Hindawi
Security and Communication Networks
Volume 2023, Article ID 2801421, 17 pages
https://doi.org/10.1155/2023/2801421

Research Article

IoT Security Detection Method Based on Multifeature and Multineural Network Fusion

Zihao Zhu, Leilei Zhang, Jianhua Liu, and Xianer Ying

Department of Computer Science and Engineering, Shaoxing University, Shaoxing 312000, China

Correspondence should be addressed to Jianhua Liu; ljh_541@163.com

Received 20 January 2023; Revised 6 May 2023; Accepted 11 May 2023; Published 29 May 2023

Academic Editor: Taimur Bakhshi

IoT security detection plays an important role in securing the IoT ecosystem. The current detection systems suffer from poor fault tolerance and inefficient detection results. To address the IoT security vulnerability, the paper designs a multifeature fusion-based IoT security detection model to simulate an attacker sending test commands to IoT nodes. Firstly, the data collection algorithm is introduced, and the collected dataset is analyzed by three neural network models, namely, RNN, LSTM, and GRU, respectively. The best scoring model is selected as the classifier to identify vulnerabilities and achieve IoT security detection.

1. Introduction

The Internet of Things (IoT) is a vast network that allows the exchange of information with anyone or anything at any time, place, or network. With the development of information technology and economic growth, the IoT industry is growing rapidly and more and more companies are using IoT and OT devices to improve productivity and increase the visibility of their operations. As a result, networked devices deployed on enterprise networks have access to sensitive data and critical systems.

The Internet of Things (IoT) is mainly used to replace manual work that is more complex, hazardous, or mechanical, and detection devices are mostly left unattended and monitored. This is compounded by the fact that sensing devices are multifunctional and simple, and different standards and protocols are used for different applications, making it impossible to use unified security techniques to protect against external attacks. This leaves these devices with numerous security issues that make them vulnerable to manipulation and sabotage by attackers, exposing them to data breaches, botnets, and other security threats and putting other parts of the organization at risk. For example, cyber threat actors often target unprotected printers, smart lighting, IP cameras, and other networked devices to gain access to an organization's network. From these devices, they can move laterally through the network to access more critical devices and sensitive data and create ransomware or double-ransom cyberattacks that can render corporate networks useless. IoT security is therefore essential to the company's growth.

The diversity and heterogeneity of the IoT make IoT system security different from traditional system security.

(1) Security risk of the architecture: The IoT cloud platform is essentially a PaaS, which will open some new ports and APIs and other services, and IaaS cloud security does not understand the use of these newly opened ports and services, so it is difficult to cover the security policy.

(2) Protocol security risks: IoT communication protocols such as ZigBee, Bluetooth, and NB-IOT are not used in traditional system security and therefore cannot be covered by the corresponding security policy.

(3) Security risks at the border: Unlike the traditional Internet, IoT is not a client/server model; it has no clear border, and traditional protective devices such as firewalls cannot be applied, while hackers can launch attacks directly on devices. The above three aspects show that traditional security policies are not

effective in maintaining IoT security. To address IoT security vulnerabilities, this paper designs a multifeature fusion-based IoT security detection model.

In terms of technical layers, IoT can be divided into three layers: the sensing layer, the network layer, and the application layer, each of which faces several security threats.

(1) Sensing layer: IoT senses a variety of objects, has a large demand for monitoring data, has complex and changing application scenarios, and is susceptible to natural damage or human destruction, resulting in nodes not working properly.

(2) Network layer: In the transmission of information, it is easy to have cross-network transmission and also easy to have security risks. The network layer is more vulnerable to security issues such as DDoS attacks, malicious data, and privacy breaches due to the large amounts of data and information.

(3) Application layer: The security threat facing the IoT application layer is mainly the false terminal trigger threat, where an attacker can send a false trigger message to the terminal via SMS to trigger terminal misoperation.

In terms of the types of protection, IoT security protection falls into three main categories. Network security: IoT network security implements a zero-trust security policy to minimize the attack surface of the enterprise. Embedded: This provides on-device security for IoT devices using an embedded system agent. Runtime protection monitors the device's current state and takes action to identify and remediate zero-day attacks based on anomalies. Firmware assessment: Firmware security starts with assessing the firmware of protected IoT devices. This can identify potential vulnerabilities in the firmware of IoT devices. The model provided in this paper focuses on the use of embedded protection. The model is mainly applied to the perception layer in the enterprise IoT through the detection IDS deployed in the enterprise, collecting data and passing it into model training to realize the network security awareness and evaluation of the IoT. IoT security is a product of the convergence of cyber security with other engineering disciplines and includes, in addition to traditional network data, server, network infrastructure, and information security, direct or distributed monitoring and control of the state of networked physical systems. Their security techniques can be broadly summarized as follows. (1) Cryptography can effectively guarantee the confidentiality and integrity of data. Although the Internet of Things is different from traditional networks, it is still dependent on the Internet, so traditional password protection techniques are still applicable. Cryptography-based security reinforcement of authentication and encryption mechanisms can achieve data transmission availability, confidentiality, integrity, nonrepudiation, and data freshness. (2) Authentication and access control technology: The use of authentication and access control technology can effectively suppress the security problems of terminal equipment being controlled and information being tampered with. The real identity of the user is confirmed through a preset authentication method, and access to the sensing node is legally detected and authenticated to achieve the role of protecting the security of the node. For IoT access control technology, it is difficult to implement the flexible access control policy of traditional role-based networks, and the security of IoT transmission and storage information can be improved by setting the access control policy prohibiting access to unauthorized resources. (3) Network security access technology IoT routing typically requires routing across multiple networks with IP-based routing protocols and identity-based routing algorithms. The routing problem of multiple network convergence ultimately maps endpoint identities to IP addresses to achieve unified IP-based routing. The use of sound routing control technology can reduce the security threat posed to IoT devices by illegal device intrusion systems and create a barrier for illegal intrusion users. In the IoT perception layer, intruders cannot continue to obtain data from routing nodes. For routing security control, lightweight cryptographic algorithms, cryptographic protocols, and secure encryption are commonly used today. (4) Use of hardware protection measures. Using secure hardware protection mechanisms to resist reverse analysis and unauthorized access is one of the most effective security controls for the IoT. For example, using ARM Trust Zone technology, secure drivers are developed to run on top of a streamlined microkernel in a trusted environment, enabling interaction between trusted and untrusted environments through commands provided by an API. IoT security detection plays an important role in securing the IoT ecosystem, and current detection systems suffer from poor fault tolerance and inefficient detection results. This paper adopts a longitudinal structure, following the structure of introduction, body, and conclusion, to present the current IoT security vulnerability problem, analyze where the problem lies, and then propose a multifeature fusion-based IoT security detection model to simulate an attacker sending test commands to IoT nodes. The model data collection algorithm is introduced in this paper. The collected data set is passed into three neural network models, namely, RNN, LSTM, and GRU, for analysis through feature processing, and the best scoring model is selected as the classifier to find the attack vulnerability features and achieve the purpose of IoT security detection. The model proposed in this paper can better solve the current problem that neural networks cannot detect the security status of the IoT in real time and accurately detect the security status of the entire IoT.

2. Related Works

Internet security plays an important role in national security and related security issues. Therefore, the detection of problems that threaten the security of the Internet through scientific and effective means is an important factor in maintaining national security. According to different ways to suit computing resources, malicious mining can be divided into malicious mining for browsers and malicious mining for hosts. Reference [1] addresses the lack of

