Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105 (Online)
Volume **10**, Number **3**, August **2025**

Research on Vulnerability Mining Method of Modbus TCP Protocol in Intelligent Measurement System

Mei-Shan Zhong*

Joint Laboratory of Digital Technology for New Power System
Digital Grid Research Institute China Southern Power Grid, Guangzhou 510700, China
zhongms1@csg.cn

Peng Li

Joint Laboratory of Digital Technology for New Power System
Digital Grid Research Institute China Southern Power Grid, Guangzhou 510700, China
lipeng@csg.cn

Sheng-Rong Liu

Joint Laboratory of Digital Technology for New Power System
Digital Grid Research Institute China Southern Power Grid, Guangzhou 510700, China
starlory@163.com

Yue-Huan Lin

Joint Laboratory of Digital Technology for New Power System
Digital Grid Research Institute China Southern Power Grid, Guangzhou 510700, China
linyh2@csg.cn

Zhao-Hui Hu

Guangdong Provincial Key Laboratory of Digital Grid Technology
Southern Power Grid Digital Platform Technology (Guangdong) Co., Ltd, Guangzhou 510700, China
Huzh@csg.cn

Zhen-Heng Xu

Joint Laboratory of Digital Technology for New Power System
Digital Grid Research Institute China Southern Power Grid, Guangzhou 510700, China
xuzh@csg.cn

Jian Ma

Guangdong Provincial Key Laboratory of Digital Grid Technology

Digital Grid Research Institute China Southern Power Grid, Guangzhou 510700, China
352236881@qq.com

Ting-Wen Yu

Guangdong Provincial Key Laboratory of Digital Grid Technology
Southern Power Grid Digital Platform Technology (Guangdong) Co., Ltd, Guangzhou 510700, China
ytw@csg.cn

*Corresponding author: Mei-Shan Zhong

Received January 28, 2024, revised August 20, 2024, accepted December 3, 2024.

Abstract. *With the construction and development of digital power grid, Modbus TCP as an automatic equipment management and control protocol is more and more widely used in intelligent measurement system. How to mine the vulnerability of Modbus TCP protocol and deal with its potential risks has important theoretical significance and practical application value. The existing vulnerability mining methods of Modbus TCP protocol have the problems of low test case coverage and reception rate and high time complexity. In this paper, a vulnerability mining method for Modbus TCP protocol of intelligent measurement system is proposed. By constructing vulnerability fingerprint and systemting test case generation and optimization strategy, an anomaly monitoring mechanism is established. Through reverse feature extraction, vulnerability mining and feature field localization are finally realized. The simulation results show that the proposed scheme has good performance in vulnerability mining.*
Keywords: Intelligent Energy Measurement System; Modbus TCP Protocol; Vulnerability Mining; Fuzzy Testing; Public Vulnerability Library.

1. **Introduction.** With the rapid development of digital power grid, Modbus TCP protocol has been widely used in intelligent measurement system [1]. However, because Modbus TCP protocol does not consider security issues at the beginning of its design, it has some problems such as lack of authentication, lack of authorization, lack of encryption and abuse of function code [2]. In recent years, the vulnerability attack on Modbus TCP protocol of intelligent measurement system is increasing year by year, which has a serious impact on the system. In order to ensure the safe and stable operation of the system, it is necessary to mine the vulnerability of Modbus TCP protocol in intelligent measurement system.

Fuzzy testing has been widely used in protocol vulnerability mining because of its simplicity and efficiency. Fuzzing has been widely used in protocol vulnerability mining because of its simplicity and efficiency. In reference [3], a fully connected neural network model was constructed, and a lightweight analysis method was used to avoid the waste of computing resources and improve the fuzzing test performance. The fuzzing testing method based on program control flow graph improves the coverage of vulnerability mining paths by combining fuzzing and symbolic execution [4]. IoTInfer, a fuzzing test method based on finite state machine inference, balances the distribution of test cases by monitoring the possibility of conflicts between generated test cases and predictions [5]. The industrial control network protocol vulnerability mining method based on fuzzy testing identifies ICS vulnerability characteristics through change factors and generates test cases. Bypass monitoring is performed on the request-response relationship to realize anomaly monitoring for vulnerability mining [6]. Reinforcement learning algorithm MAB and a new reward evaluation model are used to improve the coverage rate of tool arrival

per unit time in kernel module fuzzy testing [7]. Directional grey box fuzzy testing method dynamically adjusts the energy of seed use cases through energy scheduling to improve the focus of fuzzy testing and the efficiency of vulnerability mining [8]. According to the existing mining methods, Li et al. designed a set of vulnerability mining performance evaluation index system including vulnerability detection ability, false positive rate, recall rate and resource overhead [9]. Based on data format discrimination and differential mutation method, frequency characteristics are found according to the location distribution of effective mutation, and effective mutation content is applied to data generation to improve fuzzing test efficiency [10].

Some researchers have tried to combine fuzzing techniques with emerging techniques such as machine learning to further improve fuzzing performance. The fuzzy test data set optimization algorithm based on LSTM identifies and eliminates invalid use cases through LSTM, effectively mining the vulnerability of Modbus TCP protocol in industrial control system [11, 12]. Gao et al. [13], combined with the deep learning is proposed based on a symbolic execution and fuzzy combination of hybrid testing method, interact by mixing mechanism, improve the fuzzy test coverage. The vulnerability mining tool based on generative adversarial learning (WGAN) performs generative adversarial learning training based on high availability abnormal use cases to generate test case sets to discover program vulnerabilities [14, 15]. Fuzzy testing method based on the guidance of historical version information calculates the fitness of test cases according to the execution trajectory, and generates test case sets through genetic algorithm [16].

In view of the abnormal monitoring and positioning hole digging, fuzzy test active state detection and positioning method with the method of periodic active detection of abnormal monitoring, and through the record feedback test done by tracing the timing of the anomaly orientation [17]. Wang et al. [18], puts forward a kind of collective abnormal monitoring method based on traffic, based on the abnormal flow on the result of clustering analysis, has the high traffic anomaly detection rate. Yang et al. [19], proposes a heuristic reverse path tracking localization algorithm, based on two-dimensional simulation graph model of heuristic reverse search for leak path tracing and back positioning.

Most of the above vulnerability mining methods are guided by a single performance index such as coverage rate or acceptance rate, which is difficult to improve the performance of vulnerability mining in various aspects. Accordingly, In this paper, an intelligent measurement system Modbus TCP protocol vulnerability mining method (MT-FUZZ) is proposed. By analyzing the vulnerability information related to Modbus TCP protocol in the public vulnerability database, a vulnerability information database was established and a set of vulnerability fingerprints were constructed. A suitable test case generation strategy for Modbus TCP protocol vulnerability mining was designed, and an optimization mechanism was established to screen the test cases. Then, the Modbus TCP protocol vulnerabilities are mined and anomalies are monitored through test cases. Finally, the vulnerability feature fields are located by feature selection and the vulnerability database is updated to guide the subsequent test case generation direction and improve the vulnerability mining performance.

