packets than malicious packets, which makes it difficult for the machine learning model to detect a real vulnerability.

In this work, we use a convolutional neural network (CNN) as a classifier to learn which vulnerability can be identified. A candidate $x \in X$ is converted to a 1D representation and then fed through a one-dimensional convolution layer. The outputs are then fed through a max-pooling layer. After a one-dimensional max-pool layer, the results are then passed through a dense layer (Hidden layer), which is a fully connected layer and outputs a single number that represents the probability of the input is a vulnerable or non-vulnerable candidate. Figure .1 presents the overall architecture of the featureless model.

Featureless instance-based DL model for detecting Ripple20 vulnerability in IoT environment to protect the network from any outside or inside attacks. Featureless means dealing with data as it is in order to be more realistic and save time whereas the steps involved in feature engineering include defining the features, extracting them from the raw data, cleaning, scaling, and consolidating. It is a complex, lengthy, and instantaneous procedure.

Featureless modeling recently is crucial in the IoT ecosystem due to the benefits that it has. The main point of featureless is converting the data to bytes and then passing these data to the model without needing for modifications on the raw traffic data. As a result, the procedure for extracting features has been removed from the machine learning steps [10]. Using featureless would be beneficial and effective especially in IoT environments due to the limitations of the IoT devices in memory and time consuming.

B. Experiments design

1) Collecting Data: The experiment has been implemented by collecting data from the Cardiff University Smart Home lab. The data is a traffic network that is collected from an isolated smart home network. The smart home network contains of 12 smart home devices 6 of them have Ripple20 vulnerabilities and others have not. To train and imply the model, we used NVIDIA GeForce RTX 3080 on a 32 GB, Ubuntu 22.04.1 LTS server and the CPU Model is 12th Intel i7-12700.

In the begging, we selected six devices that are associated with the Ripple20 vulnerability and six devices that are free and secure from the vulnerability, by scanning the devices using the Nessus platform to determine and confirm which devices have Ripple20 vulnerability and which have not. Then, start to collect the network packets from these devices by using Wireshark for four hours period due to as known Ripple20 vulnerability associated with Treck TCP/IP which could be shown at the beginning of the connection when the used protocols start with the authentication process. It resulted in the total of all packets being around 253914 in a Pcap file.

2) Analyse and label the data: This step aims to separate the network traffics that are contained Ripple20 vulnerability and the network traffics that are free from this vulnerability. Then, label these data with the appropriate label which will be clear for the model to train and learn the pattern from these traffics.

3) Pre-processing the data: As using featureless, in this step there was just one point that aims to convert the network packets to bytes to pass them to the model by using Python scripts.

C. The model architecture

This study creates a featureless, small, and simple one-dimensional Convolutions Neural Network (1D-CNN) model that could be embedded within small devices such as IoT and routers to detect the critical threat of Ripple20 vulnerability throughout the network. Fig. 1 shows featureless model details which could be summarized as a 1D-CNN with fully connected layers, two CNN layers, and hidden layer(s). This case could be considered a time series issue due to the network packets coming into the device could be presented as cases of dispersed over time. There are many previous studies that used 1D-CNNs to classify data of time series [14] [15] [16].

For the input layer it has been used Embedding layer to convert the bytes into continuous vectors which are represented have common contexts and relationships between them. Then pass it into CNN layers which are used two types layers to learn from the instances from higher level features [17].

The input layer is followed by the Conv1D layer, then the output will be passed to the MaxPooling1D layer process to decrease the output size. Then, we used in this step two (Dense) hidden layers and also tried one hidden layer which is implemented to make the model weights are taught a nonlinear connection. Finally, the output layer which is also Denes with 'sigmoid' activation that is the appropriate function for using a prediction model which has a probability that only occurs between 0 and 1 [18]. Due to our model working with binary classification, during the model compiler, we used binary cross-entropy as a loss function and Adam function for optimiser with a learning rate of 0.0001 to avoid the model over-fitting.

IV. Evaluation and Results

The evaluation of this experiment is conducted by calculating the metrics accuracy, loss, precision, recall, and F_1 score. Table I presents the general result and some of the main characteristics of the experiment for both using of the

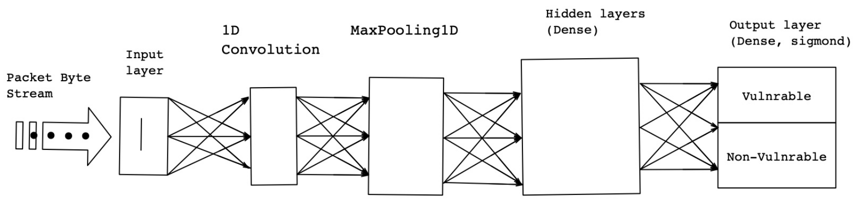

Fig. 1. The featureless instance-based DL model

model -with one hidden layer and two hidden layers. Table II compares more details of training and testing results.