a dedicated detection and forensic integration tool for malicious web pages and proposes a method for detecting malicious mining web pages and forensic research based on multifeature recognition. The detection model can detect and classify the URL and the forensic method of forensics from three layers: plane layer, code layer, and network data layer, to ensure the trustworthiness of the data. Identity authentication is an important means of securing user information on the Internet and has received widespread attention as a digital cryptocurrency-Bitcoin. The authors propose a decentralized identity authentication system based on the Ethernet blockchain as a platform, using smart contracts to realize a decentralized identity authentication system that allows the management of multiple types of user identity information [2]. Bitcoin mining can be divided into four steps: generating the initial transaction, forming the block header from the root hash and other fields, calculating the workload, and authenticating the workload. Reference [3] combines the advantages of traditional heuristic address clustering methods and machine learning-based address classification methods to propose a new Bitcoin address classification model based on machine learning. The authors in [4, 5] propose a study on malicious code detection and analysis techniques based on feature modeling, including a study on malicious code detection and analysis techniques based on feature modeling and a web shell detection and analysis model based on the distance synthesis evaluation method. In [6, 7], the authors propose a traceable blockchain anonymous transaction model and a blockchain identity privacy protection scheme based on ring signatures under this model. According to the different mining languages [8], the author proposes a variety of machine learning-based JavaScript malicious code detection methods by analyzing the characteristics of the JavaScript language, which improves the detection accuracy of the JavaScript malicious code, and also extracts and categorizes the features of obfuscated malicious JavaScript code, from attribute-based features, redirect-based features, suspicious keyword-based features, and obfuscated features [9]. The features are extracted from four main categories, and the dataset is evaluated by anomaly detection patterns using various supervised machine learning algorithms. The authors fuse web mining features and train them using four machine algorithm models—logistic regression, support vector machine, decision tree, and random forest—while extracting Websocket, Web Worker, Postmessage and message events, Messageloop events, CPU usage, keyword matching, computational functions, and dynamic functions to train the model, and the first three features were found to be the highest identifiers for identifying malicious mining sites [10]. In general, there are two forms of illegal mining, browser-based mining and binary-based malicious mining code, with the latter being stealthier and harder to detect. Of all the dynamic features used to detect malicious mining, CPU events are the most commonly used features. In [11], the authors propose a hardware-assisted analysis-based approach to malicious mining detection based on internal CPU information recorded by a counter during program execution, as a feature reflecting program behavior. In addition, an analyzer is proposed in [12] for hash functions and a stack structure-based analyzer to monitor the frequency of occurrence of hash algorithms commonly used for malicious mining. Accordingly, MineThrottle is proposed in [13] to design a detection tool based on dynamic code stacking. With a dynamically inserted performance counter, MineThrottle can accurately analyze the CPU usage of different code blocks at runtime to detect malicious mining. The authors [14] propose a virtual machine introspection (VMIA) based tool to detect malicious mining in the same operating system, where mining software can easily bypass the detection system and where there is currently insufficient protection against the widespread binary mining software (VMI) approach for malware detection and defense. In terms of neural networks, the authors [15] use a combination of a stacked autoencoder (SAE) and convolutional neural network (CNN) approach to classify network data for traffic feature classification and application type identification. An approach to network intrusion detection using recurrent neural networks and attention mechanisms has also been proposed in [16], which uses LSTM networks for feature extraction over long-time sequences and introduces attention mechanisms to improve the detection effectiveness of network intrusions. In terms of detection engine products for practical protection, the author examines more than 50 detection engines on the market and concludes that the implementation of static feature-based detection engineering leaves much to be desired [17]. A CNN detection model is obtained in [18] based on the opcode obtained by compiling PHP files, using a glossary model to extract word order features, and training to obtain a method that outperforms traditional machine learning algorithms in accuracy, recall, and F1 values. In virtual currency mining, the eclipse attack [19] and routing attack [20] are the main attacks against Bitcoin P2P networks. An eclipse attack is a network attack on a specific node or cluster of nodes, causing the network to "partition" for purposes such as "double-spending." Routing attacks are attacks where an attacker controls the routing infrastructure to attack the network based on a vulnerability, intending to partition the Bitcoin network (partitioning attacks) or delay the propagation of new blocks to specific Bitcoin nodes (delay attacks). These network attacks can lead to a waste of mining arithmetic resources and increase the likelihood of duplicate payments. In terms of feature analysis, the feature filtering is used in [21, 22] to reduce the dimensionality of features based on both forward-increasing features and reverse-decreasing features, improving the efficiency of the model and the accuracy of the prediction. In [23], a dimensionality reduction method based on feature similarity in training data is proposed, which can reduce computational complexity and improve prediction accuracy in coarse-grained software defect prediction. In terms of DNS detection, the author [24] uses the characteristics of DNS accesses in lateral movement and data return phases to extract hidden features in DNS traffic through a machine learning model using a semi-supervised approach, which in turn analyses the behavior of suspected APT attacks. The researchers also used DNS traffic decision tree modeling in conjunction with

disclosed threat intelligence and used the results to feed into a local threat intelligence repository as a way to subsequently reduce false alarm rates [25]. A comparison of this model detection method with other model detection methods is shown in Table 1.

IoT nodes are vulnerable to persistent attacks and can form botnets between different IoT nodes, making it difficult to derive IoT security status just by analyzing individual nodes or features. Machine learning detection techniques currently available on the market can only distinguish whether each node is secure or not, but this approach cannot capture the entire IoT security state. For this reason, we chose recurrent neural networks as the research direction, deploying ids to continuously detect the entire IoT and feature the collected data to classify the IoT security status dataset using three classifiers, namely, long short term memory network (LSTM), recurrent neural network (RNN), and gate gap unit network (GRU), which were trained to obtain three classifiers. Good classifiers and classification results were evaluated using the AUC. The existing method is unable to dynamically detect the security status of the IoT and uses multinetwork training to improve detection accuracy and achieve real-time dynamic and accurate detection of network information.

3. IoT Security Detection Model

3.1. Principle of the Attack. Malicious attackers generally use vulnerabilities in network security to analyze and exploit these vulnerabilities to attack. The vulnerabilities that are attacked are divided into seven main types, namely: NE (network information encryption), network authentication mechanism, network access control, key security level, network node authenticity, network node nonrepudiation, and network identification of foreign routing mechanisms.

Attackers typically use attack command frames to exploit vulnerabilities in IoT devices, and if IoT has not deployed vulnerability detection defenses, the attacker can exploit the vulnerability directly (Figure 1).

An IoT deployed with a vulnerability detection node can detect anomalies and scan for vulnerabilities when an attacker conducts an IoT vulnerability attack, submitting vulnerability information for administrators to fix (Figure 2).

IoTs that have deployed IoT vulnerability detection mechanisms can continuously fix existing vulnerabilities to defend against the same or similar vulnerability attacks, while IoTs that do not deploy detection mechanisms are often vulnerable to similar attacks and incur some unnecessary losses.

3.2. Analysis of IoT Attack Protocols. The attack command is usually delivered in the form of a data frame, which is formatted as shown in the diagram and consists of two main parts: the frame header and the command frame. The header holds the basic information of the frame, including the control field (which holds the basic information of the frame), sequence number, destination address, and short source address. The command frame is responsible for storing command identifiers and data requests, where command identifiers are strings used to identify a specified object to perform a certain action, and data requests are used to pass data or call a function. Table 2 shows the structure of the attack frame.

3.3. Feature Engineering. The dataset used in this paper returns test data by simulating an attacker sending test commands to IoT nodes, determining whether the network is encrypted, whether there is an authentication mechanism, whether the key is leaked, and whether the network can identify real nodes, detecting nonrepudiation of network node communication, and detecting whether the network can identify foreign nodes after returning test data, based on whether there is a vulnerability, with 1 indicating yes and 0 indicating no, and labeled features to establish a feature dataset with 510,000 training data and 510,000 test data. The program processes these data by reading the dataset and decoding it. The flowchart of data collection is shown in Figure 3.

3.4. Data Acquisition Algorithms. The simulated attack device uses 14 microcontrollers as virtual attack devices and 2 microcontrollers as virtual IoT nodes to simulate the attack through IoT nodes. The attack is carried out by sending test commands, and the test results are sent back to the device. Therefore, a custom communication protocol is required to enable communication between the microcontrollers and the device.

Receiving data are achieved by setting the microcontroller's serial port. Due to the limitation of receiving data on the microcontroller (only 1 byte of data can be received at a time), it is necessary to set a receive interrupt and set the end character in the data frame. When sending command frames, similar to receiving, only 1 byte of data can be sent at a time, so when sending data, write the data to the queue and always poll the queue to ensure that the data are sent in full. When the device receives data from the microcontroller, it determines whether there is a vulnerability based on the data in the data frame and characterizes it.

The custom data structure is given in Appendix.

3.5. Detection Methods for Multifeature Fusion. First, the original dataset is self-guided several times at a certain sampling rate to generate subsets containing different samples and data distributions. We then train the base selector on each sampled subset. Due to the inherent diversity of the different subsets, different base selectors are trained with different results, and the feature importance of each base selector will be different. In other words, the ranking of feature importance will be different for each base selector. Then, we accumulate the feature importance obtained from each base selector to obtain the final feature importance. Finally, we use a forward search method to select the best combination of features.

Table 1: Comparison of this model detection method with other model detection methods.