2. **Modbus TCP protocol vulnerability library and vulnerability fingerprint construction.**

First, from the Common Vulnerabilities and Exposures (CVE), National Vulnerability Database(NVD), China National Vulnerability Database (CNVD) and other public vulnerability databases to collect vulnerability information about Modbus TCP. Then, according to the vulnerability information, the key vulnerability feature fields were extracted, including vulnerability description, vulnerability type, affected software version, vulnerability level etc, and these fields were stored in the Modbus TCP protocol

vulnerability database. Through the analysis of the corresponding relationship of protocol vulnerabilities and feature field building vulnerability fingerprint (vulnerability characteristics field) And it uses the legal field value to fill the missing field to construct a new effective use case suitable for the vulnerability mining environment in order to find the open vulnerabilities in Modbus TCP protocol. The schematic diagram of vulnerability fingerprint construction process is shown in Figure 1.

FIGURE 1. Schematic diagram of vulnerability fingerprint construction process of Modbus TCP

3. Modbus TCP Vulnerability Mining Test Case Generation Strategy.

3.1. **Analysis of Modbus TCP protocol architecture.** The data frame of Modbus TCP protocol is divided into two parts: packet header MBAP and protocol data unit PDU, as visualized in Figure. 2.

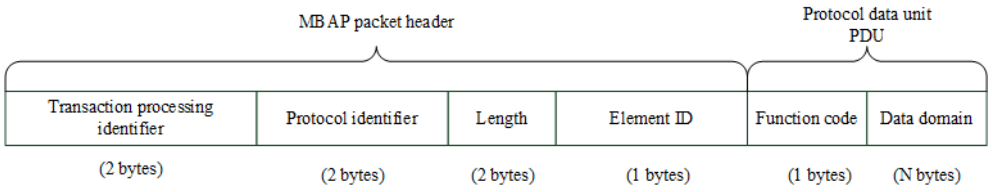

FIGURE 2. Modbus TCP protocol structure

Function codes are used to identify the master's indication to the slave of the operation to be performed. If the request sent by the slave is error-free, then the master's response packet will contain the function code in the request packet; if the request sent by the slave is in error, then the master's response packet will contain an exception code and an error code.

Exception codes are mainly used to diagnose and debug communication errors and find problems in communication. Such as data format, beyond the scope of work, according to the corresponding exception code and exception information description, guide users to take targeted solutions according to the situation. Table 1 shows the Modbus TCP common exception codes 0x01-0x06, 0x08, 0x0A, 0x0B etc. and their related functional descriptions. The value of the error code in the Modbus TCP protocol exception response data packet is the function code of the request data packet add 0x80.

TABLE 1. Modbus TCP exception code and function Description

Exception code	Description
0x01	Illegal function. The requested function code is an impermissible operation.
0x02	Invalid data address. The requested data address is an inadmissible address.
0x03	Invalid data values. The value included in the request is not allowed.
0x04	Slave equipment failure. The slave device produced an unresponsive error while performing the requested operation.
0x05	Confirm. The slave accepts and is processing the request.
0x06	Slave device busy
0x08	Store parity errors. The extended file area failed the consistency check.
0x0A	Gateway path not available. Usually refers to misconfigured or overloaded gateways.
0x0B	The gateway target device failed to respond.

3.2. **Test case generation.** The Modbus TCP protocol has normative constraints as shown in Equation (1).

$$Num_{len} = Len_{unit\_id} + Len_{PDU} \tag{1}$$

Where, Num_{len} is the corresponding value of the packet length field, Len_{unit_id} and Len_{PDU} is the unit identifier field and PDU length respectively.

Spearman correlation coefficient analysis was performed on the use case fields of the preprocessed data set. According to the weight of each field value on the effectiveness of the use case, design field variation probability set P_f. In Figure 3, $trans\_id$, $proto\_id$, len, $unit\_id$, fc, $data$, tag represent the transaction identifier, protocol identifier, length, unit identifier, function code, data field, and use case label of Modbus TCP protocol use case, respectively. The invalid use case is represented by $tag = 0$ in the dataset, $tage = 1$, represents a valid use case.

From the data in the seventh column of the matrix in Figure 3, the influence weight of each field value of the use case on its effectiveness can be obtained. Positive values indicate positive correlation between them, and negative values indicate negative correlation between them. The set of field variation probabilities is determined to have $P_f = \{P_{trans\_id}, P_{proto\_id}, P_{len}, P_{unit\_id}, P_{fc}, P_{data}\}$ through the correlation matrix. Among them, P_{trans_id}, P_{proto_id}, P_{len}, P_{unit_id}, P_{fc}, P_{data} it represents the mutation probability of 6 fields of test case transaction identifier, protocol identifier, length, unit identifier, function code and data field respectively, and its initial value is the absolute value of the value of column 7 of the correlation matrix of the application case field in turn.

Analyze the vulnerability entries related to Modbus TCP protocol in the vulnerability repository and design the set of probabilities for choosing the field variant method P_{mu}.

$$P_{mu} = 1 - \frac{1}{2 + N_{mu}} \tag{2}$$

$$N_{mu} = N_{pu} + N_{pr} \tag{3}$$

$P_{mu} = \{P_{mu1}, P_{mu2}, P_{mu3}, P_{mu4}\}$, P_{mu1}, P_{mu2}, P_{mu3}, P_{mu4}, respectively represents the selection probability of the field for variation mode $mu1$, $mu2$, $mu3$, $mu4$, N_{mu} represents the number of vulnerabilities associated with this mutation in the Modbus TCP protocol vulnerability library. N_{pu} denotes the number of Modbus TCP vulnerabilities associated

