electronics

Article

Fuzzing Technology Based on Information Theory for Industrial Proprietary Protocol

Xin Che [1], Yangyang Geng [2], Ge Zhang [3] and Mufeng Wang [3,*]

[1] College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China; chexin@zju.edu.cn
[2] State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China; young9471@163.com
[3] China Industrial Control Systems Cyber Emergency Response Team, Beijing 100040, China; zhangge@cics-cert.org.cn
* Correspondence: wangmufeng@cics-cert.org.cn

Abstract: With the rapid development of the Industrial Internet of Things (IIoT), programmable logic controllers (PLCs) are becoming increasingly intelligent, leading to improved productivity. However, this also brings about a growing number of security vulnerabilities. As a result, efficiently identifying potential security vulnerabilities in PLCs has become a crucial research topic for security researchers. This article proposes a method for fuzzing industrial proprietary protocols to effectively identify security vulnerabilities in PLCs' proprietary protocols. The aim of this study is to develop a protocol fuzzing approach that can uncover security vulnerabilities in PLCs' proprietary protocols. To achieve this, the article presents a protocol structure parsing algorithm specifically designed for PLC proprietary protocols, utilizing information theory. Additionally, a fuzzing case generation algorithm based on genetic algorithms is introduced to select test cases that adhere to the format specifications of the proprietary protocol while exhibiting a high degree of mutation. The research methodology consists of several steps. Firstly, the proposed protocol structure parsing algorithm is used to analyze two known industrial protocols, namely Modbus TCP and S7Comm. The parsing results obtained from the algorithm are then compared with the correct results to validate its effectiveness. Next, the protocol structure parsing algorithm is applied to analyze the proprietary protocol formats of two PLC models. Finally, based on the analysis results, the PLCs are subjected to fuzzing. Overall, the proposed protocol fuzzing approach, incorporating the protocol structure parsing algorithm and the fuzzing case generation algorithm, successfully identifies two denial-of-service vulnerabilities in the PLCs' proprietary protocols. Notably, one of these vulnerabilities is a zero-day vulnerability, indicating that it was previously unknown and undisclosed.

Keywords: industrial proprietary protocol; fuzzing test; information theory; genetic algorithm

check for updates

Citation: Che, X.; Geng, Y.; Zhang, G.; Wang, M. Fuzzing Technology Based on Information Theory for Industrial Proprietary Protocol. *Electronics* **2023**, *12*, 3041. https://doi.org/10.3390/electronics12143041

Academic Editors: Shibo He, Fangyuan Xing, Victor C. M. Leung, Lei Yang, Huan Zhou and Nurul I. Sarkar

Received: 16 May 2023
Revised: 15 June 2023
Accepted: 21 June 2023
Published: 11 July 2023

1. Introduction

With the increasing number of attacks on industrial control systems (ICS), the exploration of potential vulnerabilities to enhance the security of ICS has garnered attention from security researchers [1,2]. Common methods of industrial vulnerabilities mining include programming software structure analysis, vulnerability mining, and protocol fuzzing testing [3]. However, programming software structure analysis has drawbacks, such as high workload and coverage deficiencies, despite its ability to parse protocol formats and communication processes and identify security vulnerabilities. In addition, there are no mature automatic vulnerability mining tools and there is too much work to rely only on manual reverse engineering. With the continuous increase in the scale and complexity of software, fuzzing technology, as a simple and efficient vulnerability mining technology, has incomparable advantages over other vulnerability mining technologies, and has gradually become the focus of researchers [4].

Fuzzing is a vulnerabilities mining method of potential software through unexpected inputs and abnormal results monitoring. Protocol fuzzing sends constructed malformed packets to the industrial equipment and monitors industrial equipment abnormality, so as to excavate the security vulnerabilities of the industrial equipment [5,6]. For public protocols with known structures, researchers perform fuzzing test on one or more public protocols by secondary development on the basis of existing tools [7,8]. Considering that the existing research on fuzzing relates to public industrial protocols, and the existing techniques are powerless against those proprietary protocols whose structures have not been disclosed, security researchers began to concern themselves with how to fuzz proprietary protocols with undisclosed structures [4,9].

The structure analysis of industrial proprietary protocols can be divided into the following aspects: protocol format inference, protocol semantic analysis, and protocol state machine construction [10,11]. Among them, the boundaries divide the packets into fields, protocol semantic analysis is used to parse the function of each field, and a protocol state machine is constructed to parse the protocol state transition constraints in the protocol communication. The existing protocol structure analysis can be divided into network traffic-based and program-based analysis according to the objects and technologies. Script-Gen [12] first filters the traffic data, and then uses the sequence alignment algorithm to perform macro-clustering and micro-clustering of packet fields on the traffic data. Macro-clustering is used to complete field boundary partitioning. On the basis of field partitioning, micro-clustering uses the threshold value to parse field semantics, and then the results of field partitioning and semantic parsing are used for state machine generation. The IPART algorithm [13] first selects high-frequency fields through the N-Gram algorithm and then uses the global voting algorithm to divide field boundaries. In protocol semantic analysis [14], field recognition algorithms are designed according to the characteristics of fields such as constant, length field, serial number, and function code. Pacheco [15] combines sequence alignment algorithms and dynamic stain analysis techniques for protocol reverse engineering. This work first uses the dynamic stain analysis method to parse out the field boundary, and then uses the sequence alignment algorithm to optimize the inference result of the field boundary in the previous step. However, the existing fuzzing methods require a known protocol format, and they cannot perform effective fuzzing on proprietary protocols with unknown formats.

In the fuzzing test of protocols, fuzzing tools such as Peach [16] are used to model the protocol format and build a state machine. After completing the configuration, fuzzing tools will automatically generate test cases for the devices fuzzing test. The advantage of this method is that fuzzing testing can be carried out automatically after configuration, which can reduce the workload of security personnel. At the same time, fuzzing can be carried out on all industrial equipment using the specified protocols, which is applicable to a wide range. Lai et al. designed a fuzzer [16] to mine the possible security vulnerabilities of Modbus/TCP. In the test, a detection phase is added for better predictive ability, which greatly reduces the randomness of fuzzing. Shen et al. designed a fuzzing tool, LZFuzz [17], for the Scada system. They used the LZ comparison algorithm to parse the key feature domain of network packets, marked the newly parsed types, updated the marked list, and sent the network message and its tag type to the GPF mutator for the key fields. However, the existing fuzzing research on proprietary protocols does not consider the construction of targeted test sets for different types of vulnerabilities when constructing test cases, resulting in a low test success rate. In addition, the existing research does not consider the redundancy of test cases; Peach may produce a large number of redundant malformed packets, thus leading to low efficiency of fuzzing.

Based on the characteristics of industrial protocol fuzzing, this study highlights the significance of identifying security vulnerabilities in PLCs' proprietary protocols due to the increasing intelligence of IIoT systems. By developing and applying the protocol fuzzing approach, this study demonstrates an effective method for detecting vulnerabilities in PLCs and contributes to enhancing the security of industrial systems. Major contributions include:

- We propose a structure analysis algorithm for an industrial proprietary protocol based on mutual information rate. From the point of information theory, we introduce information entropy and mutual information and put forward three judgment conditions for mutual information rate that adjacent bytes do not belong to the same field to complete the analysis of proprietary protocol structure, so as to support fuzzing test of proprietary protocols with unknown formats.
- We propose a fuzzing test case generation method for an industrial proprietary protocol based on a genetic algorithm. First, the Smith–Waterman algorithm is used to evaluate the similarity between test cases and normal packets, and then the fitness value of test cases is calculated to screen out proprietary protocol fuzzing test cases that satisfy the proprietary protocol structure and have a high degree of variation.
- We design an industrial protocol fuzzing test system. In the experiments of fuzzing on PLC, which supports the Modbus TCP protocol and unknown proprietary protocol, two denial-of-service vulnerabilities are discovered, including one original CNVD vulnerability. Experimental results confirm the feasibility of the proposed structure analysis method and the efficiency of vulnerability mining.

The rest of this paper is organized as follows. Section 2 provides details on the structure analysis algorithm of proprietary protocol. Section 3 analyses the proprietary protocol fuzzing test case generation algorithm. To clearly demonstrate how to calculate the evaluation function, Section 4 presents a concrete example and shows the efficiency of the proposed vulnerability mining technique for PLCs. Section 5 concludes the paper.