Papers	Scenarios	Model type	Advances
Awajan [26]	Neural network-based IoT detection	DNN-based	(i) Get dynamic access to IoT security data (ii) Improved accuracy
Rami Reddy et al. [27]	Feature selection based on an improved version of the grey wolf algorithm	ECHIGWO-based algorithm	(i) Forward search to derive the optimal feature subset (ii) Better optimization in class II and class V models
Bhandari et al. [28]	Artificial intelligence-based malware discovery framework	DNN-based ml model	(i) Capture dynamic IoT security status data to proactively uncover vulnerabilities
Thandapani et al. [29]	AI-based IoMT	CNN-based	(i) Multimodel training to improve feature accuracy
Kalutharage et al. [30]	Interpretable artificial intelligence (XAI) detection methods	Based on the automatic coding model	(i) Save data collection costs by capturing dynamic data with distributed deployment ids (ii) Detecting diverse IoT security threats
Alzahrani and Alzahrani [31]	IoT anomaly mitigation system based on a multialgorithm model	Three algorithms based on ML models (KNN, EWMA, and CUSUM)	(i) Always acquire IoT data and proactively discover vulnerabilities (ii) Streamlined model deployment costs

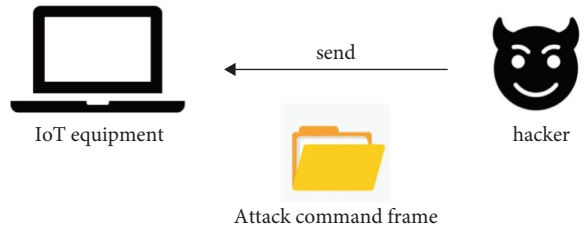

Figure 1: The process of an attacker's vulnerability attack on Internet of Things devices without a detection mechanism.

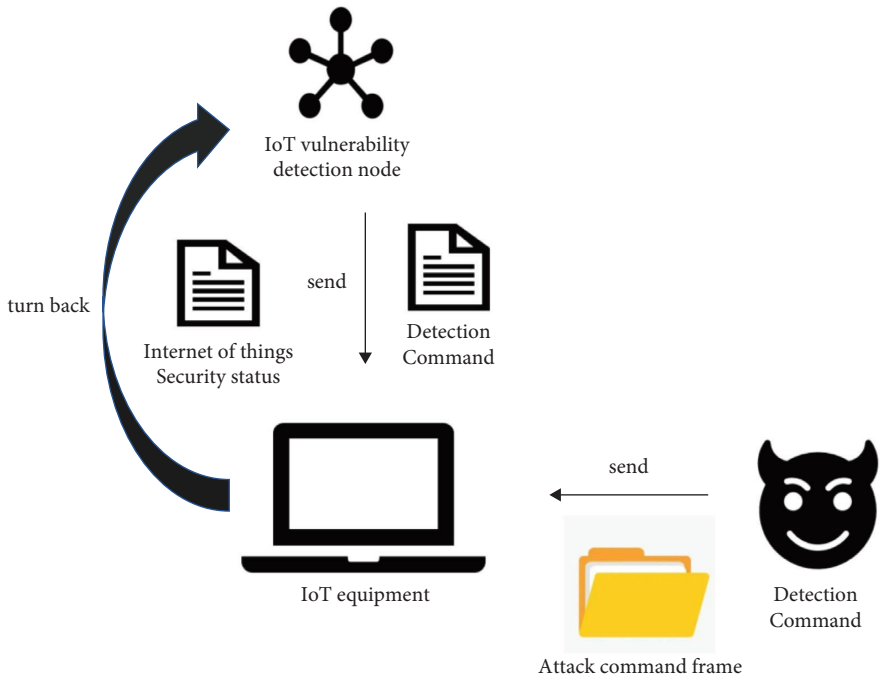

Figure 2: Process for attackers to exploit the Internet of Things deployed with vulnerability detection.

Table 2: Attack frame structure.

Frame header	Control domain	Frame type Security controls Frame unprocessed marker Request a confirmation mark Internal pan marking Reserved bits Destination address mode Protocol version Source address mode Serial number Purpose pan id Destination short address Source short address
Command frame		Command identifiers Data requests

In the forward search process, the features are first sorted in descending order of importance to ensure that the most important features appear in the first position. The sorting process is necessary because the important final features are unordered, which prevents the following operation. The importance of features is then accumulated from the first (most important) to the last (least important). When the accumulated value is greater than a predetermined threshold, the forward search process stops. At this point, the features are combined into the optimal subset.

The classification results were then processed to obtain multiple features based on fully exploiting the security vulnerabilities in IoT operations. To compensate for the inability of existing methods to detect IoT security status, detection accuracy was improved using multinetwork such as long short term memory (LSTM) network, recurrent neural network (RNN), and gate empty unit (GRU), respectively. Multinetwork training is used, and the network results with the highest accuracy are used as the classification network. Based on the results, vulnerabilities are proactively discovered, and the current state of IoT security is assessed and analyzed.

4. Neural Network Detection Models

4.1. Principle of Neurons. The core component of an artificial neural network is the artificial neuron. Each neuron receives inputs from several other neurons, multiplies them by the assigned weights, sums them up, and passes the sum to one or more neurons.

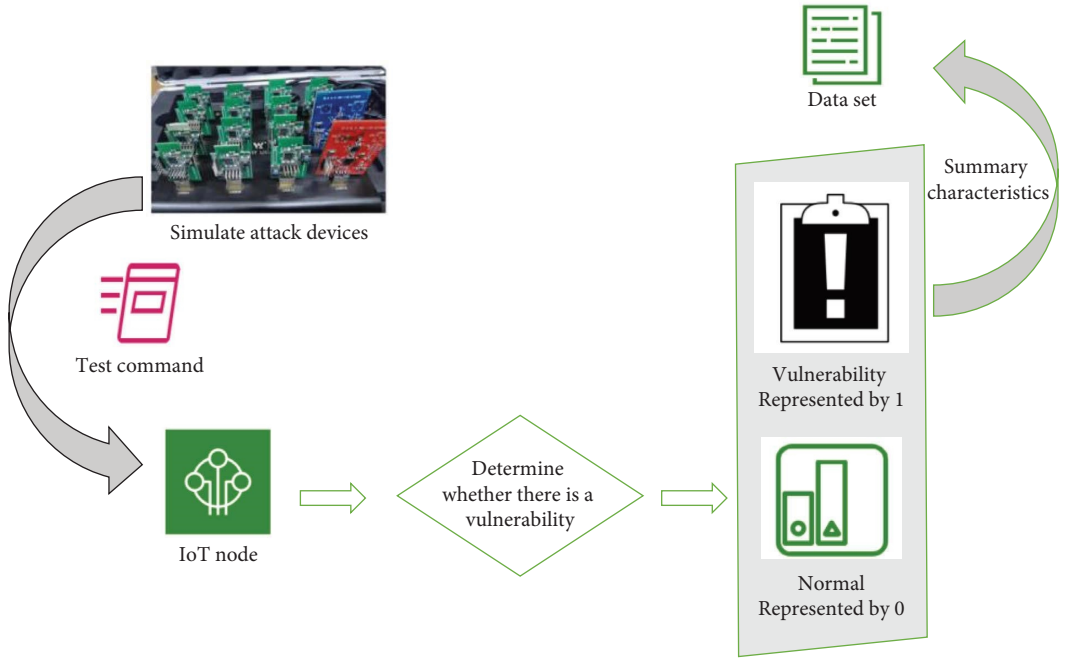

Figure 3: Data collection process.

Scientists have proposed an artificial neural model (M-P model) based on the study of neurons in the brain. The artificial neural model is shown in Figure 4.

The M-P model treats neurons as n inputs corresponding to the production of 1 output, and the expression for the function of this model is as follows:

$$y = f\left(\sum_{i=1}^{n} W_i X_i - b\right). \tag{1}$$

A BP network is a hierarchical neural network with 3 or more layers in Figure 5. Each neuron in the upper and lower layers is fully connected, i.e. At its core, it uses the forward conduction formula to calculate the output value of the nth layer, performs error analysis based on residuals between the output value and the actual value, and then modifies weights and thresholds to obtain a model in which the output is consistent with the expected result.

To capture the error loss, the neural network uses forward propagation. In a forward neural network, only the neurons between two adjacent layers are connected; there is no feedback between the other neurons. Each neuron in Figure 6 can receive multiple inputs from the previous layer and output only one to the neuron in the next layer.

$$h = f\left(\sum_{i=1}^{n} x_j \cdot w_i + biais\right). \tag{2}$$

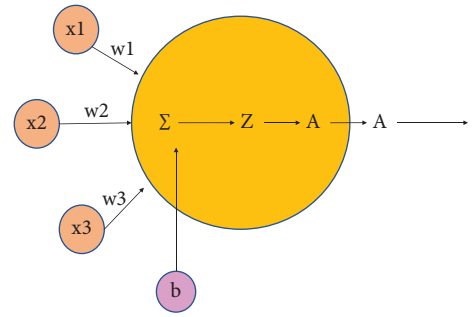

Figure 4: Artificial neuron model.