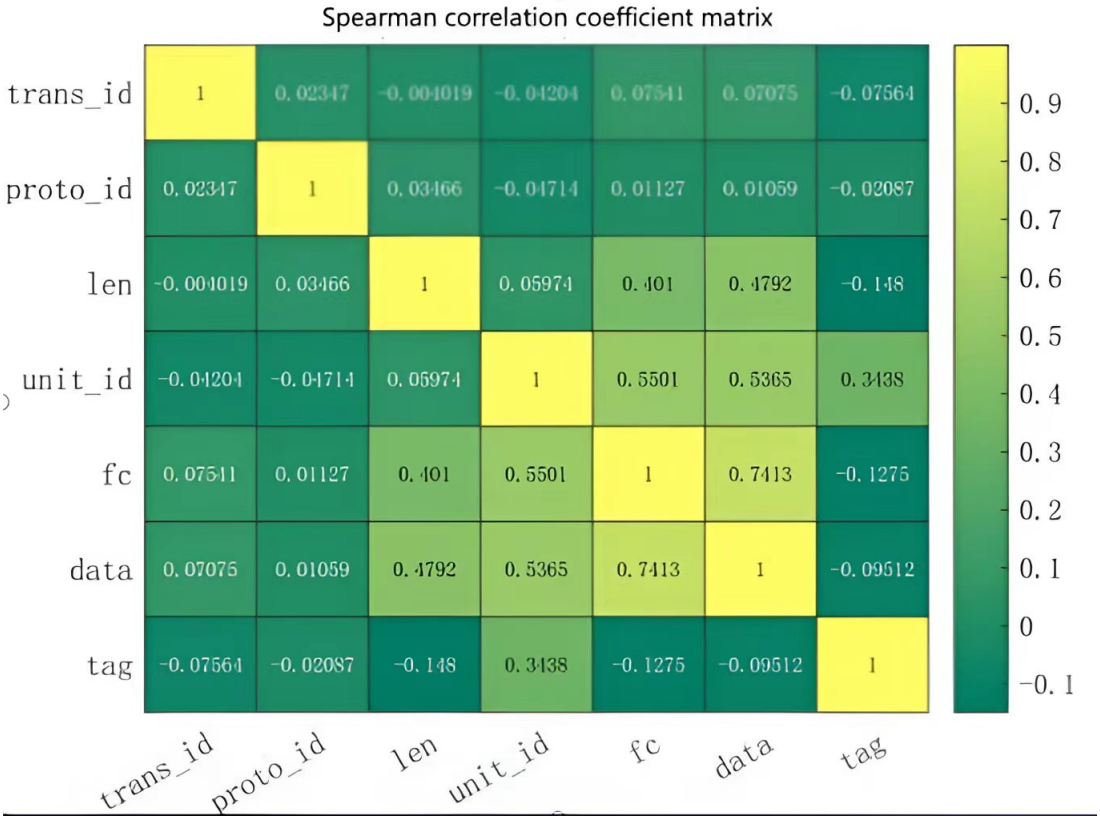

FIGURE 3. Test case field correlation matrix

with this mutation in the public vulnerability library. N_{pr} represents the number of vulnerabilities of Modbus TCP protocol recently mined in this paper related to this mutation mode.

As shown in Figure 4, the mutation operation is performed on the Modbus TCP protocol seed case to generate test cases. Construct a random array of length 6, Array values are random numbers in the range of $[0, 1]$. Corresponding to the six fields of the Modbus TCP protocol seed use case, Compare the random number and mutation probability corresponding to each field in the order of field constraints P_{mu}. When the random number is less than the mutation probability, a roulette wheel is constructed with the mutation probability P_{mu} to mutate the seed use case field.

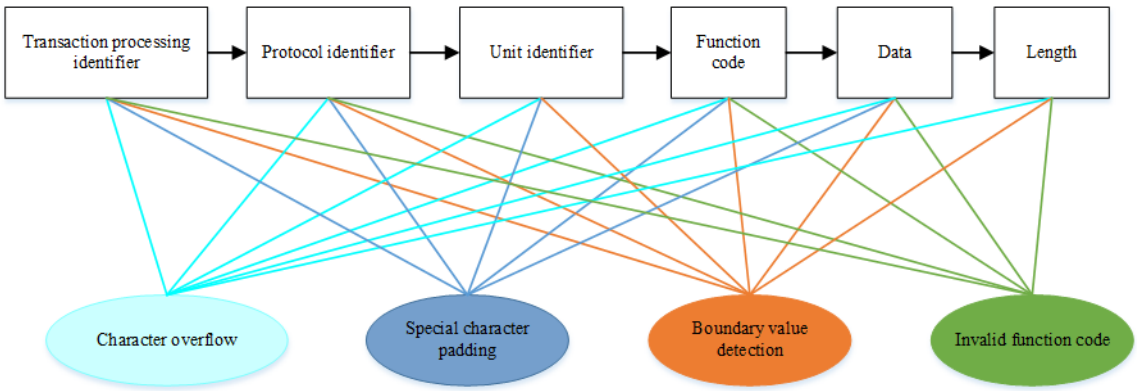

FIGURE 4. Test case generation strategy diagram

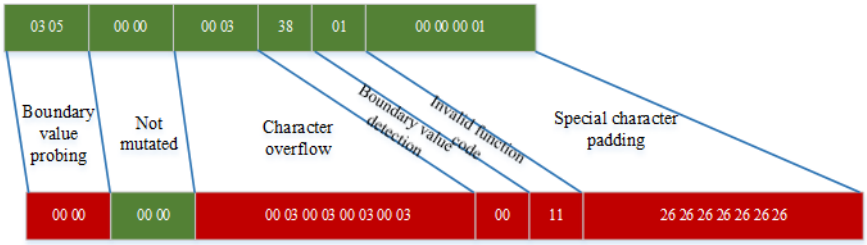

FIGURE 5. Test case variation diagram

The relevant constraints between fields are reflected through the systemtion of mutation order, and the guiding mutation is carried out through probability selection operation. Take the capture Modbus TCP protocol reads the coil state request data packet (data packet string composition:030500000003380100000001) as an example. The variation operation diagram is shown in Figure 5. The seed use cases were subjected to their respective mutation operations by field, and for the transaction identifier and unit identifier fields, the proposed policy chose the boundary value probing approach for the mutation operations. For the length field, the proposed strategy selected the character overflow method for mutation operation. For the function code field, the proposed strategy selected the illegal function code method for mutation operation. For the protocol identifier, the proposed strategy did not mutate it. For the data domain, the proposed strategy selected a special character filling method for mutation operation.

4. Test case optimization strategy for Modbus TCP vulnerability mining.

4.1. Test case generation. Due to the segmentation processing of the original ultra-long test case by the Modbus TCP protocol sending module, the original test case is divided into multiple identical sub-cases. For the cases of the test is for repeat operation cases. In order to solve the problem of repeated testing, the proposed vulnerability mining method specifies the maximum length of a single test case len_{max}:

$$len_{max} = 2 \times len_{lim} \tag{4}$$

len_{lim}represents the Modbus TCP protocol packet length specification. For function code 0x01-0x06 corresponding fixed length of 12 bytes of test case $len_{lim} = 12$. For variable length test cases with function code 0x0F corresponding to 0x10 there is $len_{lim} = 256$.