2. Proprietary Protocol Structure Analysis Algorithm

The difficulty of industrial proprietary protocol fuzzing is that the structure of the proprietary protocol is not disclosed, which requires the tester to analyze the protocol structure through various methods, and the analysis result of the protocol structure will affect the acceptance rate of test case.

In industrial protocols, the value range of each field has its own characteristics because of the different functions of each field. Taking the Modbus TCP as an example, its protocol identifier field is used to indicate that the data packet is the Modbus TCP protocol data packet, thus the value of this field is always 0×0000, and its length field is used to represent the byte length of the unit identifier of the data packet plus, so the value of the length field often changes randomly. Therefore, each field carries different amount of information due to different functions. Considering this characteristic, information theory is used to analyze the structure of the industrial proprietary protocol.

2.1. Proprietary Protocol Structure Analysis Algorithm Based on Mutual Information Rate

In this section, the mutual information rate is proposed to analyze the structure of industrial proprietary protocols from the perspective of information entropy and mutual information.

2.1.1. Information Entropy

Information entropy [18] is used to measure the mathematical expectation of the amount of information brought by the occurrence of a random event, as shown in Equation (1).

$$H(B_i) = -\sum_{b_i \in B_i} p(b_i) \log p(b_i) \tag{1}$$

Figure 1 shows the distribution of information entropy of each byte of a Modbus TCP protocol packet. As shown in Figure 2, the first two bytes are the transaction identifier, and the value of the transaction identifier increases with each client request, so the information entropy of this field is high, and the 0th byte is the high-order byte of the transaction identifier, and the 1st byte is the low-order byte. Thus, the information entropy of the 0th byte will be lower than the information entropy of the 1st byte.

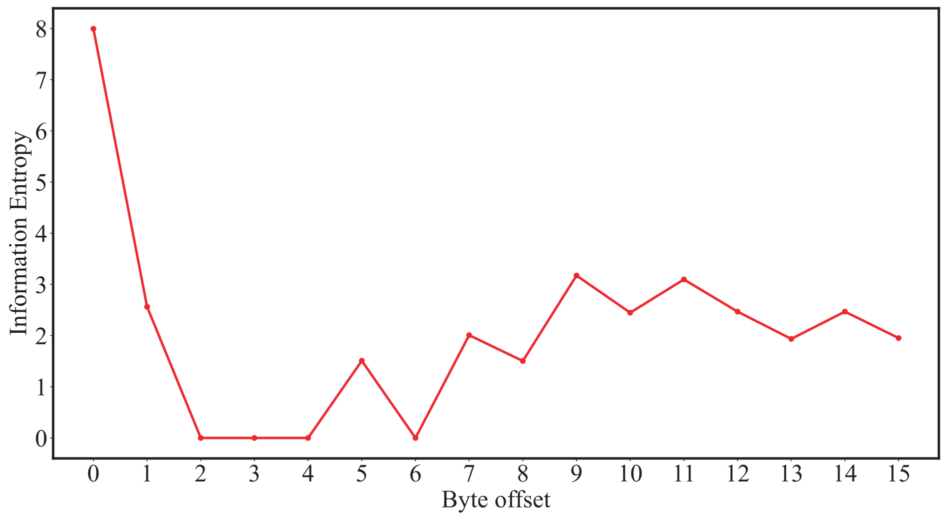


Figure 1. Information entropy distribution of each byte of Modbus TCP packets.

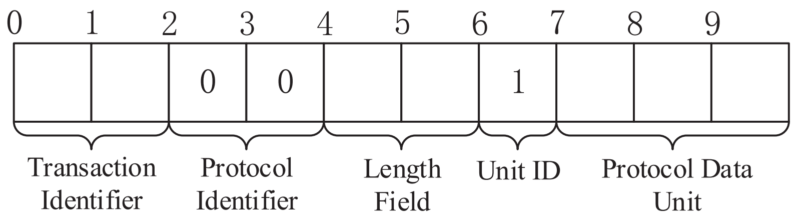


Figure 2. Modbus TCP format.

2.1.2. Mutual Information

When parsing the protocol structure, the mutual information between the ith and $i+1$th bytes of packet set P is calculated by Equation (2). If adjacent bytes belong to the same field, the values of the two bytes may have high correlation, so their values of mutual information are high. If adjacent bytes belong to different fields, the values of the two bytes have no correlation, and their values of mutual information are low.

$$I(B_i; B_{i+1}) = \sum_{b_i \in B_i} \sum_{b_j \in B_{i+1}} p(b_i, b_j) \log \frac{p(b_i, b_j)}{p(b_i) p(b_j)} \tag{2}$$

Figure 3 shows the mutual information distribution of two adjacent bytes of the same Modbus TCP packet. The first two adjacent bytes belong to the transaction identifier field. However, due to the fact that the PCAP sample files chosen for this study were pre-filtered, there is a significant difference in the numerical values of the transaction identifier field between adjacent requests. As a result, in Figure 3, the entropy of the first byte is higher than the entropy of the second byte. The second adjacent bytes are the 2nd byte of the transaction identifier and the 1st byte of the protocol identifier. The values of the two bytes have no correlation, so the mutual information value is 0. The 3rd adjacent bytes belong to the protocol identifier field, because in Modbus TCP packets the value of this field is constant 0, so the mutual information value of the 3rd adjacent byte is 0. The adjacent bytes in the eighth place are the 1st byte of the function code field and the data address field. Because the value range of different function codes and the corresponding data address has a great correlation, the mutual information of these two bytes is relatively high.

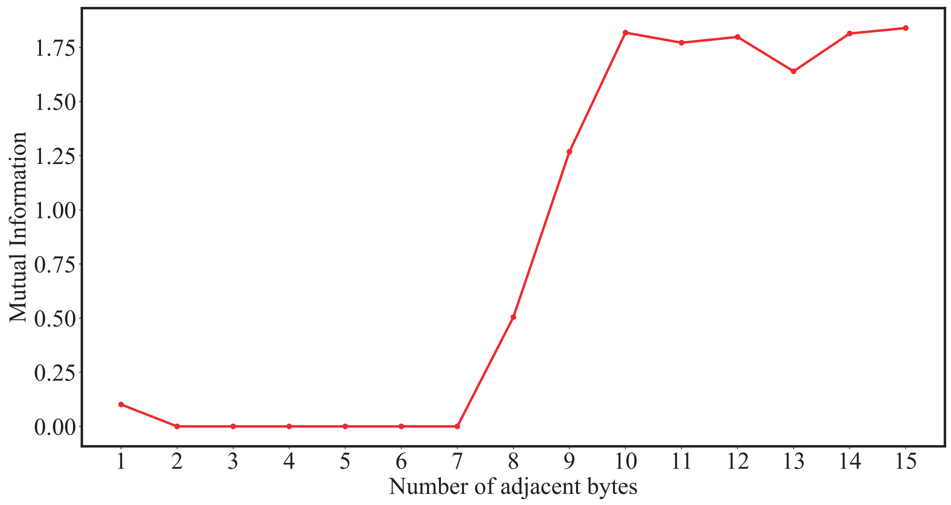


Figure 3. Mutual information distribution of adjacent bytes of Modbus TCP packets.

