





Research Article

A Test Cases Generation Method for Industrial Control Protocol Test

Wenli Shang ¹, Guanyu Zhang ^{2,3}, Tianyu Wang ², and Rui Zhang ³

¹School of Electronic and Communication Engineering, Guangzhou University, Guangzhou 510006, China

²Industrial Control Network and Systems Department, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China

³Information and Control Engineering Faculty, Shenyang Jianzhu University, Shenyang 110168, China

Correspondence should be addressed to Wenli Shang; shangwl@gzhu.edu.cn

Received 15 October 2020; Revised 8 January 2021; Accepted 4 March 2021; Published 13 March 2021

Academic Editor: Ting Yang

Copyright © 2021 Wenli Shang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The coverage of test cases is an important indicator for the security and robustness test of industrial control protocols. It is an important research topic to complete the test with less use cases. Taking Modbus protocol as an example, a calculation method of case similarity and population dispersion based on weight division is proposed in this paper. The method can describe the similarity of use cases and the dispersion degree of individuals in the population more accurately. Genetic algorithm is used to generate and optimize test cases, and individual similarity and population dispersion are used as fitness functions of genetic algorithm. Experimental results show that the proposed method can increase the population dispersion by 3.45% compared with the conventional methods and effectively improve the coverage of test cases.

1. Introduction

The industrial control systems control the data collection, image and sound signal processing, information transmission, and process control during the entire production process. The safety and reliability during operation are related to the stability of the entire system. In recent years, with the rapid popularization and application of computer networks, the traditional industrial control system is gradually developing towards the direction of interconnection and intelligence, and some new concepts such as Internet of things, industrial Internet of things, and industry 4.0 are proposed. However, the Internet has injected new vitality into the industrial control system but also brought the same challenges [1–3].

In security system of the industrial control system, protocol is an important guarantee for the secure transmission of information. Attacks against the protocol are one of the most common methods because of low cost, and, with the rapid development of network, remote attack becomes possible [4, 5]. As the information transmission medium of

industrial control system, it is necessary to mine possible vulnerabilities of industrial control protocol through automated testing method to ensure its security and stability.

At present, the commonly used vulnerability mining techniques are divided into static analysis, dynamic analysis, binary comparison, fuzzy testing, and so on [6–12]. Fuzzy testing has the advantages of high automation, low system consumption, low false-alarm rate, and being independent of the source code of the object program [7]. The key step in fuzzy testing is test case generation. Traditional fuzzy testing often blindly mutates a part of normal test cases when generating test cases; this blind mutation method makes the scale of test cases reach 100000 or millions, but the test effect is not ideal. Therefore, the design and improvement of test case generation strategy are one of the hot research contents of fuzzy test technology.

Test case generation algorithms for fuzzer can be divided into three categories: generation-based method, mutation-based method, and combination of the two methods [13–17]. In the current protocol testing, there are some irrationalities in the coding method and similarity determination of test

cases, which will affect the coverage of the test, and it needs to be improved. Therefore, we compared the advantages and disadvantages of the three methods, combined with the data packet structure characteristics of the test protocol, and propose a new method based on weight division to calculate the case similarity and the use case average similarity. The goal is to generate use cases with better coverage and improve test efficiency. Compared with the existing literature, this paper has the following major contributions:

- (i) A new method to determine the similarity of use cases and the concept of population dispersion are proposed, which provides a new idea and method to improve the use case coverage in the process of protocol testing.
- (ii) Different weight and distance calculation methods are set according to different protocol fields, so the similarity can be determined more accurately according to the function and data content of the use case. The change of coding method also solves the problem of inaccurate similarity judgment caused by data mutation.
- (iii) The genetic algorithm is used to generate the use case, and the similarity and the population dispersion of the case are used as the fitness function of the genetic algorithm. Automatic optimization of the use case generation is realized.

The rest of this paper is organized as follows: In Section 2, we discuss the related work. In Section 3, we provide an introduction to Modbus protocol test case design method. Section 4 is about the computing method for test cases average similarity and population dispersion. Section 5 contains simulations and results and evaluates the results based on the requirements, while Section 6 draws conclusions and reviews based on the results.

2. Related Work

2.1. Generation-Based Method. Generation-based method is to build mathematical model according to the protocol specification of test object and then generate test cases automatically. Martins et al. [18] describe a tool called ConData used as test generation for communication protocols specified as extended finite state machines. The strategy for test generation combines different specification-based test methods. Although the values for fields of interactions are automatically generated, the human intervention is always needed to determine more suitable values for test case purposes. Banks et al. [19] present SNOOZE, a tool for building flexible, security-oriented network protocol fuzzers. SNOOZE implements a stateful fuzzing approach that can be used to effectively identify security flaws in network protocol implementations. But SNOOZE is not evaluated using the code coverage metric. Li et al. [20] present an automatic vulnerability discovering method that combines automatic Protocol Reverse Engineering technology and Fuzz Testing. The method is a four-step program involving packets clustering, multiple sequences alignment, special fields recognition, and