4.2. Lstm-based use case optimization. The proposed algorithm selected 1000 different types of data from the CSET 2016 Modbus TCP protocol public dataset by uniform sampling for LSTM model construction, 70% is used for training and 30% for testing. In this paper, the validity of the test case is predicted based on each field of the test case, so the input and output dimensions are 6 and 1, respectively. The single hidden layer network structure is adopted, and the root mean square error RSME is selected as the loss function:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Y - Y_{pre})^2}{N}} \tag{5}$$

Y represents the true value of test case validity, Y_{pre} represents the validity prediction value of the test case in the LSTM use case preference model. N represents the number of test cases. In this model, there are $N = 700$ and $N = 300$ for the training set and test set respectively.

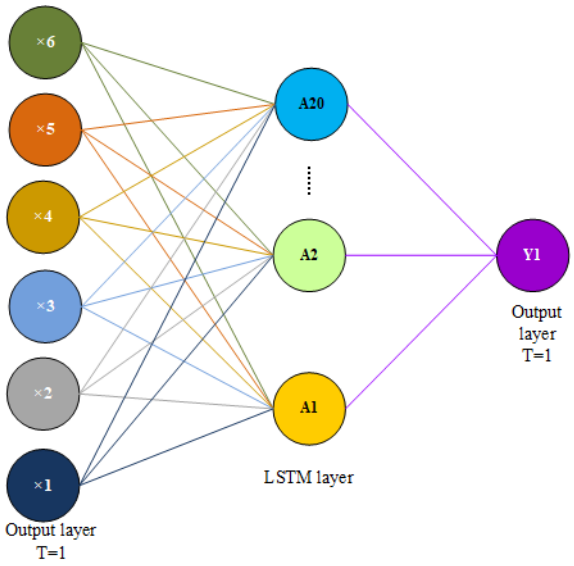

FIGURE 6. Lstm-based Optimization model diagram for use cases

Using Adam optimizer, the activation function of hidden layer input gate and forgotten gate is sigmod function, and the activation function of output gate is tanh function. The weights were randomly initialized, the learning rate was 0.01, and the number of hidden layer neurons was 20, *batch_size* for parallel processing is 64. The LSTM model shown in Figure 6 is established based on the above parameters. $X_i, i = 1, 2, \cdots, 6$ represents the decimal value of string transformation corresponding to each field of the test case, and it is used as input to the LSTM model at time T, $T = 1, 2, \cdots, 100$ is the time node, Y_1 denotes the validity of the corresponding test case at time $T = 1$. Corresponding to the data set of each test case the tag label, $A_1, A_2, \cdots, A_{20}$ represents the 20 hidden layer neurons of the LSTM model at time T.

Figure 7 shows the simulation results of LSTM effectiveness prediction for training and testing with given parameters.

Figure 7 shows that the prediction of the validity of the test cases of the dataset based on the LSTM model has a good fitting effect, At the same time, the root mean square error in the training set and the test set reaches 0.0041 and 0.0086 respectively, which can well meet the requirements of this topic for the optimization of test cases.

The proposed algorithm uses part of the selected CSET 2016 Modbus TCP protocol public dataset to train the LSTM use case optimization model. In addition, the trained model is used to predict the validity of test cases. If the use case triggers the vulnerability, the training output is 1, otherwise it is 0. According to the LSTM model to predict the effectiveness of test cases, a large number of invalid cases can be removed, and finally the optimal selection of test cases can be realized.

5. **Modbus TCP vulnerability mining anomaly monitoring mechanism.**

5.1. **Analysis of Modbus TCP packets.** Different request and response patterns for different types of use cases in smart energy measurement systems. Table 2 - Table 5 shows the PDU structure of common request response packets of Modbus TCP protocol, respectively.

As shown in Table 2, read coil and read discrete input register operations correspond to function codes 0x01 and 0x02, respectively. Both of them are mainly used to read LED display and switch status, and have similar request and response packet structure. For the

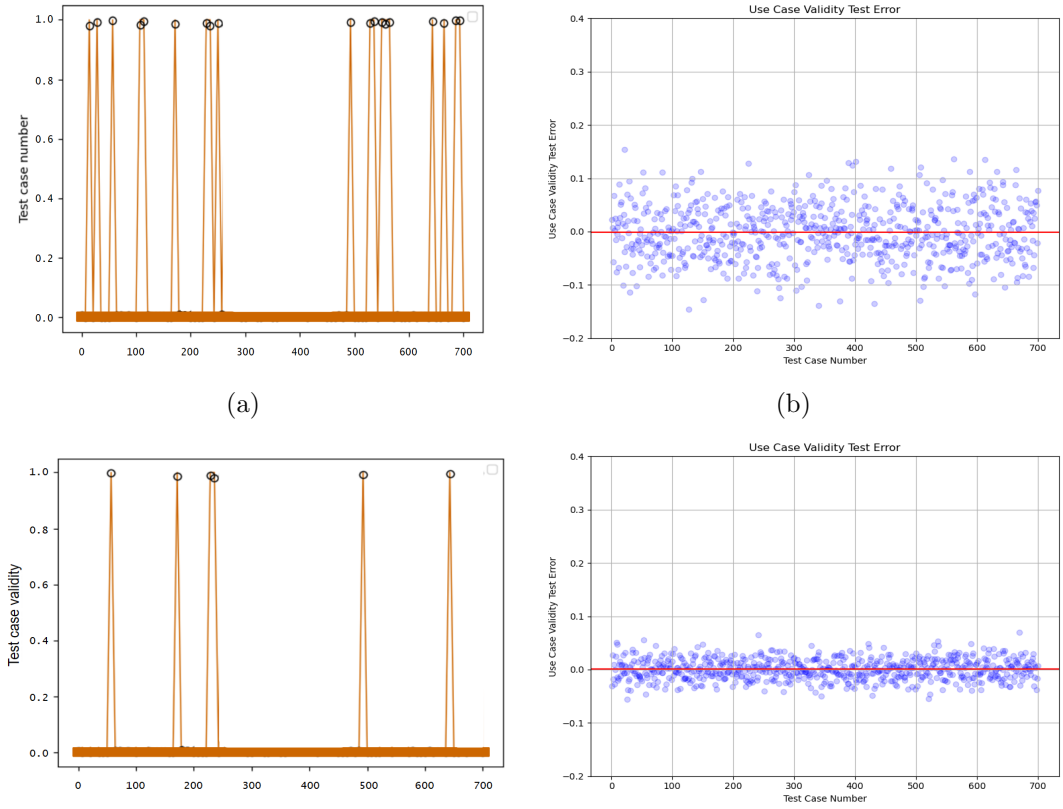

FIGURE 7. LSTM effectiveness prediction simulation diagram

remote device, you can use the function code 0x01, 0x02 request reads continuous coil and the dispersed quantity input in 2000. Each byte in the data part of the response message is converted from hexadecimal to 8-bit binary, corresponding to the coil and discrete input state of the continuous address of the request message, and binary 1 represents ON and 0 represents OFF.