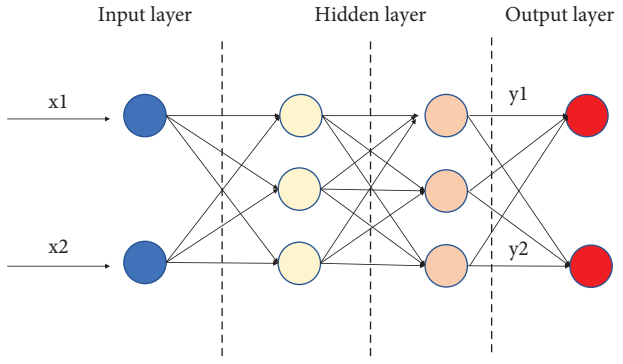

Figure 5: Neural network model diagram.

4.2. IoT Vulnerability Detection Model. The model is trained using three neural models, RNN, LSTM, and GRU, and the dataset is self-guided several times at a certain sampling rate to generate different subsets. Base selectors (i.e., LightGBMs) are trained on each sampled subset due to the inherent diversity of different subsets, different base selectors are trained with different results, and the feature importance of each base selector will be different. A forward search method is used to select the optimal combination of features. Finally, the subsets are trained using these three neural network models to obtain different results, and the classifier with the

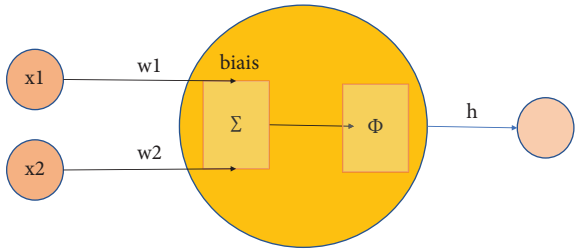

Figure 6: Generalisation of neurons.

best results is selected and used by comparing the results through AUC evaluation. The flowchart to filter the best classifier process is shown in Figure 7.

4.3. Recurrent Neural Network (RNN). RNNs are recurrent neural networks that process sequential data so that the current output of a sequence is also related to the previous output, as shown by the network remembering the previous information and applying it to the current output. In theory, RNNs can process data of any length. An important feature is the sharing of parameters at each step.

In the clockwise RNN of Figure 8, X_t is the input at moment t, a vector of $[x_0, x_1, x_2, \ldots, x_n]$; U is the weight matrix from the input layer to the hidden layer; S_t is the value of the hidden layer at moment t; W is the weight matrix when the value of the hidden layer at the previous moment is passed to the hidden layer at the next moment; and V is the weight matrix from the hidden layer to the output layer. It is the output of the RNN network at the moment t. The value of the hidden layer can be calculated according to the formula. After calculating the hidden layer value at the time t, the output layer value is then calculated.

$$\begin{aligned} &\mathrm{St} = \mathrm{f}\left(\mathrm{U} \cdot \mathrm{Xt} + \mathrm{W} \cdot \mathrm{St} - 1 + \mathrm{b}\right) \text{hidden layer formulae},\\ &\mathrm{Ot} = \mathrm{g}\left(\mathrm{V} \cdot \mathrm{St}\right) \text{output layer formulae}. \end{aligned} \tag{3}$$

4.4. LSTM Neural Network. The LSTM is designed to be a memory cell with selective memory, which can selectively remember important information and filter noisy information to reduce memory burden whereas repetitive modules in RNNs have a chained form with a very simple structure and repetitive modules in LSTMs have different structural interactions (Figure 9).

In Figure 10, the LSTM has three types of gate structure: forget/forget gates, input gates, and output gates, to protect and control the cell state. 0 means "no amount allowed to pass" and 1 means "any amount allowed to pass."

The function of the forget gate is to decide what information to discard from the cell state. It reads the previous output h_{t-1} and the current input x_t, does a Sigmoid nonlinear mapping, then outputs a vector f_t, and finally multiplies it with the cell state C_{t-1}.

$$f_t = \sigma\left(w_f \cdot \left[h_{t-1}, x_t\right] + b_f\right). \tag{4}$$

In Figure 11, the input gate identifies the new information that is stored in the cell state. In the formula, it can be seen as input information saved to the cell state and g_t can be seen as new information added to the cell state.

$$\begin{aligned} i_t &= \sigma\left(w_f \cdot \left[h_{t-1}, x_t\right] + b_i\right),\\ \widetilde{c}_t &= \tan h\left(w_c \cdot \left[h_{t-1}, x_t\right] + b_c\right). \end{aligned} \tag{5}$$

The output gate is based on the cell state, and thus the output value is determined in Figure 12. The sigmoid layer is first run to determine the portion of the output that is based on the cell state and then processed by tanh and multiplied by the output value to obtain the value that determines the output.

$$\begin{aligned} O_t &= \sigma, W_0\left[h_{t-1}, X_t\right] + b_0,\\ h_t &= O_\Sigma \cdot \tan h\left(c_t\right). \end{aligned} \tag{6}$$

4.5. GRU Neural Networks. GRU is a simplified and efficient variant of LSTM with fewer parameters and faster speed, which can solve the long dependency problem and reduce the risk of overfitting in RNN networks (Figure 13).

z_t and r_t in the equation represent the update gate and the reset gate, respectively. The update gate is used to control how much information from the previous state is brought into the current state. The larger the value of the update gate is, the more information from the previous state is brought in. The reset gate controls how much information from the previous state is written to the current candidate set $\widetilde{h}_t$, h_t. The smaller the reset gate is, the less information from the previous state is written. The description of the GRU formula is given as follows:

$$\begin{aligned} z_t &= \sigma\left(w_z \cdot \left[h_{t-1}, x_t\right]\right),\\ r_t &= \sigma\left(W_r \cdot \left[h_{t-1}, X_t\right]\right),\\ \widetilde{h}_t &= \tan h\left(W \cdot \left[r_t \cdot h_{t-1}, x_t\right]\right),\\ h_t &= \left(1 - z_t\right) \cdot h_{t-1} + z_t \cdot \widetilde{h}_t. \end{aligned} \tag{7}$$

4.6. Description of the Parametric Algorithm

4.6.1. Loss Function. The MSE has a smooth, continuous function curve that is derivable everywhere, facilitating the use of gradient descent algorithms, which are commonly used for loss functions. Moreover, as the error decreases, so does the gradient, which facilitates convergence to a minimum relatively quickly, even when using a fixed learning rate.

$$\mathrm{MSE} = \frac{\sum_{i=1}^{n}\left(f\left(x\right) - y\right)^2}{n}. \tag{8}$$

4.6.2. Activation Functions. Softmax often acts as an activation function in the output layer of a multiclassification neural network, mapping the output layer value to the 0-1 interval through the activation function and constructing

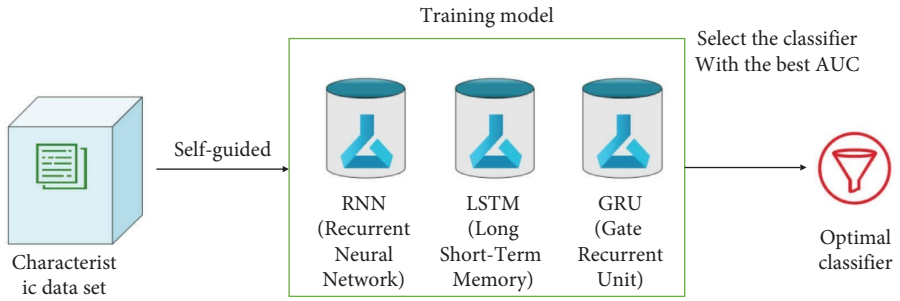

Figure 7: Filter the best classifier process.

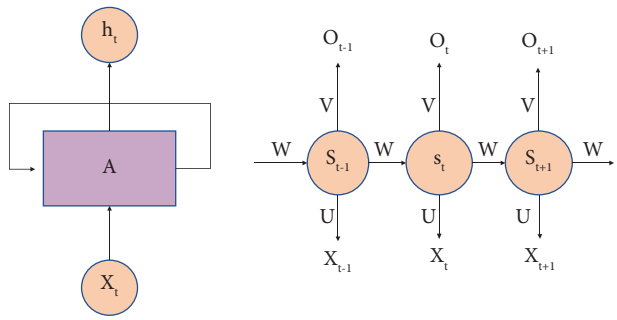

Figure 8: RNN structure diagram unfolding by timeline.