fuzzer production, which find the structure of network packets and pursue Fuzz Testing. However, the effectiveness of the proposed method depends on the diversity of the sampling packet itself, so it is necessary to sample the network protocol multiple times and try to ensure that the network protocol is used with different parameters each time. Voyiatzis et al. [21] present the design and implementation of MTF, a Modbus/TCP Fuzzer. The MTF incorporates a reconnaissance phase in the testing procedure so as to assist mapping the capabilities of the tested device and to adjust the attack vectors towards a more guided and informed testing rather than plain random testing. The disadvantage is that Modbus/TCP Fuzzer should be redesigned for different implementations of the Modbus protocol. Liu et al. [22] proposed a heuristic network protocol fuzzy test case generation method based on the heuristic search algorithm and classification tree thought. The Peach and FTP are selected as the verification platform and target protocol, respectively. The test result verified the feasibility and effectiveness of fuzzy test case generation method of heuristic network protocol. However, the coverage of test cases in this paper depends on the accuracy of network protocol classification tree construction. Felix et al. [23] introduced a novel fuzzer, Policy Generator (PG). PG utilizes a number of heuristic techniques to improve space coverage over existing fuzzers. The empirical study demonstrates that PG generates superior coverage compared to current generation techniques. However, many of the metrics correlate and care needs to be taken when interpreting the presented data. In addition, while it is believed that the experimental framework describes this evaluation accurately, the analysis cannot be safely generalized beyond the grammatical expression of the generic firewall policy utilized in this article. Liu et al. [24] propose a vulnerability mining method combining protocol reverse analysis and fuzzy method. An improved effective counting method based on local greedy algorithm is proposed to improve the accuracy of protocol keyword extraction by 65%. Combining the lossy counting method to construct a protocol syntax tree reduces the number of spanning tree nodes by 40%. Although the performance of the proposed method is better than traditional method, it still needs to be improved in terms of operation efficiency and applicability. For example, due to the NLP method, the performance will decrease significantly while extracting keywords for pure binary protocol reverse analysis.

The main advantage of generation-based method is that the same set of test cases can be used directly for the same test objectives, and the generated test cases have high coverage [25, 26]. The main disadvantage of generation-based method is that it takes a lot of time and effort to complete the understanding of file format or protocol specification and the writing of rules. Different target types of software differ greatly. It is difficult to reuse and has a small scope of application [25, 26].

2.2. Mutation-Based Method. Mutation-based method is that a new generation of test cases is generated by mutation strategy designed based on the existing input samples. Gu et al. [27] propose a novel message matrix perturbing mode to generate test case through data mutation for application

layer protocol. Additionally, a new statistical keyword extracting technique with priority recursive splitting pattern is introduced to provide useful information for intelligent data mutation. The work presented in the paper is not perfect at several aspects. First, the static statistical analysis just finds a balance between extracting performance and computational complexity. Second, the keywords with low occurrence frequency cannot be grasped through the current method. Last but not the least, the discrimination on different protocol elements is not explicit enough for intelligent fuzzing. A test case generation technique based on mutation algorithm of precaptured IPC data is introduced in [28] in order to improve the fuzzing test efficiency. Two high-risk vulnerabilities are detected in Android 5.1.0. Analysis of these vulnerabilities highlights a critical design issue in the system services of Binder mechanism. The test case generation algorithm needs to be improved leveraging program analysis technique. Lai et al. [29] proposed a vulnerability mining method for industrial control network protocol based on fuzz testing. Protocol feature values were generated by testing cases variation factors for industrial control network protocol, each of which represented a type of ICS vulnerability features. Different test cases were generated by Modbus TCP features and variation factors. Through bypass monitoring method and Modbus TCP features relation between request and response, the difficult problem of determining the validity of testing cases was solved. However, the learning results of industrial control private protocol feature learning method will produce uncertainty due to different data sets. If the characteristics of private protocol need to be analyzed deeply, some manual analysis needs to be done. Cai et al. [30] give a fuzzy security test method based on the grammatical model and propose a grammar model for industrial control protocol based on high-order attribute grammar. The model proposes a fuzzy security test algorithm, combined with the characteristics of the industrial control protocol, and elaborates on the analysis tree structure, test case generation, and mutation strategy. The model performs comparative experiments by simulating Modbus/TCP communication which verifies that anomalous results can still be found at a lower time cost when generating fewer test cases. Accuracy of description model for the industrial control protocol based on subjective understanding will impact test case coverage. Xu et al. [31] proposed the use of deep learning technology to assist test case generation. Using the advantage of recurrent neural network to deal with character text sequences, it learnt training structure features through sample data, predicted new data that conformed to structural features, and constructed an automatic generation model to combine with random mutation algorithm. In order to make the test case generation more targeted and easier to trigger exceptions, the appropriate deep learning network should be studied to learn the auxiliary weight knowledge such as the characteristics of vulnerable points and the oriented distribution of anomalies. A fuzzing test data generation method was proposed in [32] based on dynamic construction of mutation strategy. The method was designed to use the feedback information of instrumentation to dynamically construct the