2.1.3. Mutual Information Rate

Joint entropy is used to measure the uncertainty degree of a jointly distributed random system, as shown in Equation (3). The joint entropy has the properties demonstrated in Equation (4).

$$\begin{aligned} H(B_i, B_{i+1}) &= -\sum_{b_i \in B_i} \sum_{b_j \in B_{i+1}} p(b_i, b_j) \log p(b_i, b_j) \\ &= H(B_i) + H(B_{i+1}) - I(B_i; B_{i+1}) \end{aligned} \tag{3}$$

$$\max\{H(B_i), H(B_{i+1})\} \leq H(B_i, B_{i+1}) \leq H(B_i) + H(B_{i+1}) \tag{4}$$

According to the information entropy and the mutual information of adjacent bytes of the Modbus TCP protocol packet, it can be seen that there will be large errors in the structure analysis of the proprietary protocol, only relying on the information entropy and mutual information. In order to analyze the proprietary protocol more accurately, this paper proposes a new indicator mutual information rate, which is calculated based on information entropy and mutual information, as shown in Equation (5). This section evaluates the degree of dependence between two adjacent bytes based on the mutual information rate. The closer the mutual information rate is to 1, the higher the degree of dependence between the two adjacent bytes, indicating that the two bytes belong to the same field. Conversely, if the mutual information rate is close to 0, the lower the degree of the dependency, the higher the possibility that these two bytes belong to different fields.

$$MIR(B_i, B_{i+1}) = \begin{cases} 1, & H(B_i) = 0 \ \ and \ \ H(B_{i+1}) = 0 \\ \frac{I(B_i; B_{i+1})}{H(B_i, B_{i+1})}, & H(B_i) \neq 0 \ and \ H(B_{i+1}) \neq 0 \\ 0, & \text{else} \end{cases} \tag{5}$$

When the values of $H(B_i)$ and $H(B_{i+1})$ are both 0, it indicates that the ith byte and the $i+1$th byte are fixed constants. At this point, the dependence degree of the two bytes is maximum, so the mutual information rate of the two bytes is 1. When only one of the values of $H(B_i)$ and $H(B_{i+1})$ is 0, it indicates that the value of one byte is constant, and the value of the other byte will change randomly, indicating that the value of the two bytes has no dependence. At this point, the dependence of the two bytes is minimum defined, so the mutual information rate of the two bytes is 0. When neither the value of $H(B_i)$ nor $H(B_{i+1})$ is 0, it indicates that both bytes carry information, and the mutual information rate of the adjacent bytes is the ratio of the mutual information and joint entropy of the bytes, as shown in Equation (5). When neither the value of $H(B_i)$ nor $H(B_{i+1})$ is 0, the value of mutual information rate belongs to the interval $[0, 1]$.

Figure 4 is the distribution of the mutual information rate of adjacent bytes of the same Modbus TCP protocol packet. The mutual information rate of the 3rd and 4th adjacent bytes is 1, because the three bytes of byte offsets 2, 3, and 4 are the constant 0, and byte offsets 2 and 3 belong to the protocol identifier field. The byte offset 4 belongs to the length byte, but due to the small length of each packet in this file, most of the 1st byte in the length field is set to 0, which leads to the mutual information rate value of 1 for the 4th adjacent byte.

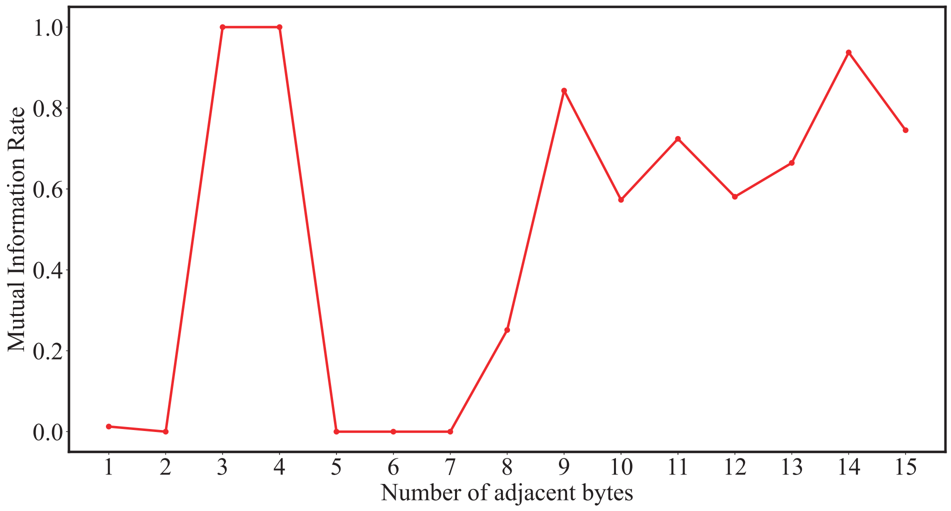


Figure 4. Mutual information rate distribution of adjacent bytes of Modbus TCP packets.

2.1.4. Proprietary Protocol Structure Analysis Algorithm

According to the mutual information rate of the adjacent bytes, the three judgment conditions that the adjacent bytes do not belong to the same field are contributed to complete protocol structure analysis. Three judgment conditions are as follows.

(1) As in Equation (6), if the mutual information rate between the ith and the $i+1$th byte is less than the mutual information rate between the $i-1$th and the ith byte and the mutual information rate between the $i+1$th and the $i+2$th byte, then the ith byte and the $i+1$th byte do not belong to the same field. The relationship score between the ith and the $i+1$th byte is set to 1 in this round; otherwise, the relationship score is 0.

$$\begin{aligned} MIR(B_i;B_{i+1}) &< MIR(B_{i+1};B_{i+2}) \\ MIR(B_i;B_{i+1}) &< MIR(B_{i-1};B_i) \end{aligned} \tag{6}$$

(2) As in Equation (7), if the mutual information rate between the ith byte and the $i+1$th byte is greater than the mutual information rate between the $i-1$th and the ith byte and the mutual information rate between the $i+1$th and the $i+2$th bytes, then the $i-1$th byte, the ith byte, the $i+1$th byte, and the $i+2$th byte are all judged to be different from the same field. Moreover, the relationship score between the $i-1$th and the ith byte and the relationship score between the $i+1$th and the $i+2$th byte are both set to 1 in this round; otherwise, the relationship score is 0.

$$\begin{aligned} MIR(B_i;B_{i+1}) &> MIR(B_{i-1};B_i) \\ MIR(B_i;B_{i+1}) &> MIR(B_{i+1};B_{i+2}) \end{aligned} \tag{7}$$

(3) As in Equation (8), if the mutual information rate between the ith byte and the $i+1$th byte is less than or equal to the threshold, the ith byte and the $i+1$th byte are judged to be different from the same field, and the relationship score between the ith and the $i+1$th byte is set to 1 in this round; otherwise, the relationship score is 0.

$$MIR(B_i;B_{i+1}) \le threshold \tag{8}$$

Because the three judgment conditions complement each other but overlap to some extent, the scores of the three rounds are accumulated to determine whether the adjacent bytes belong to the same field according to the final relationship score.

The pseudo-code of the protocol structure analysis algorithm is shown in Algorithm 1.

Algorithm 1 Segmentation Algorithm

```
Require: Information Entropy Array: $H(B_i), i = \{0, 1, 2, ...\}$
         Joint Entropy Array: $H(B_i, B_{i+1}), i = \{0, 1, 2, ...\}$
         Threshold: $T$
Ensure: Relationship between adjacent bytes Array: $R$
 1: Initialize Mutual Information Array: $I$
 2: $num \leftarrow len(H)$
 3: for $i = 0 \rightarrow num - 2$ do
 4:   if $H(B_i) == 0$ and $H(B_{i+1}) == 0$ then
 5:     $MIR(B_i; B_{i+1}) \leftarrow 1$
 6:   else if $H(B_i) \neq 0$ and $H(B_{i+1}) \neq 0$ then
 7:     $MIR(B_i; B_{i+1}) \leftarrow \frac{I(B_i;B_{i+1})}{H(B_i,B_{i+1})}$
 8:     $I(B_i; B_{i+1}) \leftarrow H(B_i) + H(B_{i+1}) - H(B_i, B_{i+1})$
 9:   else
10:     $MIR(B_i; B_{i+1}) \leftarrow 0$
11:   end if
12: end for
13: Initialize Score between adjacent bytes Array $S$ With 0
14: for $i = 0 \rightarrow num - 2$ do
15:   if $MIR(B_i; B_{i+1}) < MIR(B_{i+1}; B_{i+2})$ and $MIR(B_i; B_{i+1}) < MIR(B_{i-1}; B_i)$ then
16:     $S(B_i, B_{i+1}) \leftarrow S(B_i, B_{i+1}) + 1$
17:   end if
18:   if $MIR(B_i; B_{i+1}) > MIR(B_{i+1}; B_{i+2})$ and $MIR(B_i; B_{i+1}) > MIR(B_{i-1}; B_i)$ then
19:     $S(B_{i-1}, B_i) \leftarrow S(B_{i-1}, B_i) + 1$
20:     $S(B_{i+1}, B_{i+2}) \leftarrow S(B_{i+1}, B_{i+2}) + 1$
21:   end if
22:   if $MIR(B_i; B_{i+1}) \geq threshold$ then
23:     $S(B_i, B_{i+1}) \leftarrow S(B_i, B_{i+1}) + 1$
24:   end if
25: end for
26: Initialize Relationship between adjacent bytes Array $R$
27: for $i = 0 \rightarrow num - 2$ do
28:   if $S(B_i, B_{i+1}) \geq 1$ then
29:     $R(B_i, B_{i+1}) \leftarrow 1$
30:   else
31:     $R(B_i, B_{i+1}) \leftarrow 0$
32:   end if
33: end for
34: Return $R$
```