TABLE 2. 0x01, 0x02 Request and Response PDU structure

Fields	*Length (bytes)*	Data rang
Request function code	1	0x01, 0x02
Request start address	2	0x0000-0xFFFF
Number of requested coils	2	0x0001-0x07D0
Normal response code	1	0x01, 0x02
Normal response bytes	1	N_{byte}
Positive normal response coil/discrete input	N_{byte}	—
Abnormal response difference error code	1	0x81, 0x82
Exception response exception code	1	0x01-0x04

When the request coil and the number of discretizers are not multiples of 8, the high bit of the last response byte is filled with 0. Therefore, for the normal response data part of bytes N_{byte} :

$$N_{byte} = ceil\frac{N_{request}}{8} \tag{6}$$

$N_{request}$ denotes the number of requests, $N_{request} \in \{N_{coil\_status}, N_{input\_status}, N_{register}\}$, N_{coil_status}, denotes the number of requested coil states, N_{input_status} denotes the number of discrete input states requested, $N_{register}$ denotes the number of request registers, *ceil* stands for the round up function.

As shown in Table 3, read register and input register operation code 0x03, 0x04 corresponding function. Both of them are mainly used to read the upper and lower thresholds of parameters and sensor inputs, and have similar request and response packet structures. The difference codes of the exception response between the device read hold register and the input register are 0x83 and 0x84. Support 1-125 consecutive register reads and responses. Every 2 bytes of the data part in the response packet correspond to 1 register content, and there is a $N_{request} = N_{register}$.

Table 3. 0x03, 0x04 Request and Response PDU structure

Fields	*Length (bytes)*	Data rang
Request function code	1	0x03, 0x04
Request start address	2	0x0000-0xFFFF
Number of requested registers	2	0x0001-0x07D
Normal response code	1	0x03, 0x04
Normal response bytes	1	$2N_{byte}$
Normal response register value	$2N_{byte}$	—
Abnormal response error code	1	0x83, 0x84
Exception response exception code	1	0x01-0x04

As shown in Table 4, write a single coil with a single keep register corresponding operation function code 0x05, 0x06. Both are mainly used to write the output and operating parameters of a single solenoid valve, and have similar request and response packet structures. The abnormal response error codes written by the device to a single coil and register are 0x85 and 0x86 respectively, and each 2 bytes of the data part in the request and response message corresponds to 1 value. The two hexadecimal bytes 0xFF00 are ON and 0x0000 are OFF. The normal request and response PUD structures should have exactly the same content.

Table 4. 0x05, 0x06 Request and Response PDU structure

Fields	*Length (bytes)*	Data rang
Request function code	1	0x05, 0x06
Request output address	2	0x0000-0xFFFF
Requesting output values	2	0x0000-0xFF00, 0x0000-0xFFFF
Normal response code	1	0x05, 0x06
Normal response output address	1	0x0000-0xFFFF
Normal response output value	2	0x0000 or 0xFF00, 0x0000-0xFFFF
Abnormal response error code	1	0x85, 0x86
Exception response exception code	1	0x01-0x04

As shown in Table 5, write multiple coils and multiple keep register corresponding function code 0x0f operation, 0x10. They are mainly used to write multiple solenoid valve output and operating parameters, and have similar request and response packet structure. Equipment respectively support multiple coils and register x0001 x0001 x07b0 and 0-000x007b continuous coil and written to the register. The exception response

difference codes are 0x8F and 0x90, respectively. Every byte of the output value in the request packet of the write coil is converted into 8-bit binary code, corresponding to write 8 values, binary 1 means ON, 0 means OFF. When the output number is not a multiple of 8, the high bit of the last output value of the request packet is filled with 0.

TABLE 5. 0x0F, 0x10 Request and Response PDU structure

Fields	*Length (bytes)*	Data rang
Request function code	1	0x0F, 0x010
Request start address	2	0x0000-0xFFFF
Requested output quantity	2	0x0001-0x07B0, 0x0001-0x007B
Requested bytes	1	N_{byte}, $2Nbyte$
Requesting output values	N_{byte}	
Normal response code	1	0x0F, 0x10
Normal response start address	2	0x0000-0xFFFF
Number of normal response outputs	2	0x0001-0x07B0, 0x0001-0x007B
Abnormal response error code	1	0x8F, 0x90
Exception response exception code	1	0x01-0x04

5.2. **Modbus TCP protocol status monitoring.** In traditional vulnerability mining methods, anomaly monitoring often uses heartbeat packets sent at fixed time intervals to detect the viability of slaves, it has a good discovery effect for denial of service vulnerabilities. However, the detection of other types of vulnerabilities is prone to false negatives. By analyzing the Modbus TCP anomaly response pattern, designed a abnormal monitoring mechanism to capture of effective use cases in hole mining, it realizes the anomaly monitoring of Modbus TCP protocol vulnerability mining process. The anomaly monitoring module is deployed in the Modbus TCP master device, which establishes a connection with the slave device in the system through Socket communication through the switch and sends test cases to observe its response.

If the master device sends an abnormal use case to the slave device, and the end device does not respond to all subsequent use case requests, the use case may trigger a denial of service vulnerability of Modbus TCP protocol. If the Modbus TCP protocol fails to recognize the irrationality of the abnormal use case after the master device sends the abnormal use case to the slave device and the slave returns the illegal data response, the use case may trigger the Modbus TCP protocol structure and specification vulnerabilities.

5.3. **Vulnerability feature field location.** The proposed method uses the K-means++ algorithm to cluster the effective use cases. The Modbus TCP protocol vulnerability feature fields are located by the dispersion reflected by the variance of each field. The specific implementation process is as follows:

Step 1: Select a central use case at random in the set of valid use cases.

Step 2: Calculate the shortest Euclidean distance $D(x)$ of each non-central use case to all the use cases in the existing central use case set.

$$D(x) = min\{d(x, c_i)\} \tag{7}$$

represents the non-central use case, and c_i represents the i central use case, $i = 1, 2, \cdots, k$, k denotes the number of clusters, and the initial value is 1.

Step 3: The central case selection probability $P(x)$ is calculated and the next central case is selected according to $P(x)$.

$$P(x) = \frac{[D(x)]^2}{\sum_{x \in X} [D(x)]^2} \tag{8}$$

X denotes the set of non-central use cases.

Step 4: Repeat step 2 and step 3 until the selected case k initial center.