control mutation strategy and the keyword mutation strategy and to guide the fuzzer to generate test data with high coverage. However, the test effect of this method is not ideal for the target program with large input. Dynamic construction mutation method needs repeated exploration of test data and program structure. If the test data is large, it will increase the exploration time and reduce the efficiency of test data generation. Lyu et al. [33] present a novel mutation scheduling scheme MOPT, which enables mutation-based fuzzers to discover vulnerabilities more efficiently. MOPT utilizes a customized Particle Swarm Optimization (PSO) algorithm to find the optimal selection probability distribution of operators with respect to fuzzing effectiveness and provides a pacemaker fuzzing mode to accelerate the convergence speed of PSO. Yue et al. [34] present a knowledge-learn evolutionary fuzzer based on AFL, which is called LearnAFL. LearnAFL does not require any prior knowledge of the application or input format. Based on our format generation theory, LearnAFL can learn partial format knowledge of some paths by analyzing the test cases that exercise the paths. Then LearnAFL uses this format information to mutate the seeds, which is efficient to explore deeper paths and reduce the test cases exercising high-frequency paths compared to AFL.

The main advantage of the mutation-based method is that this method does not need to understand the structure and format of the current sample file, so it can be widely used [25, 26]. The main disadvantage of the mutation-based method is that it is highly dependent on the initial samples. Different initial samples will bring different code coverage, test depth, and test effect, so the efficiency is low [25, 26].

2.3. Combination of Two Methods. Hodován et al. [35] present Grammarinator, a general-purpose test generator tool that is able to utilize existing parser grammars as models. Since the model can act both as a parser and as a generator, the tool can provide the capabilities of both generation-based and mutation-based fuzzers. The presented tool is actively used to test various JavaScript engines and has found more than 100 unique issues. Grammarinator can exploit the fact that the same grammar that can generate new tests can also be used to parse existing test suites and then create new content resulting from their recombination or mutation. The tool has proven its usefulness in the hardening of real-life projects by revealing more than 100 valid unique issues. Atlidakis et al. [36] introduced Pythia, the first fuzzer that augments grammar-based fuzzing with coverage-guided feedback and a learning-based mutation strategy for stateful REST API fuzzing. Pythia's mutation strategy helps generate grammatically valid test cases and coverage-guided feedback helps prioritize the test cases that are more likely to find bugs. Pythia is the first fuzzer that augments grammar-based fuzzing with coverage-guided feedback and a learning-based mutation strategy for stateful REST API fuzzing.

A new test case generation method based on the advantages of the above methods is proposed in this paper. Firstly, the characteristics of general transmission message of

industrial control protocol are analyzed, test cases are designed based on the construction of description model, and coding method of use cases is designed for genetic algorithm. Secondly, genetic algorithm is used to generate and optimize use cases, which realizes the automatic iteration and update of use case population. Finally, in order to improve test coverage and vulnerability discovery rate, the concept of dangerous point is proposed, and, based on this, a composite fitness function is designed to monitor and adjust the state of use case population.

3. Modbus Protocol Test Cases Design

3.1. Message Feature Analysis and Encoding. Choosing appropriate encoding method of use cases for protocol testing can reduce the time complexity of generating test cases and complete the conversion from encoding files to data packets faster. Figure 1 shows the data fields contained in the data packets of Modbus communication protocol and the byte length of each field [37].

In Modbus protocol packets, because the transmission identifier and protocol identifier are independent of the packets' content, these two fields cannot be considered when constructing test cases [38], so each test case can be mathematically expressed as in the following equation:

$$\text{case} = [l \ u \ f \ d], \quad (1)$$

where l is the length of the data field, and its value matches the data length contained in the following three fields. u is the address identifier, and value range is 0 to 255. f is the function code, which is divided into public function code and user-defined function code in Modbus, and its value range is 1 to 127. d is a data field, and the data information of this field depends on the function code.

When encoding test cases, binary encoding is the most common encoding method, and Hamming distance can be used to measure similarity between two test cases, as shown in the following equation:

$$d(A, B) = \sum_{i=0}^n A_i \oplus B_i, \quad (2)$$

where A_i and B_i denote the i -th characters of the strings A and B ; \oplus means to judge whether A_i and B_i are the same; when they are the same, $A_i \oplus B_i = 0$; when they are not, $A_i \oplus B_i = 1$.

However, when comparing the similarity of two test cases to calculate the Hamming distance, the Hamming cliff problem may occur [39]. Therefore, Gray code is used in this paper, which can effectively avoid the Hamming cliff problem and realize a more accurate description of the similarity of protocol packets. Assuming that there is a binary code of $B = B_n B_{n-1} B_{n-2} \dots B_1 B_0$ and its corresponding Gray code is $G = G_n G_{n-1} G_{n-2} \dots G_1 G_0$, then the value of the two codes satisfied the following equation:

Transaction identifier	Protocol identifier	Length	Unit identifier	Function code	Data
Byte 0/1	Byte 2/3	Byte 4/5	Byte 6	Byte 7	Other

FIGURE 1: Modbus data packets structure.

$$\begin{cases} G_i = B_i, & i = n, \\ G_{i-1} = B_{i-1} \oplus B_i, & i = 1, 2, \dots, n-1, \end{cases} \quad (3)$$

where G_i are the i -th bits of binary code and Gray code and \oplus is XOR operation.

Figure 2 shows the effect of clustering on the same set of data when calculating distance using two different encoding methods. It can be seen from the figure that some data may not be able to find the cluster center accurately when using binary code (Figure 2, left) to calculate the distance, while Gray code (Figure 2, right) can effectively avoid this problem.