2.2. Implementation of Proprietary Protocol Structure Analysis Algorithm

The main function of the proprietary protocol structure analysis module is to identify the field boundaries of the captured industrial proprietary protocol data set, as shown in Figure 5, so as to complete the structure analysis of the proprietary protocol. The execution process of the module is shown in Figure 6, and the main steps are as follows:

(1) Obtain the industrial proprietary protocol data set. The communication data packets between the host computer and PLC are captured and screened, and the number of data packets is not less than 3000. Treat the packets in the generated PCAP file as proprietary protocol data sets.

(2) Collect necessary information about the packet set, including the packet number, N, the maximum byte length, L, and the byte offset set, B_i, of the packet.
(3) Calculate information entropy $H(B_i)$, joint information entropy $H(B_i, B_{i+1})$, and mutual information $I(B_i; B_{i+1})$. Count the occurrence frequency and the number of elements of each element in B_i and B_{i+1}, and then calculate mutual information $I(B_i; B_{i+1})$ as in Equation (9).

$$I(B_i; B_{i+1}) = H(B_i) + H(B_{i+1}) - H(B_i; B_{i+1}) \tag{9}$$

(4) Calculate the mutual information rate, $MIR(B_i; B_{i+1})$.
(5) The relationship between adjacent bytes in the data set of the industrial proprietary protocol is judged by three rounds, and than the array, J, is obtained.
(6) Count and store the final byte-relational array, R. Initialize the byte-relational array, R. If element J_i in array J is greater than threshold 1, then the ith and $i+1$th bytes do not belong to the same field, let $r_i = 1$; otherwise, the ith and $i+1$th bytes belong to the same field, let $r_i = 0$. Add r_i to array R.
(7) Output the byte relation to array R.

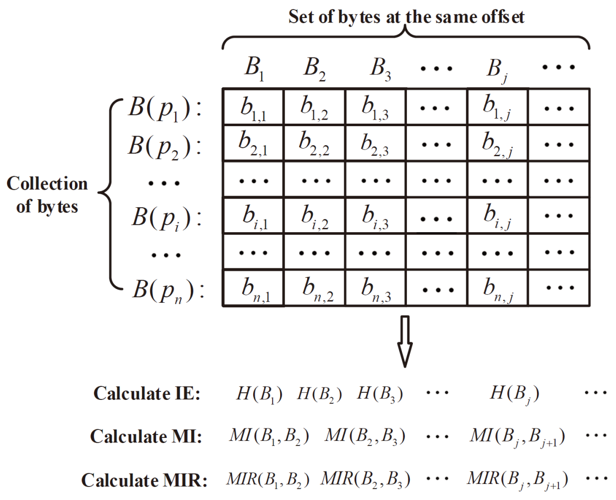


Figure 5. Proprietary protocol structure analysis method.

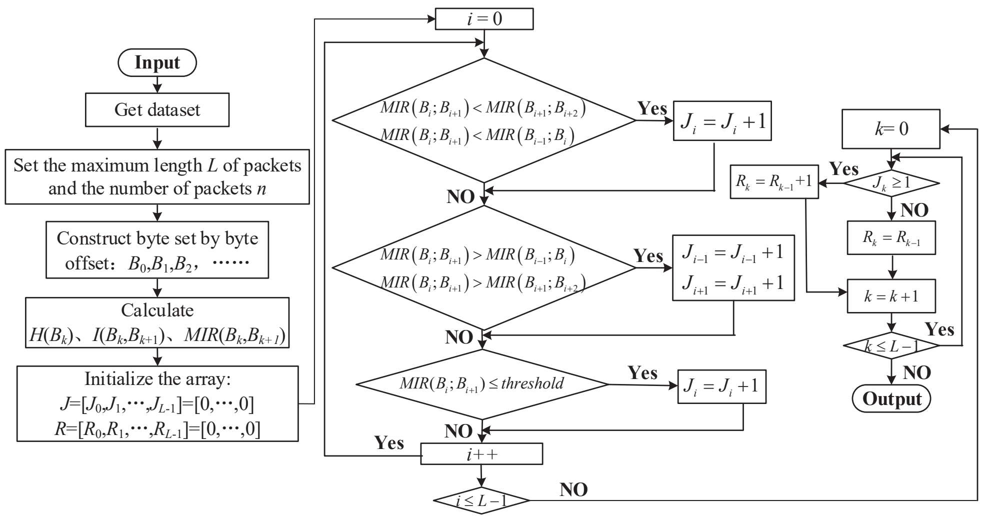

Input
Get dataset
Set the maximum length L of packets and the number of packets n
Construct byte set by byte offset: B0,B1,B2, ……
Calculate H(Bk)、I(Bk,Bk+1)、MIR(Bk,Bk+1)
Initialize the array: J=[J0,J1,…,JL-1]=[0,…,0] R=[R0,R1,…,RL-1]=[0,…,0]
i = 0
Yes
NO
threshold
i++
i ≤ L−1
k=0
k = k+1
k ≤ L−1
Output

Figure 6. Proprietary protocol structure analysis implementation flow.

3. Proprietary Protocol Fuzzing Test Cases Generation Algorithm

Compared with conventional software vulnerability mining, the difficulty of fuzzing-based industrial proprietary protocol vulnerability discovery lies in the fact that the proprietary protocol is a state machine and the correctness of data packets has strong constraints.

If an abnormal packet is constructed that does not conform to the format, there is no chance of triggering the vulnerability. Therefore, fuzzing test tools should be intelligent, and genetic algorithms [19] are well suited for this task. This section uses an improved genetic algorithm to generate fuzzing test cases for proprietary protocols.

3.1. Fuzzing Test Cases Generation Algorithm of Genetic Algorithm-Based Proprietary Protocol

The test case generation process includes individual coding, population initialization, selection, crossover, mutation operation, new population generation, decoding, byte variation, and population iteration. The specific steps are as follows:

(1) Gray coding is used to encode individuals. From left to right, the first digit of the gray code gene sequence is the first digit of the binary code gene sequence. Then, from the first digit of the binary code, the XOR operation is performed on each digit adjacent to its right, and the calculation result is taken as the gray code value at the corresponding position.

(2) Similarity calculation. The local sequence alignment Smith–Waterman algorithm [20] is used to evaluate the similarity between two packets, p_g and p_c, where p_g is the individual generated by the genetic algorithm and p_c is the packet set of the upper computer and programmable logic controller (PLC); P_c is a collection of communication packets between the engineering station and the PLC. p_c represents one of the packets in this collection, specifically, $p_c \in P_c$, as shown in Equation (10), where l_1 and l_2 represent the byte length of two packets, respectively.

$$\begin{array}{l} p_g = b_1 b_2 \cdots b_{l_1} \\ p_c = b'_1 b'_2 \cdots b'_{l_2} \end{array} \tag{10}$$

First, score matrices are established for p_g and p_c, and the first row and first column of the matrix are initialized. The initialization method is shown in Equation (11), and the size of the matrix is $(l_1+1) \times (l_2+1)$.

$$\begin{cases} G_{i,0} = 0, 0 \le i \le l_1 \\ G_{0,j} = 0, 0 \le j \le l_2 \end{cases} \tag{11}$$

Then, the rest of the score matrix, G, is filled; the element filling algorithm in matrix G is shown in Equation (12).