TABLE I
SUMMARY OF THE RESULTS

Acc	Prec	Recall	F1	CPU	Memory
0.9996	99994	0.99912	0.99953	128	32

Turning to the details, accuracy is described as the proportion of correctly classified data among all classifications made by the model, and it is used for both training and validation data. To get the F1 score, it needs to calculate Precision and recall. Precision is a classification model's capacity to isolate and highlight the most important data items. It can be defined mathematically, as the ratio of the number of true positives and the sum of the true positives and false positives. In addition, the Recall refers to the model's capacity to locate all relevant cases within a data collection, and as a mathematical expression, it is defined as the ratio between the total number of true positives and the total number for true positives along with the number of false negatives.

Another indicator of the performance of the model is the F_1 score. It represents the weighted average of the validation data's precision and recall levels. In our case, we used 80% of the data for training and 20% for evaluation [19].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

in which TP denotes *true positive*, TN denotes *true negative*, FP denotes *flase positive*, and FN denotes *false Nngative*, respectively. Then we have

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{6}$$

It is apparent from the table II that the results from this experiment are interesting by comparing the accuracy, recall, precision, and F_1 score for both using one or two hidden layers are almost 99% which does not affect the result of any evaluation matrices, whereas the differences appear in the training time and Loss matrix which consumes about $2,379s$ to train the model for two hidden layers, and about $837.57s$ for using the model within one hidden layer. The differences in Loss for training and testing data by using one or two hidden layers are shown in Table. II, it is slightly different in Loss, however, the comparison of both models is shown clearly significant difference in the time consumption and less complexity which indicates the suitability for using it in IoT devices.

To ensure the efficiency of the model, we calculated the Accuracy and Loss for training and validation data as can be seen in figure 2 and 3 for both using one hidden layer and two hidden layers, which indicate a slight difference for both due to the modifying for the learning rate for the optimiser into 0.0001 to avoid the over-fitting for the model. In addition, figures 4 show the number of TP and TN are the same whereas the number of FN and FP are slightly different, which ensures the success of the featureless model to identify the IoT devices within Ripple20 vulnerability. As a result, using the featureless model with one or two hidden layers does not affect the result of any evaluation matrices.

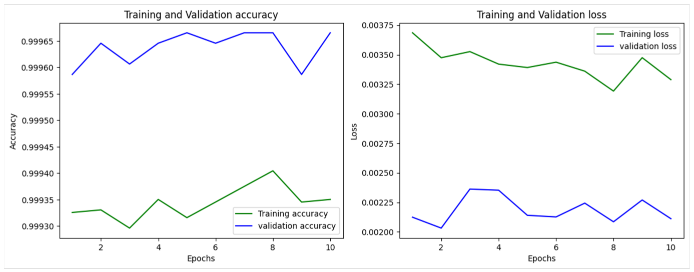

Fig. 2. Accuracy and loss values for training and evaluation using two hidden layers

V. CONCLUSION

This work proposed a featureless model for IoT device vulnerabilities detection that does not require handcrafted features but be able extract Ripple 20 vulnerabilities features from the dataset. The experimental result show that the proposed model can effectively detect this vulnerability with 99% accuracy, precision, recall, and F_1 score.

REFERENCES

[1] K. Ashton, "Making sense of iot. how the internet of things became humanity's nervous system," *Hewlett Packard Enterprise*, 2017.

TABLE II
THE RESULTS FOR TRAINING AND TESTING BY USING A DIFFERENT NUMBER OF HIDDEN LAYERS

	Training			Testing	
	Accuracy	Loss	Time	Accuracy	Loss
Two hidden layers	0.999	0.0037-00.32	2317.39s ∼38.5m	0.999	0.0024-0.0020
One hidden layer	0.999	0.0200-0.0175	837.57s ∼13.5m	0.999	0.132-0.0084

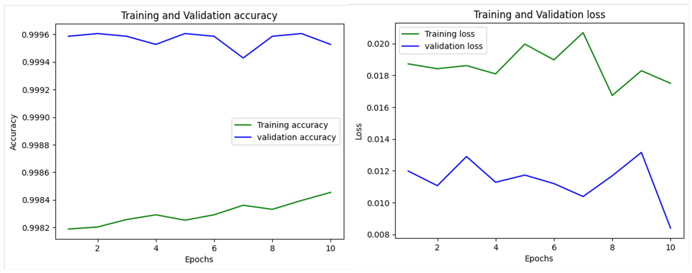

Fig. 3. Accuracy and loss values for training and evaluation using one hidden layer