In summary, Gray code avoids the Hamming cliff problem in binary coding, so the similarity between two Gray-coded strings can be described by the number of different bits, namely, Hamming distance.

3.2. Method for Calculating Similarity of Test Cases with Weights. In the Modbus protocol, the length of each field of the message sequence is basically fixed, but the length of the data storage field is dynamic, and the function of each field and the impact on the security of the message are different. Some fields are related to each other. If the Hamming distance is directly used as the similarity determination between the encoded strings of the two test cases, there is a certain irrationality.

In order to solve these problems, a weight distance calculation method based on internal classification is proposed in this paper. The weight of different fields is set in different value, and the distances of different fields are calculated according to corresponding functions. The data segment is special, because it is related to other fields, and a unique design is required to calculate the relevant distance. The weight coefficient of each field is determined by Analytic Hierarchy Process (AHP).

Assuming that there are test cases A and B , first calculate the corresponding distances of each functional field of them, then combine the weights of the fields, and calculate their overall similarity. The final calculation formula is shown in following equation:

$$\text{dis}_{AB} = \sum_{i=0}^4 w_i \cdot d(A_{vi}, B_{vi}), \quad (4)$$

where $w = [w_1, w_2, w_3, w_4]$ is the weight of each field. A_{vi} and B_{vi} are the corresponding fields of the two test cases, and $d(A_{vi}, B_{vi})$ is the distance between the two corresponding fields. The distance calculation method for different fields is slightly different.

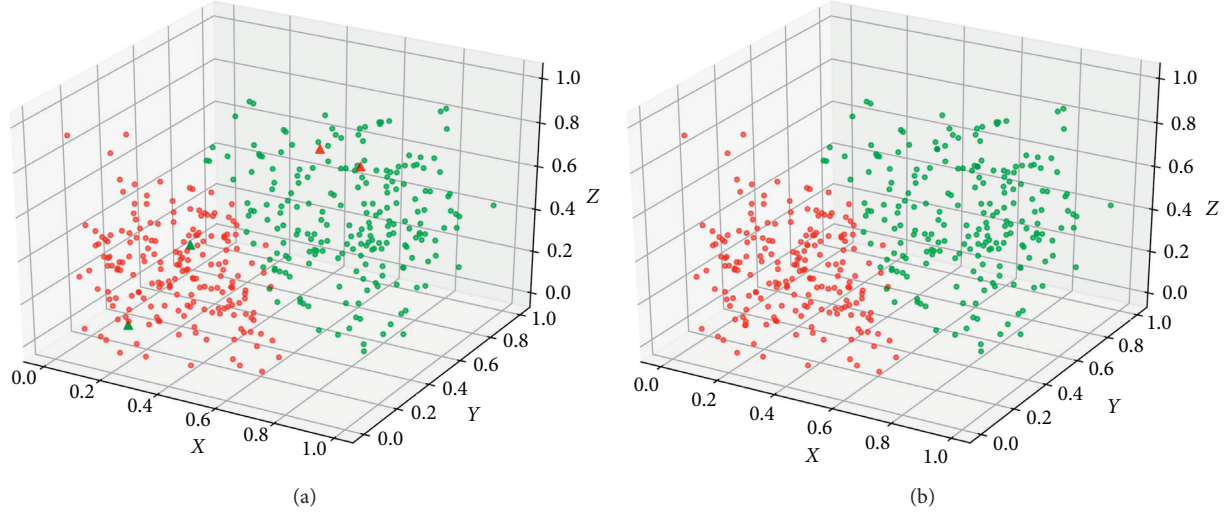


FIGURE 2: Clustering results of two encoding methods.

The pairwise comparison matrix determined by Analytic Hierarchy Process is shown in the following equation:

$$A = \begin{bmatrix} 1 & 3 & \frac{1}{3} & \frac{1}{2} \\ \frac{1}{3} & 1 & \frac{1}{5} & \frac{1}{3} \\ 3 & 5 & 1 & 3 \\ 2 & 3 & \frac{1}{3} & 1 \end{bmatrix}. \quad (5)$$

Consistency of the pairwise comparison matrix was checked. If test coefficient $CR = 0.0386 < 0.9$, then consistency check is passed. The calculated weight of each field is shown in the following equation:

$$w = [0.1682 \ 0.0769 \ 0.5167 \ 0.2382]. \quad (6)$$

According to the characteristics of the Modbus test cases, the length of length field, address identifier field, and function code field are fixed, while the length of the data field is dynamically variable and is associated with other fields. Therefore, when calculating the distance between the corresponding fields of the two use cases, two different methods are used to calculate the distance of the fixed-length and variable-length fields. For fixed-length fields, the Hamming distance can be directly used.