$$G_{i,j} = \max \begin{cases} G_{i-1,j-1} + s(b_i, b'_J), \\ G_{i-k,j} - W_k, \\ G_{i,j-l} - W_l, \\ 0 \end{cases} \quad (1 \le i \le l_1, 1 \le j \le l_2) \tag{12}$$

where $s(b_i, b'_J)$ represents the similarity score between b_i and b'_J, and the calculation process is shown in Equation (13). W_k represents the blank penalty point with length k, and the calculation process is shown in Equation (14). 0 indicates that two bytes are not similar.

$$s\left(b_i, b'_j\right) = \begin{cases} +3, b_i = b'_j \\ -3, b_i \ne b'_j \end{cases} \tag{13}$$

$$W_k = kW_1, \quad W_1 = 2 \tag{14}$$

After matrix G is calculated, the maximum value in the matrix is taken as the similarity measure of the two packets, as shown in Equation (15).

$$sim(p_g, p_c) = \max\{G_{i,j}\}, (0 \le i \le l_1, 0 \le j \le l_2) \tag{15}$$

Then, the similarity of individual p_g and all other packets in P_c is calculated, and all the similarity values are summed then divided by the number n of P_c, and the average similarity of individual p_g and P_c is calculated, as shown in Equation (16).

$$\overline{sim(p_g, P_c)} = \frac{\sum_{p_{c_i} \in P} sim(p_g, p_{c_i})}{n} \tag{16}$$

(3) The fitness function is used to distinguish between good and bad individuals in a population. The fitness function selected is shown in Equation (17), where r_{imax} and r_{imin} are individual maximum and minimum values, λ_1, λ_2, and λ_3 are weight factors.

$$\begin{aligned} f_k(x) &= \frac{r_{i\max} - r_i}{r_{i\max} - r_{i\min}}, r_{i\min} \leq r_i \leq r_{i\max}, i = 1,2,3 \\ F(x) &= \lambda_1 \cdot f_1(x) + \lambda_2 \cdot f_2(x) + \lambda_3 \cdot f_3(x) \\ s.t. \quad & \lambda_1 + \lambda_2 + \lambda_3 = 1 \end{aligned} \tag{17}$$

(4) Byte variation. In order to improve the efficiency of the fuzzing test of proprietary protocols, byte variation factors for numerical boundary vulnerability and formatted string vulnerability are proposed by means of vulnerability information analysis. In order to improve the effectiveness of test cases, the genetic algorithm produces test cases and carries out another round of byte variation.
A roulette wheel algorithm is used to pick byte variants. First, equal selection probability for each variation factor is set, for example, the selection probability of the ith variation factor is Pro_i. Furthermore, the cumulative probability, Q_i, of each variation factor is calculated according to Equation (18). The random value r is then generated in the interval $[0,1]$. If $r < Q_i$, individual 1 is selected. Then, a random number, j, is generated within the interval $[0,1]$. According to the information entropy, whether the jth byte in the packet is a fixed byte is determined. If it is not a fixed byte, it is replaced by the kth variation factor. If the byte is fixed, a pseudo-random number is generated until the replacement is complete.

$$Q_i = \sum_{j=1}^{i} Pro_j \tag{18}$$

After continuous population replacement, the final algorithm will converge to the global optimal solution. A fuzzing test case generation flow chart of the proprietary protocol is shown in Figure 7.

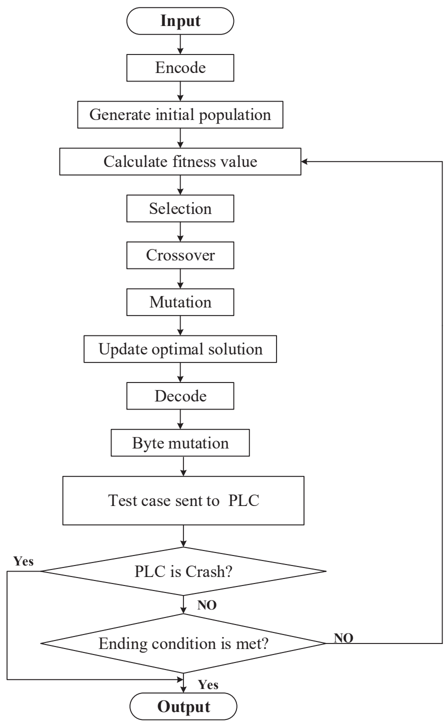


Figure 7. Proprietary protocol fuzzing test case generation module.

3.2. Implementation of Fuzzing Test Case Generation Algorithm for Proprietary Protocol

The fuzzing test case generation module of the proprietary protocol generates cases according to the analyzed proprietary protocol structure. Then, test cases are entered into the Peach fuzzer format analysis module, and the state control module guides the interactive publishing module to input the fuzzing test case into the PLC; at the same time, the agent monitoring module detects whether the PLC is abnormal. If the PLC is abnormal, it is recorded through the log recording module and the test will be terminated; if the PLC appears normal, then the work process continues to cycle.

The steps of the proprietary protocol fuzzing test case generation module are as follows:

(1) Gray coding method is used to encode individuals to form the primary population.
(2) According to Equation (8), the individual fitness value is calculated.
(3) The parent of the next crossover is selected by the roulette selection operator.
(4) For the selected paternal individuals, the crossover probability, P_c, is calculated to exchange the chromosomes of the two crossover sites to produce new individual gene sequences.
(5) The mutation probability, P_m, is calculated and mutation operations on individual gene sequence are performed.
(6) After decoding individual gene sequences, the byte mutation algorithm is used to mutate the decoded packet.
(7) The generated test case is sent to the device through the Peach fuzzer.
(8) The agent monitoring module of the Peach fuzzer is used to regularly monitor whether the device is abnormal. If there is an exception, the fuzzing test is terminated; otherwise, Step (9) is initiated.
(9) If the algorithm termination condition is satisfied, the fuzzing test is complete; otherwise, the process returns to Step (2).

4. Experimental Testing and Verification Result

This section will test and verify the proposed fuzzing test method for industrial protocols. The experiment is divided into two parts: proprietary protocol structure analysis experiment and proprietary protocol fuzzing test experiment. In the proprietary protocol structure analysis experiment, Modbus TCP and a proprietary protocol with unknown structure are analyzed, respectively. The effectiveness of the proprietary protocol structure analysis method proposed in this paper is verified by the structure analysis results of Modbus TCP packets. Furthermore, the structure of a proprietary protocol with unknown structure is analyzed and the results are used in the fuzzing test experiment of the proprietary protocol. In the experiment of the fuzzing test of the proprietary protocol, the experimental objects are two PLCS, which use the above proprietary protocol to communicate. By analyzing the experimental results and comparing with similar tools, the effectiveness of the proposed method is confirmed.

4.1. Experimental Environment

The experimental network topology is shown in Figure 8. The upper computer is connected to the Ethernet switch through the network cable, and the Ethernet switch is connected to the industrial control device through the network cable.

The upper computer environment is shown in Table 1.

Table 1. Upper computer environment.

Environment	Configuration
Operating System Version	Windows 10 Home
CPU Model	Intel(R) Core(TM) i5-7200U CPU @ 2.50 GHz
Memory	16.00 GB
System Compilation Environment	NET 4.5.1, Python

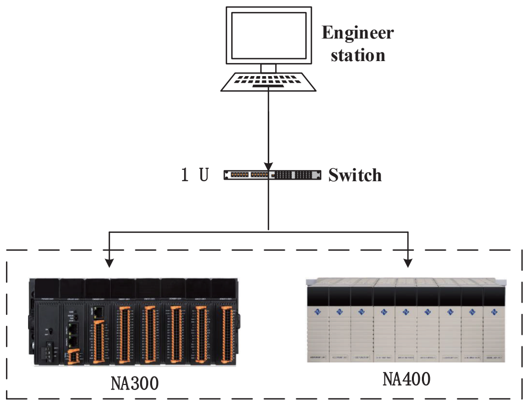


Figure 8. Experimental network topology.

The proprietary protocol fuzzing test object is represented by Nanda Auto PLC NA300 and NA400; these classes of PLC mainly communicate through the proprietary protocol designed by the manufacturer.