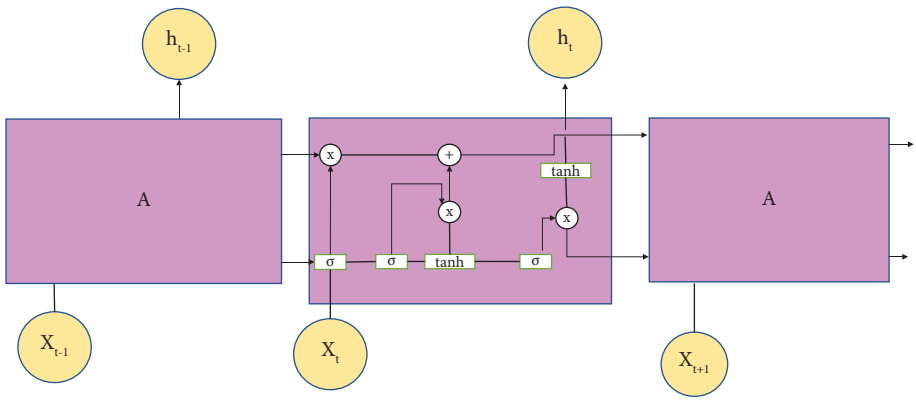

Figure 9: LSTM structure.

the neural output as a probability distribution, which is used in multiclassification problems. The larger the value of Softmax activation function mapping, the greater the probability of the true category, and the formula for calculating Softmax function is given as follows:

$$a_j = \frac{e^{zj}}{\sum_k e^{z_k}}. \tag{9}$$

4.6.3. Adaptive Learning Rate Optimization Algorithm. The RMSprop algorithm, known as root mean square prop, uses a differential square weighted average for the gradients of weights W and bias b to further optimize the problem of excessive oscillations in the loss function in updates and to further accelerate the convergence of the function.

$$\begin{aligned} S_{dw} &= \beta S_{dw} + (1-\beta)dW^2, \\ S_{db} &= \beta S_{db} + (1-\beta)db^2, \\ W &:= w - \alpha \frac{dw}{\sqrt{S_{dw}}}, \\ b &:= b - \alpha \frac{db}{\sqrt{S_{db}}}. \end{aligned} \tag{10}$$

In the abovementioned formula, S_{dw} and S_{db} are the gradient momentum accumulated by the loss function in the previous $t-1$ iteration and β is an index of gradient accumulation. The difference is that the RMSProp algorithm calculates the differential square weighted average for the gradient. This method is beneficial to eliminate directions

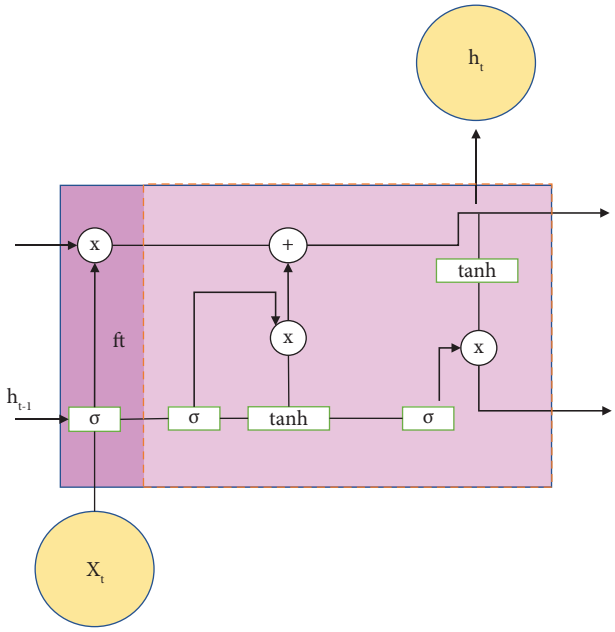

Figure 10: Forget the door process.

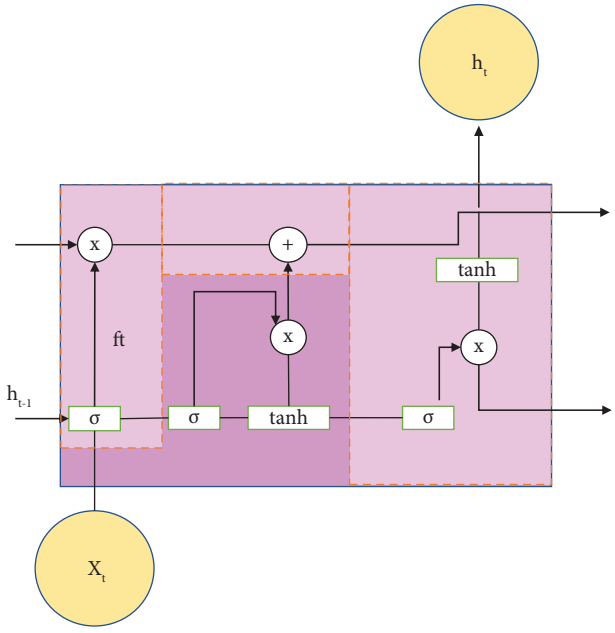

Figure 11: Input gate process.

with a large swing amplitude and is used to correct the swing amplitude so that the swing amplitude of each dimension is small. On the other hand, it also makes the network function to converge faster. To prevent the denominator from being zero, a small epsilon value is used for smoothing, which is generally taken as the power of 10 to 8. RMSprop is very similar to Momentum in that it eliminates oscillations in gradient descent, including mini-batch gradient descent, and allows you to use a larger learning rate, thus speeding up the learning of algorithms.

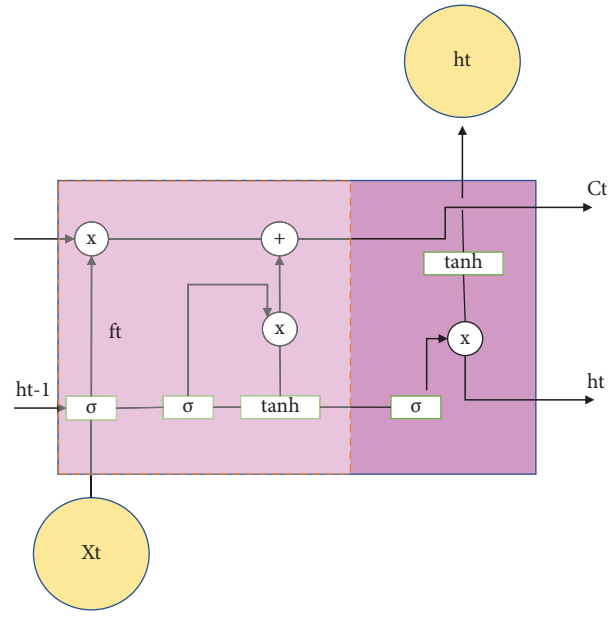

Figure 12: Output gate process.

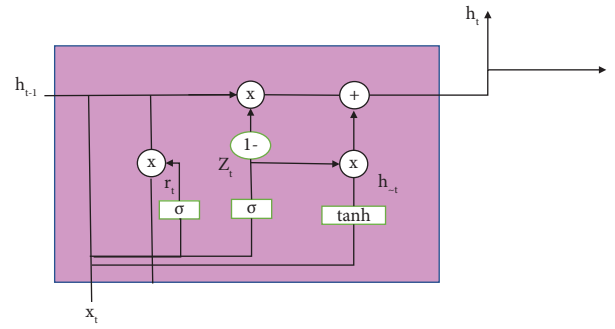

Figure 13: GRU model.

4.7. Evaluation Index. This section describes some of the evaluation metrics used in this experiment to determine the quality of feature vectors and classifiers. It can be argued that if multiple sets of feature vectors are fed into multiple identical classifiers, the better the classifier is, the better the feature representation of the feature vectors is. Thus, we can obtain an evaluation of the feature vectors by evaluating the index of classification effectiveness. First, we need to know the following basic concepts:

True positive (TP): The presence of a vulnerability in the IoT is correctly detected, i.e., the test result is correct and the result is positive.

False positive (FP): It is the false detection of an IoT vulnerability, i.e., incorrect detection and a positive result.

True negative (TN): It is correct detection of a normal IoT status, i.e., the detection is correct and the result is negative.

Table 3: RNN neural network structure.

Layer (types)	Output shape	Param
Simple_rnn_1 (SimpleRNN)	(None, 7, 128)	16640
dropout_1 (Dropout)	(None, 7, 128)	0
Simple_rnn_2 (SimpleRNN)	(None, 128)	32896
dropout_2 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 8)	1032

Table 4: LSTM neural network structure.

Layer (types)	Output shape	Param
Lstm_1 (LSTM)	(None, 7, 128)	66560
dropout_1 (Dropout)	(None, 7, 128)	0
Lstm_2 (LSTM)	(None, 128)	131584
dropout_2 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 8)	1032

Table 5: GRU neural network structure.