The length of the data field is dynamically variable. When describing the distance, Hamming distance will have a large deviation, and Levenshtein distance can solve this problem. Levenshtein distance is to find the minimum number of transformations required to convert string A to string B. It can more describe the difference between two strings of different lengths accurately. The calculation method is shown in the following equation:

$$\text{lev}_{A,B}(i, j) = \begin{cases} \max(i, j), & \min(i, j) = 0, \\ \min \begin{cases} \text{lev}_{A,B}(i-1, j) \\ \text{lev}_{A,B}(i, j-1) \\ \text{lev}_{A,B}(i-1, j-1) + 1_{A_i \neq B_j} \end{cases}, & \min(i, j) \neq 0, \end{cases} \quad (7)$$

where i and j are the subscripts of string A to string B. $\max(i, j)$ is the maximum value. $\min(i, j)$ is the minimum value.

Therefore, the similarity calculation equation (4) of the two test cases can be further optimized into the following equation:

$$\text{dis}_{AB} = \sum_{i=1}^3 w_i \cdot d(A_{vi}, B_{vi}) + \text{lev}_{A_d, B_d}(m, n), \quad (8)$$

where $\text{lev}_{A_d, B_d}(m, n)$ is the Levenshtein distance between the two data fields of m and n .

4. Average Similarity and Population Dispersion of Test Cases

In the test case generation process, the iteration is based on the population, so it is necessary to describe first-generation population from the perspective of the whole population. Here, the average similarity of population test cases is designed to describe the population state. The average similarity of test cases refers to the overall degree of dispersion among individuals in a population. When the average similarity of test cases is low, it means that the overall similarity of individuals within the population is too high, and the coverage of test cases is low [40]. At this time, the parameter information in the test cases generation process, such as the mutation probability and the similarity threshold, can be appropriately changed to adjust the

distribution of the generated test cases and improve the coverage of the test cases.

When describing the average similarity of test cases of individuals in the entire population, it can be described by the average distance between individuals. This method is feasible to some extent, but each individual needs to calculate the distance between itself and all other individuals. As a result, this method has a lot of repeated calculation and low efficiency. In addition, if an extremely uniform edge distribution occurs, it will also lead to misjudgment. Therefore, the concept of average similarity of test cases is proposed in this paper, and a new calculation method is designed to accurately reflect the distribution of individuals in the population and reduce the amount of calculation.

Firstly, values of individual fields in the population are normalized, which is expressed mathematically in the following equation:

$$v_n = \frac{c_n - c_{n_min}}{c_{n_max} - c_{n_min}}, \quad (9)$$

where c_{n_max} is the maximum value of the field in the population; c_{n_min} is the minimum value of the field in the population.

The sum of each field is averaged to calculate the mean center test case, as shown in equation (10), and the calculation method of each field is as in equation (11).

$$\overline{\text{case}} = [\overline{l_v} \ \overline{u_v} \ \overline{f_v} \ \overline{d_v}], \quad (10)$$

$$\overline{v} = \frac{1}{m} \cdot \sum_{i=1}^m v_i, \quad (11)$$

where m is the total number of test cases in the population and v_i is the current field of the test cases.

The similarity between the test cases and the central test case can be used to indicate the outlier degree of the test cases, as shown in the following equation:

$$s = \sum_{i=1}^3 w_i \cdot d(A_{vi}, \overline{C_{vi}}) + \text{lev}_{A_{vi}, \overline{C_{vi}}}(m, n). \quad (12)$$

The calculation time complexity of the average similarity of the test cases is $2n$; compared with the time complexity $n \lg n$ of the general method, there will be a significant efficiency improvement when n is larger. Then the dispersion of the whole population can be described by the following equation:

$$\text{sca} = \frac{1}{n} \cdot \sum_{i=1}^n s_i. \quad (13)$$

5. Experimental Evaluation

By designing the encoding method and the similarity calculation method between test cases, combined with the description of the average similarity of test cases in the test cases population, theoretically, it can effectively improve the efficiency of test cases generation and increase the coverage of test cases. In order to verify the correctness of the proposed method, a set of comparative experiments are

designed, and genetic algorithm is used as the core algorithm for test case generation. The encoding method, individual similarity, and average similarity of test cases are calculated by the proposed method and the conventional method, respectively, and the test cases generated by the two methods are compared and analyzed.

Genetic algorithm is an intelligent optimization algorithm, which is often used to find the global optimal solution, and we adjust the population optimization direction by designing the corresponding fitness function. In the test case generation method designed in this paper, the population convergence direction of genetic algorithm is a suspicious case in historical data. Suspicious test cases are cases that cause test target anomalies during the test process. Taking these cases as the convergence center of next genetic algorithm can effectively reduce the randomness of test case generation. These test cases are called ‘‘suspicious points.’’ Based on this, the fitness function of the genetic algorithm designed for two sets of experiments is shown in the following equation:

$$\begin{cases} f_p(s_A) = 1 - \frac{s_A}{s_{\max}}, & \text{suspicious points exist,} \\ f_p(A) = 1 - \frac{\text{dis}(dp_p, A_i)}{\max(\text{dis}(dp_p, A_i))}, & \text{else,} \end{cases} \quad (14)$$

where dis is the similarity between the test case and the suspicious point; the calculation method is shown in formula (3). s_A is the average similarity of the test case.

The meaning of fitness function is that when there are suspicious points in the population, the population converges to the suspicious case. When there is no suspicious point, the population with higher average similarity of test cases is preferred. Other parameter settings of genetic algorithm are mutation probability $P_m = 0.2$ and crossover probability $P_c = 0.6$.