4.2. *Proprietary Protocol Structure Analysis Validation*

The structure analysis experiment of the proprietary protocol can be divided into two parts. The first part is the validity verification experiment of the proprietary protocol structure analysis method; in this part, the proposed method is used to parse the Modbus TCP protocol. Because the structure of the Modbus TCP protocol is known, the validity of the proposed method can be verified according to the analysis result. The second part is the structure analysis experiment of the proprietary protocol used by NA400 and NA300. According to the analysis result, the fuzzing test experiment of the proprietary protocol is carried out.

4.2.1. Structure Analysis of Modbus TCP Protocol

According to the information entropy of each byte and the mutual information of the adjacent bytes shown in Figures 1 and 3, the mutual information rate between adjacent bytes is further calculated, as shown in Figure 4. Based on three judgment conditions, the structure analysis of the Modbus TCP protocol is finally completed, and the analytical results are shown in Table 2.

Table 2. Results of the Modbus TCP protocol structure analysis.

Field offset	0	1	2	3	4	5	6	7	8	9	10	11	···
ine Field number	1		2			3	4	5	6		7		···

Comparing the structure analysis result with the structure of the Modbus TCP protocol, field 1 is the transaction identifier; field 2 is a protocol identifier; field 3 is the length of the field; field 4 is the unit identifier; and fields 5, 6, and 7 are the function code field, data address field, and data value field in the protocol data unit (PDU), respectively. Therefore, only fields 2 and 3 differ from the correct result; all other fields agree with the correct result.

Through the analysis of the information entropy, mutual information, and mutual information rate, it is found that the information entropy of the 2nd, 3rd, and 4th bytes are all 0, but the information entropy of the 5th byte is not 0, so the mutual information rate of adjacent bytes is 1, the mutual information rate between the 2nd and 3rd bytes and the mutual information rate between the 3rd and 4th bytes are 1, and the 4th and 5th bytes of mutual information rate between them is 0. Therefore, according to the three judgment conditions, the 2nd, 3rd, and 4th bytes are finally judged to belong to the same field, while the 5th byte belongs to a field alone. It can be seen that the reason for the analysis error is that the length of the data packet in the sample set used in this paper is short, and most values of the 4th byte are 0, so the information entropy of the 4th byte is calculated to be 0, resulting in the final analysis result being inconsistent with the correct result.

Due to the short packet length in the sample set, the analysis result is inconsistent with the correct one, but the parsing of the other fields is correct. Accordingly, the validity of the proprietary protocol structure analysis method proposed in this paper can be verified through the structure analysis experiment of the Modbus TCP protocol.

4.2.2. Structure Analysis of Unknown Proprietary Protocols

The NA400 proprietary protocol packet consists of 5500 messages captured from the communication between the upper computer with NA400 PLC. The information entropy of each byte of the NA400 PLC messages is shown in Figure 9. The information entropy of the 0th, 1st, 2nd, 3rd, and 6th bytes are all 0, indicating that the values of these four bytes are fixed constants. The mutual information between adjacent bytes of the NA400 PLC packet is shown in Figure 10. Figure 11 shows the distribution diagram of the mutual information rate of adjacent bytes of the NA400 PLC packet. According to the mutual information rate, the proprietary protocol analysis algorithm proposed in Section 2 is used to inference the structure of the proprietary protocol used by NA400. The analysis result of the protocol structure is demonstrated in Table 3.

Table 3. Proprietary protocol structure of NA400.

Field number	1	2	3	4	5	6	· · ·
ine Byte offset	0˜3	4˜5	6	7˜8	9˜10	11˜12	· · ·
ine Field length	4	2	1	2	2	2	· · ·
ine Field description	length	length field			data segment		

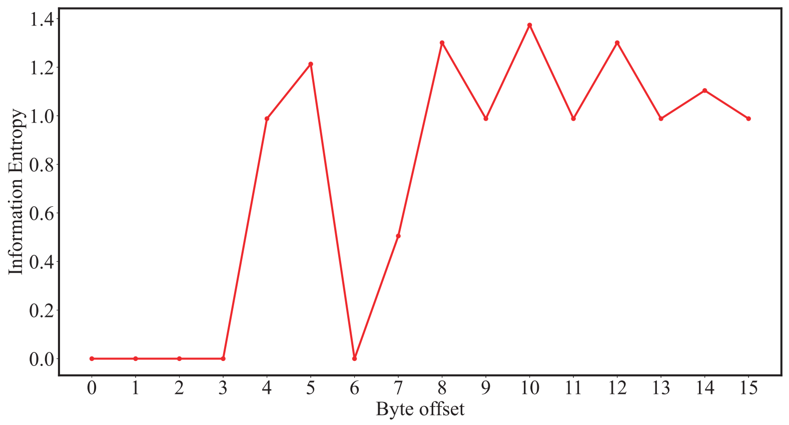


Figure 9. Information entropy of each byte of the NA400 packet.

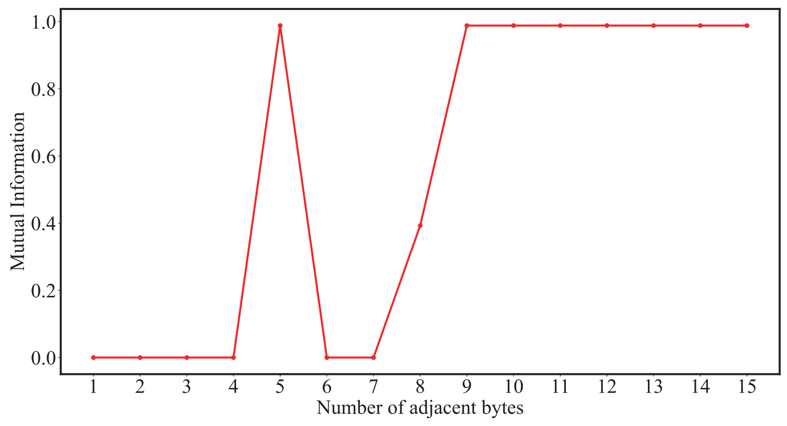


Figure 10. Mutual information of adjacent bytes of the NA400 packet.

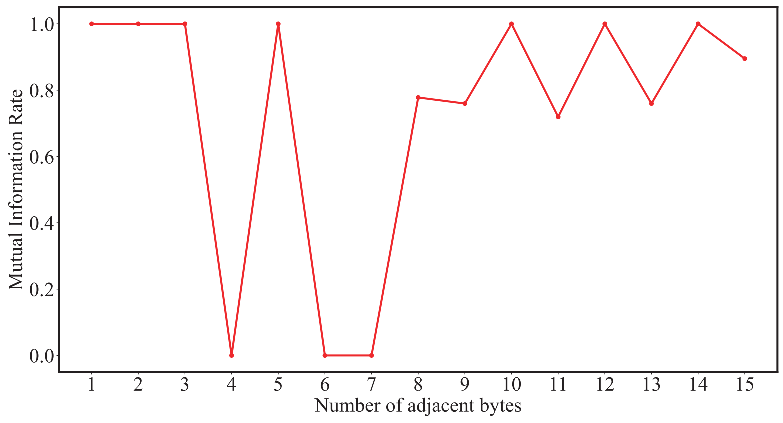
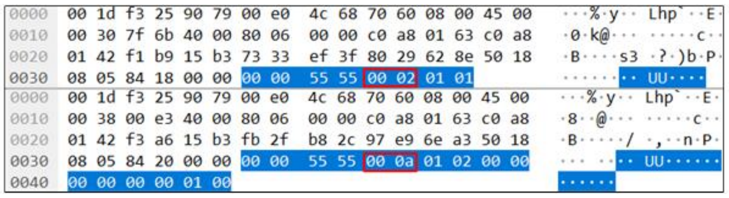


Figure 11. Mutual information rate of adjacent bytes of the NA400 packet.