Layer (types)	Output shape	Param
Gru_1 (GRU)	(None, 7, 128)	49920
dropout_1 (Dropout)	(None, 7, 128)	0
Gru_2 (GRU)	(None, 128)	98688
dropout_2 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 8)	1032

False negative (FN): It is false detection of a vulnerability in the IoT, i.e., incorrect detection and a negative result.

Precision: It is the proportion of correct predictions out of the total number of positive predictions. The higher the precision, the better the expression of the feature vector.

Precision is expressed as follows:

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{11}$$

Accuracy: It is the proportion of correct predictions out of the total number of predictions. The higher the accuracy, the better the accuracy of the feature vector. The calculation formula of accuracy is as follows:

$$a = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}. \tag{12}$$

True positive rate (TPR): It means the number of true positive samples detected divided by the number of all true positive samples, calculated as follows:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{13}$$

False positive rate (FPR): It means the number of false positive samples detected divided by the number of all true negative samples. The formula is as follows:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}. \tag{14}$$

4.8. Simulation Training

4.8.1. Training Parameter Settings. epochs (number of iterations): 600, train_data (amount of training data): 514,021

loss_fct (loss function): mse, optimizer (optimization algorithm): rmsprop

activation_fct (activation function): softmax

4.8.2. Training Parameters Setting Basis. The number of iterations is 600; when the number of iterations is 600, the accuracy of the results is good; the amount of training data is 514,021, and when the amount of training data is small, it is easy to overfit (the accuracy of the model gets better and better on the training set but worse and worse on the test set); the model learns too many irrelevant features and is not able to produce results.

The loss function I chose was mse. I tried other more common loss functions such as mae before choosing the mse function, but the model was overfitted, and I finally chose the mse function as the loss function. RMSProp gives better results under nonconvex conditions by adding a decay factor to control how much historical information is obtained. rmsprop makes more progress in directions where the parameter space is flatter (because of the flatness, the sum of squared historical gradients is smaller, corresponding to a smaller learning decline) and can make steep directions flatter, thus speeding up training. For the choice of activation function, I had previously used the sigmoid function, but these functions are mostly used in binary classification problems and do not meet the

Table 6: Evaluating indicator.

Eigenvalue	Evaluation indicators	RNN (recurrent neural network) (%)	LSTM (Long short-term memory) neural network (%)	GRU (gated neural network) (%)
Normal (normal)	TPR (true positive rate)	99.785652	99.70653	99.785632
	FPR (false positive rate)	0.02629	0.02623	0.02645
	Precision	99.924538	99.536	99.924538
Key (whether the key is compromised)	TPR (true positive rate)	55.706935	55.84048	55.825778
	FPR (false positive rate)	0.24754	0.0365363	0.365378
	Precision	97.470417	96.322675	96.321742
FR (spoofing the router to join the network)	TPR (true positive rate)	94.716305	95.237292	94.703649
	FPR (false positive rate)	0.2544174	0.2869203	2.533927
	Precision	77.965484	75.980211	78.030935
Authentication (presence or absence of authentication mechanisms)	TPR (true positive rate)	97.770954	97.751925	97.995848
	FPR (false positive rate)	0.2576831	0.2512793	2.889814
	Precision	89.793535	90.01485	88.742412
NE (network data encryption status)	TPR (true positive rate)	96.295142	96.866101	97.52915
	FPR (false positive rate)	0.2466249	0.2996873	3.521875
	Precision	84.666841	82.114505	79.797605
ACL (access control)	TPR (true positive rate)	80.545442	80.666809	82.123224
	FPR (false positive rate)	0.079273	0.010239	0.35294
	Precision	96.534611	95.576045	86.475718
NODE (identify foreign nodes)	TPR (true positive rate)	42.371681	8.106195	14.584071
	FPR (false positive rate)	1.402514	0.0259163	0.452776
	Precision	14.449541	14.764668	15.158205
NOR (detects nonrepudiation of network node communications)	TPR (true positive rate)	25.150763	31.997162	1.631784
	FPR (false positive rate)	0.795674	0.104477	0.054798
	Precision	14.926316	14.55543	14.110429
Aggregate	acc (accuracy)	90.137387	90.13333	90.1364139

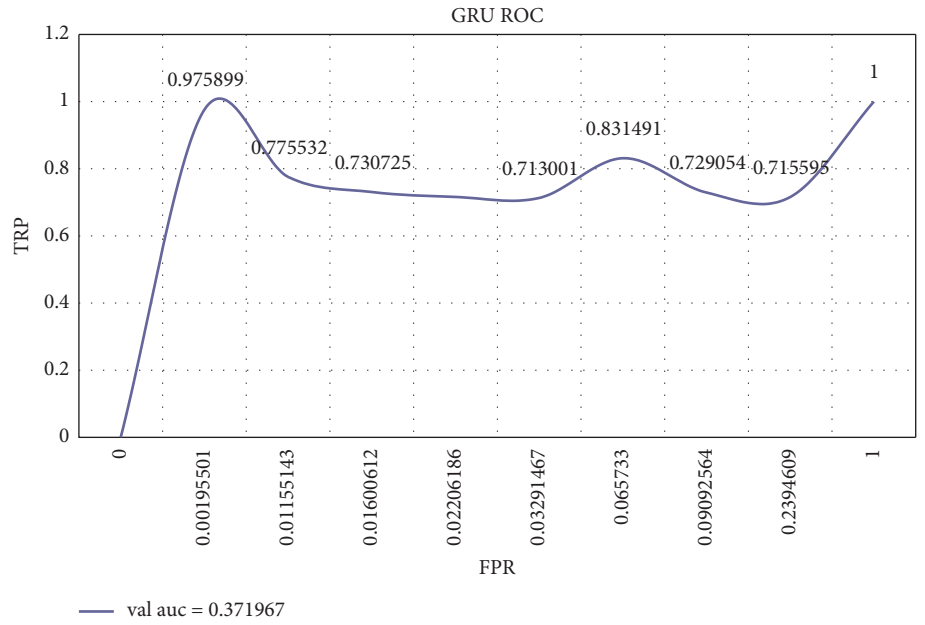

Figure 14: ROC curve of the GRU model.

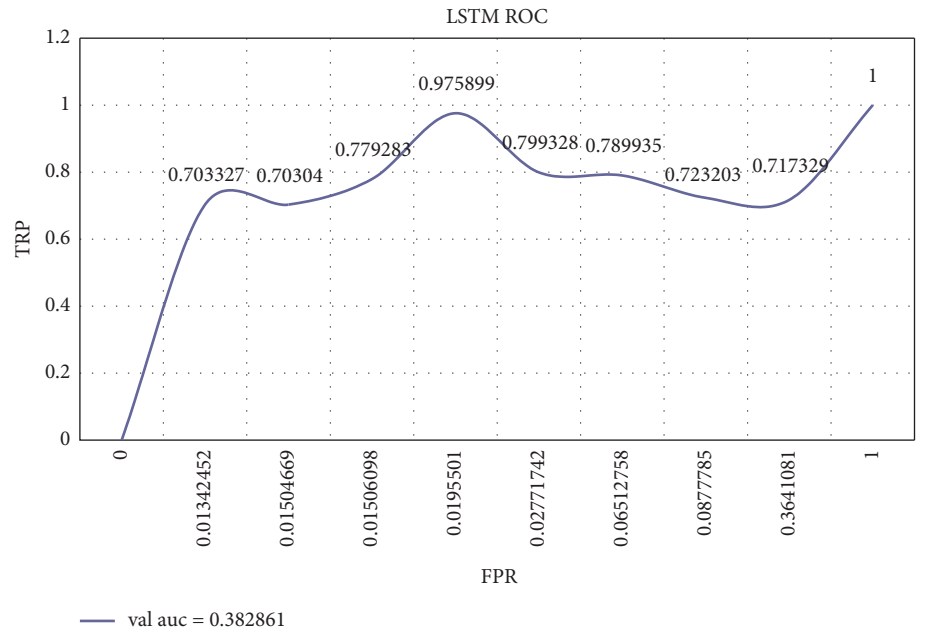

Figure 15: ROC curve of the LSTM model.

requirements of this model, so I changed the activation function to the softmax function, which is commonly used in multiclassification problems, in order to get the best results.

4.8.3. Neural Network Model Structure. The hidden layer is made up of two layers, each containing 128 neurons. Neural network structures are shown in Tables 3–5.

In Table 6, evaluation metrics derived from using different neural network models in training.

The ROC curves of GRU, LSTM, and RNN models are shown in Figures 14–16, respectively.

The area under curve (AUC) and accuracy (ACC) as the key metric show that the three models have similar AUC scores for detecting IoT security vulnerabilities, with the GRU model having the best score, so we chose the GRU model as the classifier and used its results in this training.