The whole experimental procedure designed is shown in Figure 3. Firstly, the initial test case population for the two experiments is constructed manually, and the initial population is encoded according to the encoding method mentioned above. Secondly, the initial population is input into the test case generation module, and two different fitness function calculation methods are used to generate and optimize the test cases. Finally, the result monitoring module records the operation results.

The script development language of the experiment is *Python 3*, and Modbus communication simulation software used in the test is Modbus Poll and Modbus Slave. Firstly, Modbus Poll is used to establish data communication with Modbus Slave, Wireshark packet capture tool is used to obtain normal communication messages, and representative data messages are selected to analyze the data characteristics and construct the initial population. Secondly, the initial population is sent to the test cases generation and optimization module to iterate, optimize, and update test cases. Finally, each generation of population is sent to the target for testing. Statistical analysis was performed on the test cases

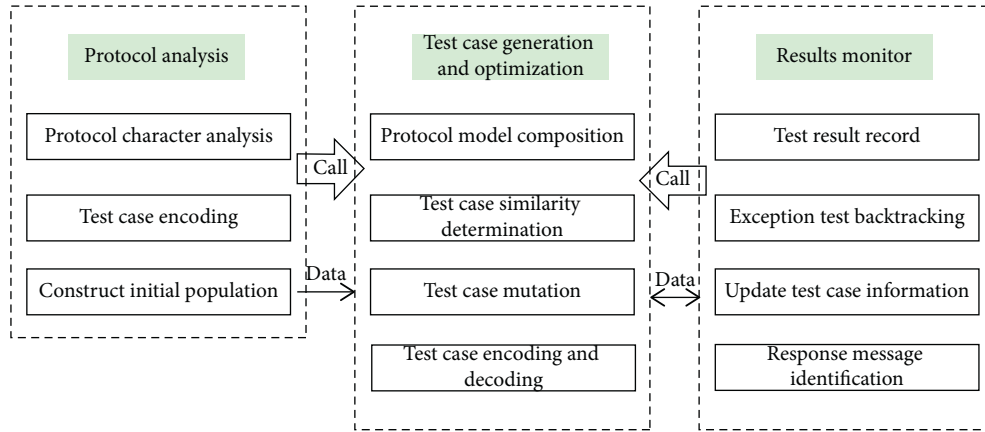


FIGURE 3: The whole experimental procedure designed.

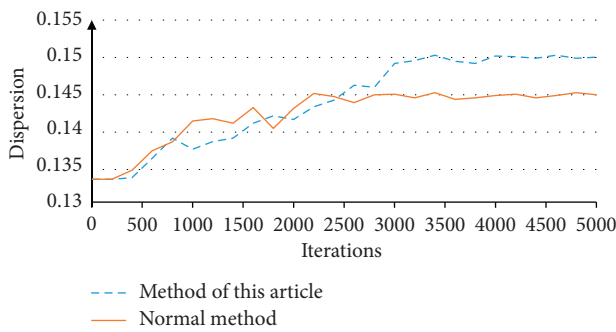


FIGURE 4: Trend of population dispersion.

data generated by the two methods. During the experiment, the average similarity of the first 5000 generations of population test cases was calculated. The results are shown in Figure 4.

In two groups of experiments using different methods, during the population iteration process, the dispersion gradually increased and eventually stabilized. At the beginning of the experiment, since the same initial population was used, the dispersions of two groups were the same. However, with the iteration of the population, when both of them are stable, the dispersion of the population produced by the improved method is 3.45%, which is higher than that of the conventional method. It is generally believed that the higher the dispersion between individuals within a population, the higher the coverage of test cases [21]. Therefore, it can be considered that the coverage of test cases generated by the improved method is higher than the conventional method, and it also proves that the method proposed in this article has certain advantages over the conventional method. Based on the proposed method, we design a fuzzy tester [41].

6. Conclusion

A new test cases similarity determination method and the concept of population dispersion are proposed in this paper, which provides a new idea and method for improving the test cases coverage in the protocol testing process. In the determination of test cases similarity, different weights and distance calculation methods are set according to different

protocol fields, which can more accurately determine the similarity according to the function of the test cases and data content, and the change of the encoding method effectively resolves the problem of inaccurate similarity determination caused by data mutation. The genetic algorithm is introduced into the test cases generation algorithm, and the test cases similarity and population dispersion are used as the basis for constructing the fitness function of the genetic algorithm, and the automatic optimization of the test cases generation is realized. The test cases data generated in the experiment shows the effectiveness of the method. Our planned future work is twofold. First, we plan to improve the applicability of the method and apply it to the generation of test cases for other protocols. Second, we plan to optimize the time complexity of the algorithm.

Data Availability

The data used to support the findings of this study have not been made available because the generated test cases were not backed up in time.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by “National Key R&D Program of China” (2018YFB2004200), the open project of Zhejiang Lab “Construction Technology of Local High Security Trusted Execution Environment for Edge Intelligent Controller” (2021KF0AB06), and the National Natural Science Foundation of China “Research on anomaly detection and security awareness method for industrial communication behaviours” (61773368).