In Table 3, the first seven fields of the NA400 protocol are listed, in which field 1 consists of the 0th to the 3rd bytes, and the length is 4 bytes. As shown in Figure 12, the value in the red box of the first message is "00 02", followed by two bytes of data, while the value in the red box of the second message is "00 0a". There are 10 bytes of data after the field; field 3 only consists of the 6th byte, and its information entropy is 0, then field 3 is a fixed field, and its value is "01". Fields 4, 5, and 6, etc., are mutable fields of length 2 bytes. In addition, the NA300 protocol structure is similar to the NA400 protocol structure.



```
0000  00 1d f3 25 90 79 00 e0  4c 68 70 60 08 00 45 00   ···%·y·· Lhp`··E·
0010  00 30 7f 6b 40 00 80 06  00 00 c0 a8 01 63 c0 a8   ·0·k@··· ·····c··
0020  01 42 f1 b9 15 b3 73 33  ef 3f 80 29 62 8e 50 18   ·B····s3 ·?·)b·P·
0030  08 05 84 18 00 00 00 00  55 55 00 02 01 01         ········ ·· UU····
0000  00 1d f3 25 90 79 00 e0  4c 68 70 60 08 00 45 00   ···%·y·· Lhp`··E·
0010  00 38 00 e3 40 00 80 06  00 00 c0 a8 01 63 c0 a8   ·8··@··· ·····c··
0020  01 42 f3 a6 15 b3 fb 2f  b8 2c 97 e9 6e a3 50 18   ·B·····/ ·,··n·P·
0030  08 05 84 20 00 00 00 00  55 55 00 0a 01 02 00 00   ··· ···· ·· UU······
0040  00 00 00 00 01 00                                   ······
```

Figure 12. NA400 proprietary protocol length field.

4.3. Proprietary Protocol Fuzzing Test Validation

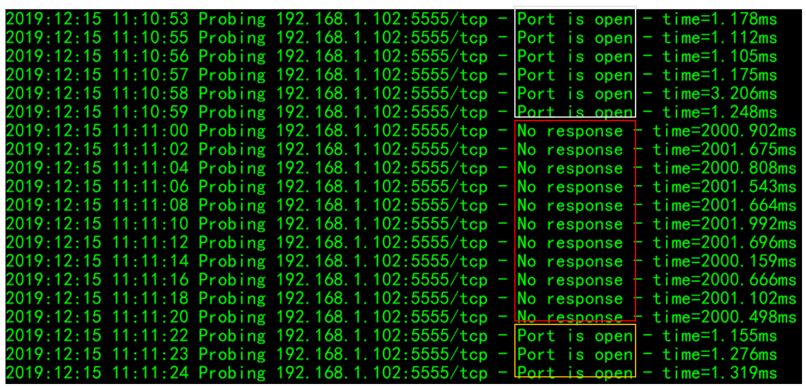
According to the analysis result of the proprietary protocol structure, the fuzzing test experiment of NA400 and NA300 is carried out. The experimental results are shown in Table 4. We discovered two vulnerabilities, one of which is 0-Day vulnerability. The following will analyze the principle of vulnerability.

Table 4. Proprietary protocol fuzzing test results.

Number	Vendor	Equipment Model	Trigger Vulnerability	Type	Serial Number
1	NA	NA400	Yes	DoS	CNVD-2020-04096
2	NA	NA300	Yes	DoS	CNVD-2018-26205

In experiment 1, the Nanda Auto PLC NA400 PLC was tested, which is a medium-to-large PLC of Nanda Automation Technology Beijing Co., Ltd. (Beijing, China). During the fuzzing experiment on the NA400 PLC, a 0-Day vulnerability was discovered with vulnerability CNVD number CNVD-2020-04096. The attacker continuously sent specific message to the NA400 PLC, causing the device to provide denial-of-service (DoS).

During experiment 1, it was found that when the NA400 PLC received a specific message, the NA400 PLC will provide denial-of-service (DoS) and the communication with the host computer will be interrupted. After approximately 2 seconds, the NA400 PLC resumed normal work. In order to demonstrate this process more intuitively, when reproducing the vulnerability, this paper uses the software TCPing to monitor the open status of the 5555 port of the NA400 PLC, as shown in Figure 13. According to the monitoring process and debugging results of the software TCPing, it is inferred that after the NA400 PLC received the message containing the attack payload, the NA400 PLC system crashed, and then the system began to restart, and the 5555 port was reopened after the PLC restarted; after 11:11:22, the 5555 port of NA400 PLC continued to be open. Therefore, if the attacker detected that the NA400 PLC with this firmware version was connected to the Internet, and the attacker continued to send specific message to the 5555 port of the NA400 PLC, the NA400 PLC will always be in a state of denial-of-service (DoS).

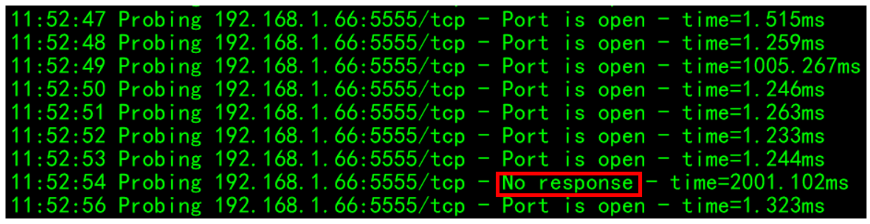


```
2019:12:15 11:10:53 Probing 192.168.1.102:5555/tcp - Port is open - time=1.178ms
2019:12:15 11:10:55 Probing 192.168.1.102:5555/tcp - Port is open - time=1.112ms
2019:12:15 11:10:56 Probing 192.168.1.102:5555/tcp - Port is open - time=1.105ms
2019:12:15 11:10:57 Probing 192.168.1.102:5555/tcp - Port is open - time=1.175ms
2019:12:15 11:10:58 Probing 192.168.1.102:5555/tcp - Port is open - time=3.206ms
2019:12:15 11:10:59 Probing 192.168.1.102:5555/tcp - Port is open - time=1.248ms
2019:12:15 11:11:00 Probing 192.168.1.102:5555/tcp - No response - time=2000.902ms
2019:12:15 11:11:02 Probing 192.168.1.102:5555/tcp - No response - time=2001.675ms
2019:12:15 11:11:04 Probing 192.168.1.102:5555/tcp - No response - time=2000.808ms
2019:12:15 11:11:06 Probing 192.168.1.102:5555/tcp - No response - time=2001.543ms
2019:12:15 11:11:08 Probing 192.168.1.102:5555/tcp - No response - time=2001.664ms
2019:12:15 11:11:10 Probing 192.168.1.102:5555/tcp - No response - time=2001.992ms
2019:12:15 11:11:12 Probing 192.168.1.102:5555/tcp - No response - time=2001.696ms
2019:12:15 11:11:14 Probing 192.168.1.102:5555/tcp - No response - time=2000.159ms
2019:12:15 11:11:16 Probing 192.168.1.102:5555/tcp - No response - time=2000.666ms
2019:12:15 11:11:18 Probing 192.168.1.102:5555/tcp - No response - time=2001.102ms
2019:12:15 11:11:20 Probing 192.168.1.102:5555/tcp - No response - time=2000.498ms
2019:12:15 11:11:22 Probing 192.168.1.102:5555/tcp - Port is open - time=1.155ms
2019:12:15 11:11:23 Probing 192.168.1.102:5555/tcp - Port is open - time=1.276ms
2019:12:15 11:11:24 Probing 192.168.1.102:5555/tcp - Port is open - time=1.319ms
```

Figure 13. TCPing monitoring of NA400.

The industrial control equipment tested in experiment 2 is the Nanda Auto PLC NA300 PLC, which is a medium-sized PLC of Nanda Automation Technology Beijing Co., Ltd. In the fuzzing test of the NA300 PLC, a known CNVD vulnerability was reproduced with vulnerability number CNVD-2018-26205. This vulnerability was caused by a function not being handled properly. The attacker was able to crash the NA300 by constructing a super-long packet that overwrites the return address of a function.