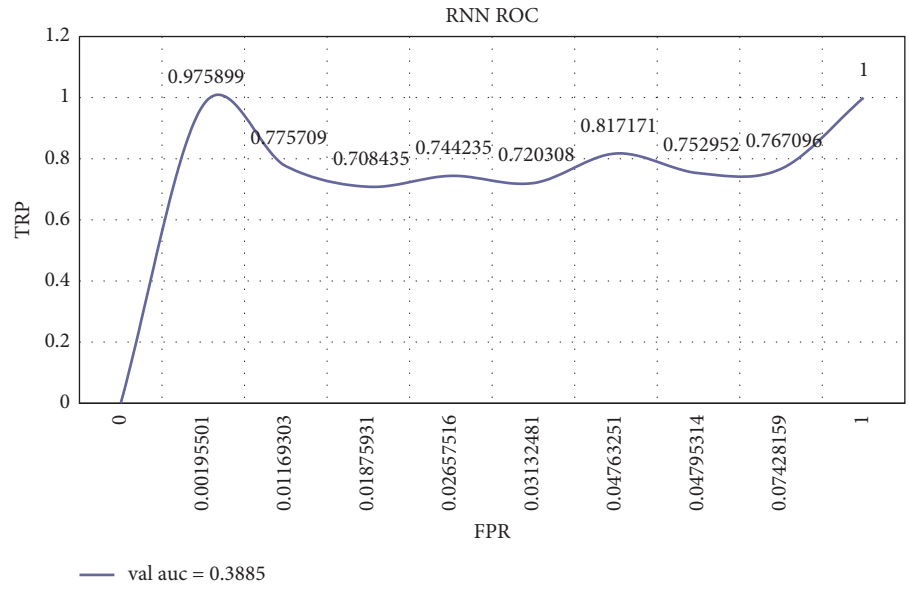

Figure 16: ROC curve of the RNN model.

5. Conclusion

With the rapid development of the Internet, the value of Bitcoin is increasing day by day, and the malicious mining behaviors associated with it are also increasing. Every year, a large number of servers in the world are attacked by mining, and IoT security has become a key concern for various countries. In this paper, starting from the analysis of the malicious frame infection process, an IoT security detection model based on multifeature fusion is used to simulate the process of attackers sending test commands to IoT nodes and data collection. The dataset is analyzed by neural network models RNN, LSTM, and GRU, and the best scoring model is selected as a classifier to find attack vulnerability features as a way to meet IoT security detection goals, such as network security, authentication, access control, key security level, node attack protection, nonrepudiation, and IoT routing attack protection.

Experimental results show that the key index AUC, ACC scores best under different neural network models using the RNN neural network model. The IoT security detection model based on multifeature fusion proposed in this paper has a good effect on IoT security detection. The model uses some evaluation indicators to judge the quality of feature vectors and classifiers, and the evaluation of feature vectors is obtained by evaluating the indices of classification effects, such as true positive, false positive, true negative, and false negative. The implementation is simple, and the detection efficiency of the model is high. The trained model is relatively small in size and easy to run and implement, making it easy to apply in practical application scenarios.

Appendix

The custom data structure is as follows:

Data header (2 bytes) + Data length (1 byte) + Function code + Data + Checksum (CRC16-MODBUS).

Custom protocol process is as follows:

(1) Initialize the microcontroller in the simulated attack device according to the frame structure, and initialize the command code at the same time:

```
Set the command code
#define CMD_READREG 0x01
#define CMD_WRITEDREG 0x02
#define CMD_CONFIGURE 0x03
#define CMD_IAP 0x04
Protocol Related
#define FRAME_LEN_POS 2//Data frame length index
#define FRAME_CMD_POS 3//Command code index
#define FRAME_HEAD1 0xAA//Data frame header 1
#define FRAME_HEAD2 0x55//Data frame header 2
typedef struct {
uint8_t len;//Data receive length
uint8_t rxbuffer[UART_RXBUFFER_SIZE]; //data receive buffer
}
typedef struct
{uint8_t len;//Data receive length
uint8_t rxbuffer[UART_RXBUFFER_SIZE]; //data receive buffer }
_S_UART_RX.
typedef struct
{uint8_t queue_head; //queue head
uint8_t queue_tail; //queue tail
}_S_QUEUE.
typedef struct {
uint8_t cmd; //command
```

```
uint8_t (* callback_func) (uint8_t cmd, uint8_t * msg, uint8_t len); //Command corresponding function
}_S_FUNCCALLBACK.
```

(2) Serial port interrupt initialization, so that the microcontroller in the attack device receives a byte of data that interrupts once when sending data, the queue is empty when interrupting:

```
Void User_UartIRQInit(uint8_t CMD)//entry parameter cmd command
{
if(ENABLE = = CMD)//judge whether the cmd command content and based on the interrupt initialization
{
__HAL_UART_ENABLE_IT(&huart1,UART_IT_RXNE).
}
if(DISABLE = = CMD).
{
__HAL_UART_DISABLE_IT(&huart1,UART_IT_RXNE).
}
}
```

(3) Serial task polling to determine if there is still data in the queue; if there are data, then continue to send, and if there are no data, then interrupt:

```
User_UartPoll(void).
{
if(0 = = s_uart_rx[s_queue.queue_head].len)//
return if there is no data in the queue
{
return 0.
}
if(s_queue.queue_head = = s_queue.queue_tail)//if the queue head pointer equals the queue tail pointer means the queue is full
{
if(s_queue.queue_tail > UART_RXBUFFER_SIZE-1)//if the tail pointer is polled
{
s_queue.queue_tail = 0.
}
else
{
s_queue.queue_tail++.
}
}
for(uint8_t The authors = 0; i < s_uart_rx[s_queue.queue_head].len; i++)//take out the data
{
User_UartDataParse(s_uart_rx[s_queue.queue_head].rxbuffer[i]).
}
s_uart_rx[s_queue.queue_head].len = 0.
if(s_queue.queue_head = = s_queue.queue_tail).
{
if(s_queue.queue_head > UART_RXBUFFER_SIZE-1)//head pointer polling
{
s_queue.queue_head = 0.
}
else
{
s_queue.queue_head++.
}
}
return 1; }
```

(4) Judgment of the processing of different parts of the data frame by the order in which the data are received in the accepted data frame:

```
User_UartDataParse(uint8_t data).
{
static uint8_t e_frame_status = frame_head1status;//frame status
static uint8_t frame_len = 0.
static uint8_t index = 0.
static uint8_t rx_bufftemp[256] = {0}.
uint16_t crc_temp = 0.
switch (e_frame_status){
case frame_head1status://judge data head 1
if(data = = FRAME_HEAD1).
{
e_frame_status = frame_head2status.
rx_bufftemp[index] = data.
index++.
}
else//if rx_bufftemp is not initialized correctly
{
e_frame_status = frame_head1status.
index = 0.
memset(rx_bufftemp,0.256).
}
break.
case frame_head2status://judge data header 2
if(data = = FRAME_HEAD2).
{
e_frame_status = frame_lenstatus.
rx_bufftemp[index] = data.
index++.
}
else//if rx_bufftemp is not initialized correctly
{
e_frame_status = frame_head1status.
index = 0.
memset(rx_bufftemp,0.256).
}
break.
case frame_lenstatus://judge the length of the data
if(data>0 && data ≤ 255).
{
e_frame_status = frame_datastatus.
rx_bufftemp[index] = data.
index++.
}
```

```
else//if rx_bufftemp is not initialized correctly
{
e_frame_status = frame_head1status.
index = 0.
memset(rx_bufftemp,0.256).
}
break.
case frame_datastatus://receive data
if(index>0 && index ≤ 255).
{
rx_bufftemp[index] = data; index++;
}
else
{
e_frame_status = frame_head1status;      index = 0;
memset(rx_bufftemp,0.256);
}
break; case frame_datastatus://Receive data
if(index>0 && index ≤ 255).
{
rx_bufftemp[index] = data;  index++;  if(index = =
(rx_bufftemp[FRAME_LEN_POS] + 3))//de-
termine if the reception of a frame is complete
based on the length of the data
{
crc_temp = rx_bufftemp[index-2]+(rx_bufftemp
[index-1]<<8);
if(crc_temp =                = CRC16(rx_bufftemp +
FRAME_CMD_POS,index-5))
//CRC checksum identical and store received data
{
User_UartFrameParse(rx_bufftemp[FRA-
ME_CMD_POS],rx_bufftemp, index); e_frame_s-
tatus = frame_head1status;                index = 0;
memset(rx_bufftemp,0.256);
User_UartFrameParseEnd();
}
}
else//Clear data, data header 1 is initialized
{
e_frame_status = frame_head1status;      index = 0;
memset(rx_bufftemp,0.256);
}
break;
default:e_frame_status = frame_head1status;
index = 0; memset(rx_bufftemp,0.256); break;
}
}
```