References

- [1] O. E. Idrissi, A. Mezrioui, and A. Belmekki, “Cyber security challenges and issues of industrial control systems—some security recommendations,” in *Proceedings of the IEEE*

- International Smart Cities Conference (ISC2)*, pp. 330–335, Casablanca, Morocco, April 2019.
- [2] M. R. Asghar, Q. Hu, and S. Zeadally, “Cybersecurity in Industrial control systems: issues, technologies, and challenges,” *Computer Networks*, vol. 165, Article ID 106946, 2019.
 - [3] T. Miyachi and T. Yamada, “Current issues and challenges on cyber security for industrial automation and control systems,” in *Proceedings of the SICE Annual Conference (SICE)*, pp. 821–882, Sapporo, Japan, October 2014.
 - [4] D. Myers, K. Radke, S. Suriadi, and E. Foo, “Process discovery for industrial control system cyber attack detection,” *ICT Systems Security and Privacy Protection 2017, Proceedings (IFIP Advances in Information and Communication Technology)*, Springer, vol. 502, pp. 61–75, Switzerland, 2017.
 - [5] C. Lin, S. Wu, and M. Lee, “Cyber attack and defense on industry control systems,” in *Proceedings of the IEEE Conference on Dependable and Secure Computing*, pp. 524–526, Taipei, Taiwan, August 2017.
 - [6] S. M. Ghaffarian and H. R. Shahriari, “Software vulnerability analysis and discovery using Machine-Learning and Data-Mining techniques,” *ACM Computing Surveys*, vol. 50, no. 4, pp. 1–36, 2017.
 - [7] Y. X. Lai, H. Gao, and J. Liu, “Vulnerability mining method for the modbus TCP using an anti-sample fuzzer,” *Sensors*, vol. 20, no. 7, p. 2040, 2020.
 - [8] C. Wang, Q. Li, X. H. Wang et al., “An android application vulnerability mining method based on static and dynamic analysis,” in *Proceedings of the IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, IEEE, June 2020.
 - [9] T. Tu, H. Zhang, B. Qin et al., “A vulnerability mining system based on fuzzing for IEC 61850 protocol,” *Advances in Engineering Research (AER) in Proceedings of the 5th international conference on frontiers of manufacturing science and measuring technology (FMSMT 2017)*, vol. 130, pp. 589–597, Taiyuan, China, June 2017.
 - [10] W.-N. Kim, M.-S. Jang, J. Seo, and S. Kim, “Vulnerability discovery method based on control protocol fuzzing for a railway SCADA system,” *The Journal of Korea Information and Communications Society*, vol. 39C, no. 4, pp. 362–369, 2014.
 - [11] S. J. Kim and T. Shon, “Field classification-based novel fuzzing case generation for ICS protocols,” *Journal of Supercomputing*, vol. 74, no. 9, 2018.
 - [12] T. Wang, Q. Xiong, H. Gao et al., “Design and implementation of fuzzing technology for OPC protocol,” in *Proceedings of the Ninth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 424–428, Beijing, China, October 2013.
 - [13] X. Zhang and Z. J. Li, “Overview of fuzzy testing technology,” *Computer Science*, vol. 43, no. 5, pp. 1–8, 2016, in Chinese.
 - [14] T. L. Munea, H. Lim, and T. Shon, “Network protocol fuzz testing for information systems and applications: a survey and taxonomy,” *Multimedia Tools and Applications*, vol. 75, no. 22, pp. 14745–14757, 2016.
 - [15] T. Kitagawa, M. Hanaoka, and K. Kono, “AspFuzz: a state-aware protocol fuzzer based on application-layer protocols,” in *Proceedings of the IEEE Symposium on Computers and Communications*, pp. 202–208, Riccione, Italy, June 2010.
 - [16] Y. J. Zhang, Z. J. Li, X. K. Liao et al., “Survey of automated whitebox fuzz testing,” *Computer Science*, vol. 41, no. 2, pp. 7–10, 2014.
 - [17] M. B. Cohen, J. Snyder, and G. Rothermel, “Testing across configurations,” *ACM SIGSOFT Software Engineering Notes*, vol. 31, no. 6, pp. 1–9, 2006.
 - [18] E. Martins, S. B. Sabiao, and A. M. Ambrosio, “ConData: a tool for automating specification-based test case generation for communication systems,” in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, vol. 8, Maui, HI, USA, January 2000.
 - [19] G. Banks, M. Cova, V. Felmetzger et al., “SNOOZE: Toward a stateful network protocol fuzzer,” in information security,” in *Proceedings of the International Conference, Isc, Samos Island, Greece, Samos Island, Greece, August 2006*.
 - [20] W.-M. Li, A.-F. Zhang, J.-C. Liu, and Z.-T. Li, “An automatic network protocol fuzz testing and vulnerability discovering method,” *Chinese Journal of Computers*, vol. 34, no. 2, pp. 242–255, 2011, in Chinese.
 - [21] A. G. Voyiatzis, K. Katsigiannis, and S. Koubias, “A Modbus/TCP Fuzzer for testing internetworked industrial systems,” in *Proceedings of the IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, pp. 1–6, Berlin, Germany, September 2015.
 - [22] J. J. Liu and Y. D. Yuan, “Research on network protocol fuzzy test case generation method based on heuristic search and classification tree,” *Modern Electronics Technique*, vol. 39, no. 21, pp. 36–39, 2016, in Chinese.
 - [23] A. Felix, A. F. Tappenden, and J. Miller, “Policy generator (PG): a heuristic-based fuzzer,” in *Proceedings of the 49th Hawaii International Conference on System Sciences (HICSS)*, pp. 5535–5544, Koloa, HI, USA, March 2016.
 - [24] H. X. Wang, C. Y. Zhu, H. Ying et al., “A fuzzy testing method of industrial control protocol based on reverse analysis,” *Electric Power Information and Communication Technology*, vol. 17, no. 4, pp. 5–13, 2019, in Chinese.
 - [25] C. Miller and Z. Peterson, “Analysis of mutation and generation-based fuzzing,” 2007, <https://www.defcon.org/images/defcon-15/dc15-presentations/Miller/Whitepaper/dc-15-miller-WP.pdf>.
 - [26] K. Chen, C. Song, L. M. Wang et al., “Using memory propagation tree to improve performance of protocol fuzzer when testing ICS,” *Computers & Security*, vol. 87, Article ID 101582, 2019.
 - [27] S. J. Gu, Y. Y. Song, X. Zhao et al., “Fuzzing test data generation based on message matrix perturbation with keyword reference,” in *Proceedings of the IEEE MILCOM 2011 Military Communications Conference*, pp. 1115–1120, Baltimore, MD, USA, November 2011.
 - [28] K. Wang, Y. Q. Zhang, Q. X. Liu et al., “A fuzzing test for dynamic vulnerability detection on Android Binder mechanism,” in *Proceedings of the IEEE Conference on Communications and Network Security (CNS)*, pp. 709–710, Florence, Italy, December 2015.
 - [29] Y. X. Lai, K. X. Yang, J. Liu et al., “Vulnerability mining method for industry control network protocol based on fuzzing test,” *Computer Integration Manufacturing System*, vol. 25, no. 9, pp. 2265–2279, 2019, in Chinese.
 - [30] J. Cai, Q. Li, Y. Chen, Y. Liu, Y. Xia, and S. Rahmany, “Troubleshooting test method based on industrial control grammar model,” in *Proceedings of the IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, pp. 404–409, New York, NY, USA, August 2019.