In experiment 2, it was found that when port 5555 of the NA300 PLC received specified message, there would be a brief DoS and communication interruption. In order to demonstrate this process more clearly, software TCPing is used to monitor the fuzzing test process when the vulnerability is reproduced. As shown in Figure 14, port 5555 of the NA300 PLC was still open until 11:52:53; at 11:52:52, the fuzzing test script was executed and the packet containing the attack payload was sent to NA300 PLC. There was a brief DoS and communication outage at 11:52:54. Compared with the NA400 PLC, the NA300 PLC restarts more quickly. It was found that the NA300 PLC system did not crash after receiving the message containing the attack payload. In combination with the description of the vulnerability in the CNVD, it is speculated that the reason for triggering this vulnerability is that the NA300 PLC does not check the length of the data packets, resulting in buffer overflow, overwriting the return address of the function. One process crashed, but the entire system did not crash, so there was a DoS, but normal work and communication quickly resumed.



```
11:52:47 Probing 192.168.1.66:5555/tcp - Port is open - time=1.515ms
11:52:48 Probing 192.168.1.66:5555/tcp - Port is open - time=1.259ms
11:52:49 Probing 192.168.1.66:5555/tcp - Port is open - time=1005.267ms
11:52:50 Probing 192.168.1.66:5555/tcp - Port is open - time=1.246ms
11:52:51 Probing 192.168.1.66:5555/tcp - Port is open - time=1.263ms
11:52:52 Probing 192.168.1.66:5555/tcp - Port is open - time=1.233ms
11:52:53 Probing 192.168.1.66:5555/tcp - Port is open - time=1.244ms
11:52:54 Probing 192.168.1.66:5555/tcp - No response - time=2001.102ms
11:52:56 Probing 192.168.1.66:5555/tcp - Port is open - time=1.323ms
```

Figure 14. TCPing monitoring of NA300.

5. Conclusions and Future Work

This paper proposed an industrial proprietary protocol fuzzing system. First, according to the proprietary protocol data set, this paper calculated the information entropy of each byte in the data set, the mutual information between adjacent bytes, and the mutual information rate. On the basis of three judgment conditions that adjacent bytes do not belong to the same field, the proprietary protocol structure was analyzed based on the judgment result. Furthermore, this paper used the genetic algorithm to generate the fuzzing test case of the proprietary protocol and calculated the similarity between the individual and the normal message by the Smith–Waterman algorithm, then calculated the fitness of the individual based on the similarity value. Moreover, in order to improve fuzzing efficiency, one byte mutation was performed on the generated test cases; this paper designed the industrial proprietary protocol fuzzing system to test the PLC equipment of many domestic and foreign manufacturers, and discovered three original vulnerabilities, one of which was the 0-Day vulnerability. The experimental results of comparative analysis demonstrated that the fuzzing system can exploit the vulnerabilities of industrial equipment more effectively.

When analyzing the proprietary protocol structure, there are certain requirements for the proprietary protocol data set, and the diversity of the value of each byte in the data set will greatly affect the parsing result of the proprietary protocol structure parsing algorithm. Future work will reduce the dependence of the proprietary protocol structure parsing algorithm. In addition, only the structure of the protocol header can be reversely analyzed, and the structure of the data segment of the protocol cannot be accurately reversed. The future work will analyze the field boundary and field function semantics of the protocol data segment part based on the specific business production context and infer protocol interaction rules.

Author Contributions: X.C. designed the main methodology and the model; Y.G. and M.W. developed the theoretical framework. G.Z. provided valuable ideas. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Nature Science Foundation of China (U1911401, 61972345), and the Fundamental Research Funds for the Central Universities (Zhejiang University NGICS Platform).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bhamare, D.; Zolanvari, M.; Erbad, A.; Jain, R.; Khan, K.; Meskin, N. Cybersecurity for industrial control systems: A survey. *Comput. Secur.* **2020**, *89*, 101677. [CrossRef]
2. Eceiza, M.; Flores, J.L.; Iturbe, M. Fuzzing the internet of things: A review on the techniques and challenges for efficient vulnerability discovery in embedded systems. *IEEE Internet Things J.* **2021**, *8*, 10390–10411. [CrossRef]
3. Aafer, Y.; You, W.; Sun, Y.; Shi, Y.; Zhang, X.; Yin, H. Android SmartTVs Vulnerability Discovery via Log-Guided Fuzzing. In Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), Vancouver, BC, Canada, 11–13 August 2021; pp. 2759–2776.
4. Zhao, J.; Lu, Y.; Zhu, K.; Chen, Z.; Huang, H. Cefuzz: An Directed Fuzzing Framework for PHP RCE Vulnerability. *Electronics* **2022**, *11*, 758. [CrossRef]
5. Nadeem, S.; Tumreen, M.; Ishtiaq, B.; Abbas, N. Three-dimensional second-grade nanofluid flow with MHD effects through a slandering stretching sheet: A numerical solution. *Waves Random Complex Media* **2022**, 1–19. [CrossRef]
6. Lin, P.; Hong, Z.; Li, Y.; Wu, L. A priority based path searching method for improving hybrid fuzzing. *Comput. Secur.* **2021**, *105*, 102242. [CrossRef]
7. Sun, Y.; Lv, S.; You, J.; Sun, Y.; Chen, X.; Zheng, Y.; Sun, L. IPSpex: Enabling Efficient Fuzzing via Specification Extraction on ICS Protocol. In Proceedings of the International Conference on Applied Cryptography and Network Security, Rome, Italy, 20–23 June 2022; Springer: Cham, Switzerlands, 2022; pp. 356–375.
8. Lin, P.; Tien, C.; Huang, T.; Tien, C. ICPFuzzer: Proprietary communication protocol fuzzing by using machine learning and feedback strategies. *Cybersecurity* **2021**, *4*, 28. [CrossRef]
9. Beaman, C.; Redbourne, M.; Mummery, J.D.; Hakak, S. Fuzzing Vulnerability Discovery Techniques: Survey, Challenges and Future Directions. *Comput. Secur.* **2022**, *120*, 102813. [CrossRef]
10. Shu, Z.; Yan, G. IoTInfer: Automated blackbox fuzz testing of IoT network protocols guided by finite state machine inference. *IEEE Internet Things J.* **2022**, *9*, 22737–22751. [CrossRef]
11. Nilizadeh, S.; Noller, Y.; Pasareanu, C.S. Diffuzz: Differential Fuzzing for Side-Channel Analysis. In Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), Montreal, QC, Canada, 25–31 May 2019; pp. 176–187.

12. Leita, C.; Mermoud, K.; Dacier, M. ScriptGen: An Automated Script Generation Tool for Honeyd. In Proceedings of the Computer Security Applications Conference, Tucson, AZ, USA, 5–9 December 2016.
13. Wang, X.; Lv, K.; Li, B. IPART: An Automatic Protocol Reverse Engineering Tool Based on Global Voting Expert for Industrial Protocols. *Int. J. Parallel Emergent Distrib. Syst.* **2019**, *35*, 376–395. [CrossRef]
14. Kumari, S.; Singh, A. Effect of correlations on routing and modeling of Time Varying Communication Networks. *arXiv* **2018**, arXiv:1811.06274.
15. Pacheco, M.L.; Hippel, M.V.; Weintraub, B.; Goldwasser, D.; Nita-Rotaru, C. Automated Attack Synthesis by Extracting Finite State Machines from Protocol Specification Documents. In Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 22–26 May 2022.
16. Lai, Y.; Gao, H.; Liu, J. Vulnerability Mining Method for the Modbus TCP Using an Anti-Sample Fuzzer. *Sensors* **2020**, *20*, 2040. [CrossRef] [PubMed]
17. Shen, Y.; Sun, H.; Jiang, Y.; Shi, H.; Yang, Y.; Chang, W. Rtkaller: State-aware Task Generation for RTOS Fuzzing. *ACM Trans. Embed. Comput. Syst.* **2021**, *20*, 83. [CrossRef]
18. Cincotta, P.M.; Giordano, C.M.; Silva, R.A.; Beaugé, C. The Shannon entropy: an efficient indicator of dynamical stability. *Phys. Nonlinear Phenom.* **2021**, *417*, 132816. [CrossRef]
19. He, Z.; Chen, G.; Hao, T.; Liu, X.; Teng, C. An optimal filter length selection method for MED based on autocorrelation energy and genetic algorithms. *ISA Trans.* **2021**, *109*, 269–287. [CrossRef] [PubMed]
20. Petti, S.; Bhattacharya, N.; Rao, R.; Dauparas, J.; Thomas, N.; Zhou, J.; Rush, A.M.; Koo, P.K.; Ovchinnikov, S. End-to-end learning of multiple sequence alignments with differentiable Smith-Waterman. *Bioinformatics* **2021**, *39*, btac724. [CrossRef] [PubMed]