Step 5: Calculate the distance of each non-central use case to each initial central use case and add the non-central use case to the cluster of the nearest central use case. The new cluster center use case c_i^{new} is:

$$c_i^{new} = \frac{\sum_{x \in C_i} (m_x + m_{c_i})}{N_{C_i}} \tag{9}$$

m_x represents the string value corresponding to the non-central use case x. m_{c_i} represents the central use case c_i corresponding to the string value, C_i represents the ith cluster, and N_{C_i} represents the number of use cases in cluster C_i.

Step 6: Calculate the clustering performance index $s(k)$, change the initial k value to maximize $s(k)$ and obtain the best k value. And repeat steps 1-5 to get the optimal clustering.

$$s(k) = \frac{tr(B_k)(N_t - k)}{tr(w_k)(k-1)} \tag{10}$$

N_t indicates the number of valid use case samples, k denotes the number of clusters, B_k is the covariance matrix of use cases between classes, w_k is the covariance matrix of the use cases within the class, tr is the trace of the matrix.

Step 7: Reverse feature selection is performed for valid use cases based on feature filtering. According to Equation (11), the variance σ^2 of each field is calculated for each valid use case cluster. Select the sum of variance value is less than 0.1 field, that is, get effective use cases hole mining characteristics of a loophole in the field.

$$\sigma^2 = \frac{\sum_{i=1}^{N_{C_i}} (m_i - \bar{m})^2}{N_{C_i}} \tag{11}$$

N_{C_i} represents the number of use cases in clustering C_i, m_i represents the field value of the ith valid use case in the cluster, $\bar{m}$ denotes the valid use case field average.

Step 8: The Modbus TCP protocol vulnerability and its characteristic fields and other related information are saved to the vulnerability database, and the field mutation probability P_f and the field mutation mode selection probability P_{mu} are updated to guide the subsequent vulnerability mining direction.

6. **Experiment and Analysis.** Based on the software and hardware platform of the intelligent measurement system, the simulation environment of Modbus TCP protocol communication is built. MT-FUZZ, the proposed Modbus TCP protocol vulnerability mining method. Peach, the traditional fuzzy testing framework, WGAN vulnerability mining method proposed in literature [9], and GA-FUZZ [11], which is based on the combination of the traditional fuzzy testing framework and genetic algorithm, are used for testing. The dataset was presented by Antoine Lemay at CSET2016. Representative Modbus TCP protocol data set generated based on SCADA network, including labeled normal and abnormal data packet files.

6.1. **Experimental environment.** In order to verify the performance of the proposed method, this paper builds an experimental platform for Modbus TCP protocol vulnerability mining. The hardware platform of vulnerability mining experiment is AMD Ryzen 73700X8-Core Processor, 3.59 GHz, 16GB RAM. The software environment of the vulnerability mining experimental platform is Windows 10, Modbus Slave 4.3.1, Peach 3.1.124.0, and the compiler environment of the vulnerability mining experimental platform is Python3.6.

6.2. **Analysis of experimental results.** Compared with the traditional fuzzing framework Peach, the WGAN vulnerability mining method proposed by literature [9] and the GA-FUZZ vulnerability mining method proposed by literature [11], the evaluation indicators including test case coverage and test case acceptance rate were compared. The experimental test results are shown in Table 6. The proposed MT-FUZZ method can find the most abnormal cases in the least time under the same test set and experimental environment.

TABLE 6. Result of Modbus TCP vulnerability mining

Scheme	Test set size	Time consuming	Number of abnormal use cases
WGAN	15000	10h16min	148
GA-FUZZ	15000	8h9min	65
Peach	15000	12h35min	17
MT-FUZZ	15000	6h56min	186

As shown in Figure 8, the proposed MT-FUZZ method uses the trained LSTM model to filter the test set, so that the number of actual test cases participating in the test is far smaller than the size of the initial test set, so as to reduce the time-consuming and realize fast and efficient vulnerability mining. In contrast, GA-FUZZ and WGAN respectively use genetic algorithm and generative adversarial learning to generate test cases, and the population evolution and GAN training process have no effect on the size of the test set. In the traditional Peach fuzzing framework, because the generated string is too long, multiple identical subcases will be generated after data processing, which eventually leads to the expansion of the test set.

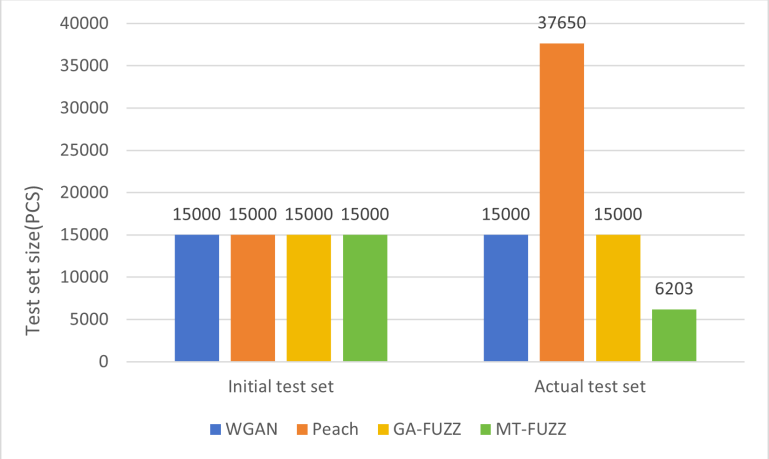

FIGURE 8. Test set size comparison diagram

Table 7 lists the abnormal use cases of Modbus TCP protocol vulnerability mining. The proposed MT-FUZZ vulnerability mining method found 186 abnormal use cases of

Modbus TCP protocol, and 8 categories (corresponding to 8 vulnerabilities) could be obtained by analyzing them. The vulnerabilities numbered (1)-(5) are included in the public vulnerability database, and vulnerabilities numbered (6)-(8) are newly discovered vulnerabilities. According to the PDU format of each function code request and response field of Modbus TCP protocol in Table 2-Table 5, it can be seen that the vulnerability numbered (6) and (7) corresponds to the problem of data field overflow in the abnormal use case, and the vulnerability numbered (8) corresponds to the problem of mismatch between the length field and the subsequent byte number.