- [31] P. Xu, J. Y. Liu, B. Lin et al., “Generation of fuzzing test case based on recurrent neural networks,” *Application Research of Computers*, vol. 36, no. 9, pp. 2679–2685, 2019, in Chinese.
- [32] L. L. Jiao, S. L. Luo, W. Cao et al., “Fuzzing test data generation method based on dynamic construction of mutation strategy,” *Transactions of Beijing Institute of Technology*, vol. 39, no. 5, pp. 539–544, 2019, in Chinese.
- [33] C. Y. Lyu, S. L. Ji, C. Zhang et al., “MOPT: optimized mutation scheduling for fuzzers,” in SEC’19,” in *Proceedings of the 28th USENIX Conference on Security Symposium*, pp. 1949–1966, Berkeley, CA; USA, August 2019.
- [34] T. Yue, Y. Tang, B. Yu, P. Wang, and E. Wang, “Learn AFL: greybox fuzzing with knowledge enhancement,” *Institute of Electrical and Electronics Engineers Access*, vol. 7, pp. 117029–117043, 2019.
- [35] R. Hodován, K. Kiss, and T. Gyimóthy, “Grammarinator: a grammar-based open source fuzzer,” in *Proceedings of the 9th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation*, pp. 45–48, Lake Buena Vista, FL, USA, November 2018.
- [36] V. Atlidakis, R. Geambasu, P. Godefroid et al., “Pythia: Grammar-Based Fuzzing of REST APIs with Coverage-Guided Feedback and Learning-Based Mutations,” 2020, <https://arxiv.org/pdf/2005.11498v1.pdf>.
- [37] I. N. Fovino, A. Carcano, M. Masera et al., “Design and implementation of a secure modbus protocol,” in *Proceedings of the International Conference on Critical Infrastructure Protection*, pp. 33–36, Springer, Arlington, VA, USA, March 2009.
- [38] J. Luswata, P. Zavorsky, B. Swar et al., “Analysis of SCADA security using penetration testing: a case study on Modbus TCP protocol,” in *Proceedings of the 29th Biennial Symposium on Communications (BSC)*, Toronto, CA, USA, June 2018.
- [39] R. A. Caruana and J. D. Schaffer, “Representation and hidden bias: Gray vs. Binary coding for genetic algorithms,” in *Proceedings of the Fifth International Conference on Machine Learning*, pp. 153–161, Ann Arbor, MI, USA, June 1988.
- [40] A. Arrieta, S. Wang, U. Markiegi et al., “Search-based test case generation for Cyber-Physical Systems,” in *Proceedings of the Institute of Electrical and Electronics Engineers Congress on Evolutionary Computation (CEC)*, pp. 688–697, Donostia-San Sebastian, Spain, June 2017.
- [41] G. Zhang, W. Shang, B. Zhang, C. Chunyu, and Z. Rui, “Fuzzy test method for industrial control protocol combining genetic algorithm,” *Computer Application Research*, vol. 38, no. 3, 2021, in Chinese.