(5) Customize the commands sent by the attacking device with a similar command structure:

```
uint8_t User_ReadRegCallback(uint8_t cmd,
uint8_t * msg, uint8_t len).
{
uint8_t                    TestData[5] =
{0x01,0x02,0x03,0x04,0x05};//Customize sent
commands
User_UartFrameSend(cmd,TestData,msg,5);
}
```

(6) Attacking each microcontroller serial port in the device to send packets of data:

```
uint8_t User_UartFrameSend(uint8_t cmd,uint8_t
* pdata, uint8_t * msg, uint8_t len).
{
//Integrate and send different parts of data
uint8_t  index = 0;  uint16_t  crc_temp = 0;  msg
[index++] = FRAME_HEAD1;  msg[index++] =
FRAME_HEAD2;  msg[index++] = len;  msg
[index++] = cmd;  for(uint8_t  The  authors = 0;
i < len; i++)
{
msg[index++] = pdata[i];
}
crc_temp =CRC16(msg + -
FRAME_CMD_POS,index-3);  msg[index++] =
crc_temp & 0x00FF; msg[index++] = crc_temp>>8
&                                    0x00FF;
HAL_UART_Transmit(&huart1,msg,index,100);
return index;
}
```

Data Availability

Data and material are available at https://github.com/descfree/Research-on-IoT-security-detection-methods-based-on-multi-feature-fusion.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Authors' Contributions

The authors are responsible for the content and writing of this article.

Acknowledgments

This work was supported in part by the Joint Funds of the Zhejiang Provincial Natural Science Foundation of China under Grant no. LZY21F020001, in part by the Humanities and Social Sciences Planning Foundation of the Ministry of Education of China under Grant no. 22YJAZH063, and in part by the University Student Science and Technology Innovation Activity Plan (Xinmiao Talent Plan) of Zhejiang Province.

References

[1] Z. Huang and Y. Qin, "Malicious mining web page detection and forensics based on multi-feature recognition," *Netinfo Security*, vol. 21, no. 7, pp. 87–94, 2021.

[2] X. Yu, R. Ge, and F. Li, "Research on blockchain-based identity authentication scheme in social networks," *International Conference on Machine Learning for Cyber Security*, Springer, Berlin, Germany, 2020.

[3] C. Lee, S. Maharjan, K. Ko, J. Woo, and J. Wk Hong, *Machine Learning Based Bitcoin Address Classification,"International*

Conference on Blockchain and Trustworthy Systems, Springer, Singapore, 2020.

[4] Z. Cui, F. Xue, X. Cai, Y. Cao, G. G.. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3187–3196, 2018.

[5] C. Iwendi and G. Wang, "Combined power generation and electricity storage device using deep learning and internet of things technologies," *Energy Reports*, vol. 8, pp. 5016–5025, 2022.

[6] S. Tanwar, N. Gupta, C. Iwendi, K. Kumar, and M. Alenezi, "Next generation IoT and blockchain integration," *Journal of Sensors*, 2022.

[7] P. Singh, M. Masud, M. S. Hossain, and A. Kaur, "Blockchain and homomorphic encryption-based privacy-preserving data aggregation model in smart grid," *Computers & Electrical Engineering*, vol. 93, 2021.

[8] Y. Fang, C. Huang, L. Liu, and M. Xue, "Research on malicious JavaScript detection technology based on LSTM," *IEEE Access*, vol. 6, pp. 59118–59125, 2018.

[9] A. Alazab, A. Khraisat, M. Alazab, and S. Singh, "Detection of obfuscated malicious JavaScript code," *Future Internet*, vol. 14, no. 8, p. 217, 2022.

[10] M.-H. Wu, Y.-J. Lai, Y.-L. Hwang, T.-C. Chang, and F.-H. Hsu, *MinerGuard: A Solution to Detect Browser-Based Cryptocurrency Mining through Machine Learning"Applied Sciences*, vol. 12, no. 19, p. 9838, 2022.

[11] R. Tahir, S. Durrani, F. Ahmed, H. Saeed, F. Zaffar, and S. Ilyas, "The browsers strike back: countering cryptojacking and parasitic miners on the web," in *Proceedings of the IEEE Conference on Computer Communications*, pp. 703–711, Paris, France, August 2019.

[12] H. Geng, Z. Yang, and S. Yang, "How you get shot in the back: a systematical study about cryptojacking in the real world," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1701–1713, Toronto, Canada, October 2018.

[13] W. Bian, M. Wei, and M. Zhang, "Minethrottle: defending against wasm in-browser cryptojacking," *Proceedings of The Web Conference*, pp. 3112–3118, 2020.

[14] A. Swedan, A. N. Khuffash, O. Othman, and A. Awad, "Detection and prevention of malicious cryptocurrency mining on internet-connected devices," in *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, pp. 1–10, Amman, Jordan, June 2018.

[15] M. Lotfollahi, M. Jafari Siavoshani, and R. Shirali Hossein Zade, "Deep packet: a novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, pp. 1999–2012, 2020.

[16] H. Chi and C. Lin, *Industrial Intrusion Detection System Based on CNN-Attention -BILSTM Network," 2022 International Conference on Blockchain Technology and Information Security (ICBCTIS)*, Huaihua City, Huaihua, China, 2022.

[17] V. G. Le, H. T. Nguyen, and D. N. Lu, "A solution for automatically malicious web shell and web application vulnerability detection," *International Conference on Computational Collective Intelligence*, Springer International Publishing, Berlin, Germany, 2016.

[18] J. Fu, L. Lai, and Y. Wang, "CNN-based web shell file detection," *Journal of Zhengzhou University (Medical Science)*, vol. 51, no. 2, pp. 4–11, 2019.

[19] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," *Proceedings of the 24th USENIX Conference on Security Symposium*, USENIX Association, 2015.

[20] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: routing attacks on cryptocurrencies," in *Proceedings of the2017 IEEE Symposium on Security and Privacy (SP)*, pp. 375–392, San Jose, CA, USA, April 2017.

[21] C. L. Prabha and N. Shivakumar, "Software defect prediction using machine learning techniques," in *Proceedings of the 2020 4th International Conference on Trends in Electronics and Informatics*, pp. 728–733, Tirunelveli India, June 2020.

[22] F. Zhang, A. E. Hassan, S. McIntosh, and Y. Zou, "The use of summation to aggregate software metrics hinders the performance of defect prediction models," *IEEE Transactions on Software Engineering*, vol. 43, no. 5, pp. 476–491, 2017.

[23] Q. Yu, S. Jiang, R. Wang, and H. Wang, "A feature selection approach based on a similarity measure for software defect prediction," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 11, pp. 1744–1753, 2017.

[24] G. Yan, Q. Li, D. Guo, and X. Meng, "Discovering suspicious APT behaviors by analyzing DNS activities," *Sensors*, vol. 20, no. 3, p. 731, 2020.

[25] G. Husari, E. Al-Shaer, and B. Chu, "Learning APT chains from cyber threat intelligence," in *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security*, pp. 1-2, Nashville, TN, USA, April 2019.

[26] A. Awajan, "A novel deep learning-based intrusion detection system for IoT networks," *Computers*, vol. 12, no. 2, 2023.

[27] M. Rami Reddy, M. L. Ravi Chandra, P. Venkatramana, and R. Dilli, "Energy-efficient cluster head selection in wireless sensor networks using an improved grey wolf optimization algorithm," *Computers*, vol. 12, no. 2, 2023.

[28] G. Bhandari, A. Lyth, A. Shalaginov, and T.-M. Grønli, "Distributed deep neural-NetworkBased middleware for cyber-attacks detection in smart IoT ecosystem: a novel framework and performance evaluation approach," *Electronics*, vol. 12, no. 2, p. 298, 2023.

[29] S. Thandapani, M. I. Mahaboob, C. Iwendi et al., "IoMT with deep CNN: AI-based intelligent support system for pandemic diseases," *Electronics*, vol. 12, no. 4, p. 424, 2023.

[30] C. S. Kalutharage, X. Liu, C. Chrysoulas, N. Pitropakis, and P. Papadopoulos, "Explainable AI-based DDOS attack identification method for IoT networks," *Computers*, vol. 12, no. 2, 2023.

[31] R. J. Alzahrani and A. Alzahrani, "A novel multi algorithm approach to identify network anomalies in the IoT using fog computing and a model to distinguish between IoT and non-IoT devices," *Journal of Sensor and Actuator Networks*, vol. 12, no. 2, 2023.