Table 7. Result of Modbus TCP vulnerability mining

Vulnerability number	Exception use case	Feature field	Variation mode
(1)	Request:3D 5A 00 00 00 03 01 01 01 Response:3D 5A 00 00 00 05 01 01 02 00 00	fc: 01 data: 01 01 01	Character overflow
(2)	Request:3B B5 00 00 00 07 01 0F 00 07 00 02 01 Response:3B B5 00 00 00 06 01 0F 00 07 00 02	fc: 0F data: 00 07 00 02 01	Character overflow
(3)	Request:6F F4 00 00 00 03 01 03 01 Response:6F F4 00 00 00 03 01 03 00	fc: 03 data: 01	Character overflow
(4)	Request:0F 5C 00 00 00 08 01 10 00 08 00 01 02 0F Response:0F 5C 00 00 00 06 01 10 00 08 00 01	fc:10 data: 00 08 00 01 02 0F	Character overflow
(5)	Request:AD 82 00 00 00 05 01 06 00 03 07 Response:AD 82 00 00 00 06 01 06 00 03 07 0A	fc:06 data:00 03 07	Character overflow
(6)	Request:01 01 00 00 00 0A 01 01 00 00 00 0A 01 01 00 00 Response:01 01 00 00 00 05 01 01 02 00 00	fc: 01 data: 00 00 00 0A 01 01 00 00	Character overflow
(7)	Request:9D 36 00 00 00 03 01 01 00 Response:9D 36 00 00 00 0A 01 01 07 00 00 00 00 00	fc: 01 data: 00	Character overflow
(8)	Request:9D 36 00 00 00 03 01 01 00 00 00 01 Response:9D 36 00 00 00 05 01 01 02 00 00	fc: 01 len: 00 03 data: 00 00 00 01	Character overflow

Test case coverage is an important indicator for vulnerability mining performance evaluation. It is found that the group dispersion of the test suite is positively correlated with the test case coverage of vulnerability mining. Therefore, different solution generated by calculation of discrete degree of test cases set sample to reflect the change of the coverage of test cases, calculation Equation is:

$$\overline{field} = \frac{\sum_{i=1}^{N_{now}} field_i}{N_{now}} \tag{12}$$

$$\mathrm{MD} = \frac{\sqrt{\frac{\sum_{i=1}^{N_{now}} d(sample_i, \overline{sample})}{N_{now}}}}{\overline{MD}} \tag{13}$$

$\overline{sample}$ represents the current test case focus on the use case, and $\overline{sample} = [\overline{len}, \overline{unit\_id}, \overline{fc}, \overline{data}]$, $\overline{len}$, $\overline{uni\_id}$, $\overline{fc}$, $\overline{data}$ respectively test cases set length, the unit identifier, function code and the average of the data fields. $\overline{field}$ represents the average value of the test case set fields, $field_i$ denotes the field value of the ith test case. $\overline{MD}$ is the population dispersion of the test suite, $sample_i$ denotes the ith test case, N_{now} is the current test suite size, $d(sample_i, \overline{sample})$ represents the Euclidean distance between the ith test case and the central case in the current test case set.

Figure 9 shows the spatial distribution of use cases generated by each method. Because GA-FUZZ is implemented based on genetic algorithm. Therefore, there is a relatively obvious clustering phenomenon. Peach has no obvious distribution due to the randomness of its generation strategy. WGAN generated cases have certain boundaries and gathered phenomenon. GA-FUZZ, Peach, and WGAN all have multiple identical use cases covering each other at the same coordinate, while the distribution of MT-FUZZ use cases is discrete, with at most and only one use case at the same location.

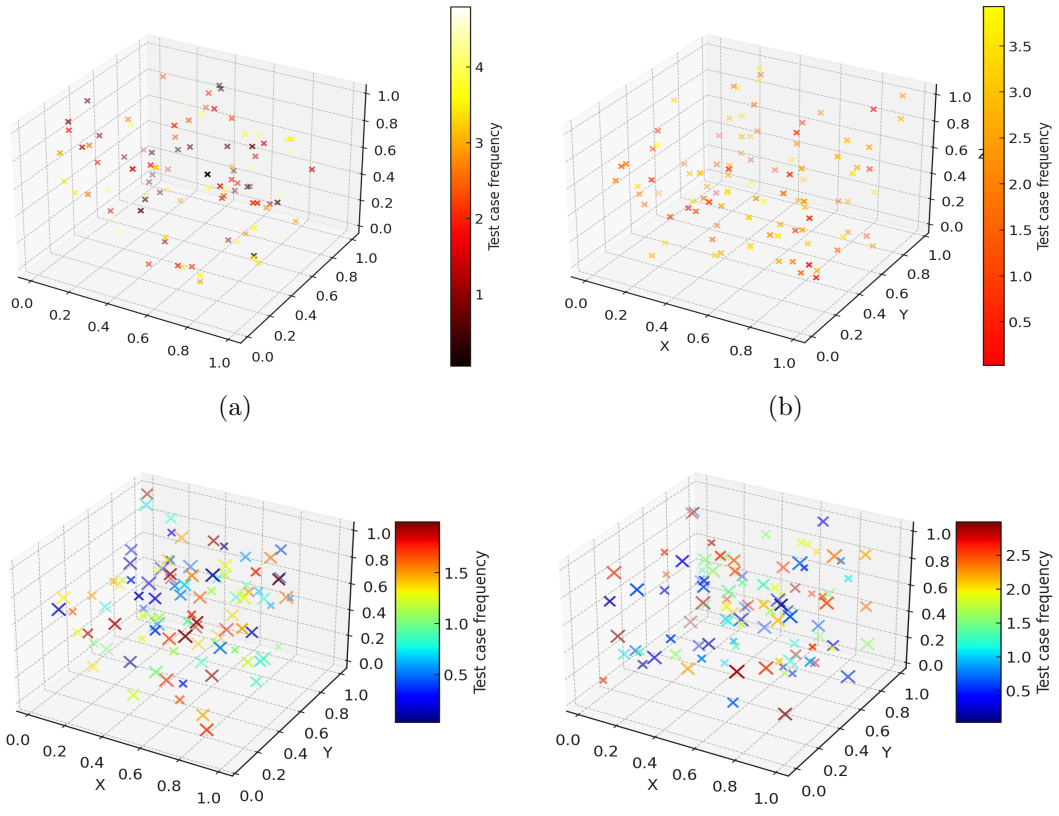

FIGURE 9. Spatial distribution of test cases

Table 8 shows the comparison results of test case acceptance rate of the four methods. MT-FUZZ proposed in this paper can go deep into the underlying exception handling logic of Modbus TCP protocol and avoid relevant handling mechanisms. As a result, the acceptance rate of the test case is improved by 25.5%, 32% and 6.2%, respectively,

compared with GA-FUZZ, Peach and WGAN. The system for calculating reception rate P_{acc} of test case is:

$$P_{acc} = \frac{N_{acc}}{N_s} \tag{14}$$

N_{acc} represents the number of different use cases received by the opposite PLC slave device in the test of the scheme, N_s represents the size of the test case set, which in this scenario has $N_s = 15000$.