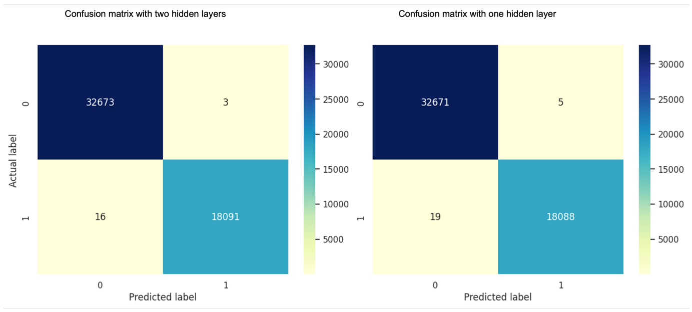

Fig. 4. Confusion matrix for one and two hidden layers

[2] JSOF, "Overview- ripple20," 2020.
[3] NHS, "Ripple20 network vulnerabilities," 2020.
[4] T. Seals, "'ripple20'bugs impact hundreds of millions of connected devices," *Threatpost, June*, vol. 16, 2020.
[5] R. Campanini, D. Dongiovanni, E. Iampieri, N. Lanconelli, M. Masotti, G. Palermo, A. Riccardi, and M. Roffilli, "A novel featureless approach to mass detection in digital mammograms based on support vector machines," *Physics in Medicine & Biology*, vol. 49, no. 6, p. 961, 2004.
[6] P. Ferrara, A. K. Mandal, A. Cortesi, and F. Spoto, "Static analysis for discovering iot vulnerabilities," *International Journal on Software Tools for Technology Transfer*, vol. 23, pp. 71–88, 2021.
[7] M. Husnain, K. Hayat, E. Cambiaso, U. U. Fayyaz, M. Mongelli, H. Akram, S. Ghazanfar Abbas, and G. A. Shah, "Preventing mqtt vulnerabilities using iot-enabled intrusion detection system," *Sensors*, vol. 22, no. 2, p. 567, 2022.
[8] H. Mei, G. Lin, D. Fang, and J. Zhang, "Detecting vulnerabilities in iot software: New hybrid model and comprehensive data analysis," *Journal of Information Security and Applications*, vol. 74, p. 103467, 2023.
[9] R. P. Duin, D. de Ridder, and D. M. Tax, "Experiments with a featureless approach to pattern recognition," *Pattern Recognition Letters*, vol. 18, no. 11-13, pp. 1159–1166, 1997.
[10] A. Khan and C. Cotton, "Detecting attacks on iot devices using featureless 1d-cnn," in *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pp. 461–466, IEEE, 2021.
[11] W. RA, S. UK, *et al.*, "Detection of iot botnet using machine learning and deep learning techniques," 2023.
[12] K. Khosla, R. Jones, and N. Bowman, "Featureless deep learning methods for automated key-term extraction," 2019.
[13] S. L. Pintea, P. S. Mettes, J. C. van Gemert, and A. W. Smeulders, "Featureless: Bypassing feature extraction in action categorization," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 196–200, IEEE, 2016.
[14] S. M. H. Rizvi, "Time series deep learning for robust steady-state load parameter estimation using 1d-cnn," *Arabian Journal for Science and Engineering*, vol. 47, no. 3, pp. 2731–2744, 2022.
[15] L. Liu and Y.-W. Si, "1d convolutional neural networks for chart pattern classification in financial time series," *The Journal of Supercomputing*, vol. 78, no. 12, pp. 14191–14214, 2022.
[16] S. M. Shahid, S. Ko, and S. Kwon, "Performance comparison of 1d and 2d convolutional neural networks for real-time classification of time series sensor data," in *2022 International Conference on Information Networking (ICOIN)*, pp. 507–511, IEEE, 2022.
[17] B. Jang, M. Kim, G. Harerimana, S.-u. Kang, and J. W. Kim, "Bi-lstm model to increase accuracy in text classification: Combining word2vec cnn and attention mechanism," *Applied Sciences*, vol. 10, no. 17, p. 5841, 2020.
[18] T. Szandała, "Review and comparison of commonly used activation functions for deep neural networks," *Bio-inspired neurocomputing*, pp. 203–224, 2021.
[19] T. M. Mitchell, *The discipline of machine learning*, vol. 9. Carnegie Mellon University, School of Computer Science, Machine Learning . . . , 2006.