TABLE 8. Result of Modbus TCP vulnerability mining

scheme	Exception use case	Feature field	Variation mode
WGAN	15000	9642	64.3%
GA-FUZZ	15000	6750	45%
Peach	15000	5771	38.5%
MT-FUZZ	15000	10578	70.5%

Figure 10 shows the comparison of test case coverage of each method. Compared with GA-FUZZ, Peach and WGAN, the coverage of the proposed method is increased by 15%, 46% and 28%, respectively.

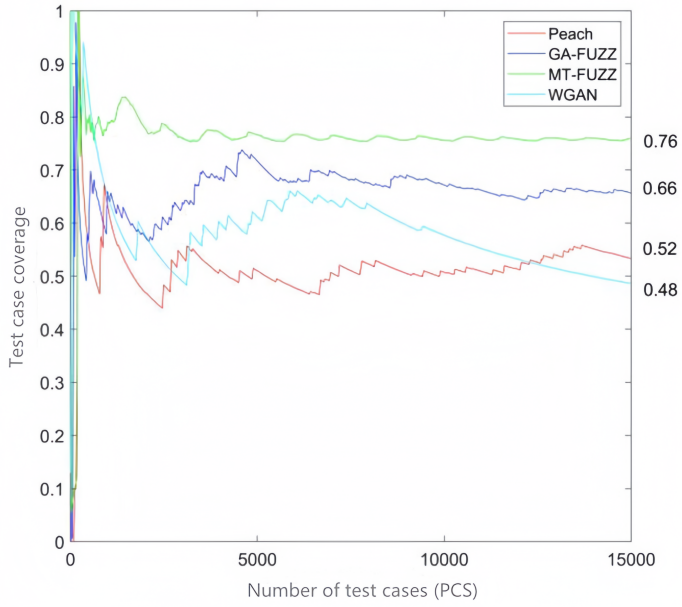

FIGURE 10. Test case coverage comparison chart

7. **Conclusions.** Aiming at the problems of the existing Modbus TCP protocol vulnerability mining methods, this paper proposed a Modbus TCP protocol vulnerability mining method of intelligent measurement system. This method constructs vulnerability fingerprint by analyzing vulnerability information related to Modbus TCP protocol in the public vulnerability database and systemtes a test case generation and optimization strategy based on vulnerability fingerprint and Modbus TCP protocol characteristics. An abnormal response model is established based on protocol monitoring mechanism of the abnormal monitoring in the process of hole digging. Finally, the vulnerability mining of Modbus TCP protocol was realized. Experiments show that the proposed method in the same concentration of test cases, can guarantee to dig holes at the same time high receiving rate and high coverage, and has good performance.

REFERENCES

[1] H. Wang, H.-L. Zhou, "Power hardware-in-the-loop simulation for noload grid-connection of DFIG," *Journal of Northeast Electric Power University*, vol. 40, no. 1, pp. 39-46, 2020.

[2] J. Yin, "Online construction of dynamic total transfer capability of transmission interface based on data mining technology," *Journal of Northeast Electric Power University*, vol. 43, no. 1, pp. 69-75, 2023.

[3] M. Wu, L. Jiang, J. Xiang, "Evaluating and Improving Neural Program-Smoothing-Based Fuzzing," in *44th International Conference on Software Engineering(ICSE 2022)*. IEEE, 2022, pp. 847–858.

[4] W.-M. Ling, J.-Y. Guo, N. Tang, "A fuzzy testing vulnerability mining method based on program control flow graph," *Engineering Journal of Wuhan University* , vol. 56, no. 9, pp. 1146-1153, 2023.

[5] Z. Shu, G. Yan, "IoT Infer: Automated Blackbox fuzz testing of Iot network protocols guided by finite state machine inference," *IEEE Internet of Things Journal* , vol. 9, no. 22, pp. 22737-22751, 2022.

[6] Y.-X. Lai, K.-X. Yang, J. Liu, "Vulnerability mining method for industrial control network protocol based on fuzz testing," *IEEE Computer Integrated Manufacturing Systems* , vol. 25, no. 9, pp.2265-2279, 2019.

[7] D. Wang, Z. Zhang, and H. Zhang, "SyzVegas: Beating Kernel Fuzzing Odds with Reinforcement Learning," in *30th USENIX Security Symposium* . USENIX, 2021, pp. 2741-2758.

[8] K. Yang, H.-P. He, H.-T. Ma, "Guiding directed grey-box fuzzing by target-oriented valid coverage," *IEEE Journal of Software*, vol. 33, no. 11, pp. 3967-3982, 2022.

[9] J. Ling, J. Chen, M. Huang, "An integration testing framework and evaluation metric for vulnerability mining methods," *China Communications*, vol. 15, no. 2, pp. 190-208, 2018.

[10] K. Yang, Y.-P. He, H.-T. Ma, "Mutation optimization of directional fuzzing for cumulative defects," *Journal of Software*, vol. 34, no. 5, pp. 2286-2299, 2023.

[11] C.-M. Shi, Y.-X. Feng, Y.-T. Zhao, "Method of Optimizing Fuzzy Test for Modbus TCP Protocol Based on LSTM," *Journal of Shenyang Ligong University*, vol. 42, no. 5, pp. 18-22, 2023.

[12] R. Fan, Y. Chang, "Machine Learning for BlackBox Fuzzing of Network Protocols" *Springer. Information and Communications Security-ICICS 2017*, Springer Berlin, 2018, pp. 621-632.

[13] F.-J. Gao, Y. Wang, L.-Y. Situ, "Deep learning-based hybrid fuzz testing," *Journal of Software*, vol. 33, no. 4, pp. 988-1005, 2021.

[14] Z. Li, H. Zhao,J. Shi, "An intelligent fuzzing data generation method based on deep adversarial learning," *IEEE Access*, vol. 7, pp. 49327-49340, 2019.

[15] Y. Zhang, W.-F. Cao, G.-K. Sun, "Network Protocol Vulnerability Mining Method Based on the Combination of Generative Adversarial Network and Mutation Strategy," *Computer Science*, vol. 50, no. 9, pp. 44-51, 2023.

[16] J. Zhang, Z. Cui, X. Chen, "DeltaFuzz: historical version information guided fuzz testing," *Journal of Computer Science and Technology*, vol. 37, pp. 29-49, 2022.

[17] W.-Y. Zhang, L. Zhang, J.-L. Mao, "An automated method of unknown protocol fuzzing test," *Chinese Journal of Computers*, vol. 43, no. 4, pp. 653-667, 2020.

[18] C. Wang, H. Zhou, Z. Hao, "Network traffic analysis over clustering-based collective anomaly detection," *Computer Networks*, vol. 2022, 108760, 2022.

[19] J. Yang, L. Huang, H. Ma, "A 2d-graph model-based heuristic approach to visual backtracking security vulnerabilities in physical protection systems," *International Journal of Critical Infrastructure Protection*, vol. 2022, 100554, 2022.