Test Case Generation Method based on Generative Adversarial Network

Jiaojiao Nie*, Xianglin Fan and Yushuang Wu

Automated institute, Chongqing University of Posts and Telecommunications, Chongqing, 400065, China

*Corresponding author’s e-mail: s180331041@stu.cqupt.edu.cn

Abstract. Traditional fuzzing tools generate low diversity of test cases and low vulnerability detection efficiency. This paper uses a test case generation model based on a generative confrontation network. The model uses LSTM as the generation network to generate data. The model uses fully connected network to construct a discriminant network for classification, and the test cases with the same protocol format are automatically generated after training. Finally the model uses the generated test cases to attack the protocol system to detect vulnerabilities. This method is compared with the traditional tools AFL, Peach and Sulley on the Modbus protocol, and the validity of the method is verified.

1. Introduction
There may be some potential vulnerability in the implementation of the protocol test system[1]. If these vulnerabilities are not detected before the protocol test, it may lead to serious consequences such as system collapse. Genetic Algorithm (GA), Simulated Annealing (SA), Particle Swarm Optimization (PSO) and other algorithms are widely used in the field of test case generation due to their unique advantages, but each has some drawbacks[2]. The convergence speed of the Simulated Annealing (SA) is slow, so the efficiency of the algorithm is low. Particle Swarm Optimization (PSO) has poor local optimization ability, easy to fall into local extremum, and prone to premature convergence or stagnation[3].

In view of the above problems, the contributions of this paper include:

- In this paper, by analyzing the characteristics of protocol data, we construct a generation adversarial network model to automatically learn protocol specification, and generate test cases of protocol with correct format by generating adversarial network.
- The simulation system of Modbus protocol is built, and the fuzzy test of the system is carried out by using the test case generated by the generation network model proposed in this paper.
- Compared with traditional tools such as AFL, Peach and Sulley, the effectiveness of the algorithm is verified in terms of test case acceptance rate and vulnerability detection efficiency.

The rest of this paper is arranged as follows: the second part introduces the background knowledge of generative adversarial network, In the third part, the test case generation model based on generative adversarial network is designed and the training results are obtained. The fourth section introduces the experimental results. We summarize this paper in Section 5.

2. Related work
Generative Adversarial Net (GAN) [4] is mainly composed of two modules, one is generating module and the other is discriminating module. The generation module takes random noise as input, and the
model outputs the desired picture or text after continuous training. The discrimination module takes the original data and the generated data as the input, classifies the input data, and distinguishes the generated data and the real data. The generation module and the discrimination module adjust the parameters of the network model by continuously training their own generation ability and discrimination ability, so that the model is finally in a state of balance.

By analyzing the characteristics of the protocol data, because the protocol data is presented in the form of text, and the data length of the protocol data is not consistent, a generation network model based on LSTM[5] is designed. Generate test cases with the same protocol data structure through randomly generated noise. The discriminant network model is composed of fully connected networks, and the discriminant network distinguishes between real test cases and generated test cases. A large number of test cases that can be accepted by the system can be generated through the training of the model, and the vulnerabilities in the system can be triggered by sending test cases. The test cases that trigger vulnerabilities can be recorded and the model can be retrained to generate more data that trigger vulnerabilities. Test case generation method based on generative adversarial learning can avoid the acquisition of prior knowledge of protocol and improve the efficiency of protocol testing.

### 3. Test case generation model design based on generation adversarial network

The overall architecture of test case generation model is composed of LSTM (long-term memory network) generation network model and discrimination network model based on full connection network. Figure 1 shows the model architecture. The generation network part is a generation network model based on LSTM, which generates test cases through random noise, and the discriminant network module is used to calculate divergence $D_{CVaD}$ to optimize both.

![Figure 1. Test case generation framework](image)

#### 3.1. Generating network model based on LSTM

As shown in Figure 1, the input of the generated network is a random noise $z$ satisfying the normal distribution, $h(t)$ representing the hidden output at the current time $t$. The final generated data of the generated network is a sequence. By constructing a word embedded matrix $W \in \mathbb{R}^{d \times v}$, where $d$ represents the dimension of word vector and $v$ represents the size of vocabulary, the final output data of each unit is $y_i$. Then, the hidden state of the LSTM at the current moment is as shown in formula (1):

$$h(t) = \sigma(w_i[h_{i-1}, z, y_{i-1}]+b_o) \times \text{Tanh}(c_i)$$

(1)
Where $y_{t-1}$ represents the real output at the last moment, $w$ and $b$ is the parameter of the output gate $o_t$. In this paper, the default Tanh function is used as the activation function. Then, According to formula 1, there is the output of LSTM unit at the current moment:

$$c_t = f_i \times c_{t-1} + i_t \times \tilde{c}_t$$

(2)

$$h_t' = w' \cdot c_t + b'$$

(3)

$$y_t = \text{softmax}(h_t')$$

(4)

Where $w'$ is a parameter matrix and $b'$ is a bias term. In this article, the initial bias is set as a zero vector. Their function is to update the output at the current moment $c_t$. The dimension of $y_t$ is the length $V$ of the vocabulary. The final generated data $y_t$ represents a distribution in the vocabulary.

3.2. A discriminant network model based on fully connected networks

The discriminant network is mainly composed of a fully connected network $F(\theta)$ and a optimization calculation module, in which the optimization calculation module is further divided into a cost matrix calculation module and a transfer matrix calculation module based on the IPOT algorithm.

1. Feature extraction of data

Although the protocol data to distinguish network input is sequence data and RNN is better at processing such data, but when using RNN for feature extraction, a large number of parameters will be generated and the training speed will slow down. Since CNN shares convolution kernel parameters and can process high-dimensional data, this paper uses CNN architecture as feature extractor.

Assume that the input test case is represented by the word embedding matrix projection, and $d_w$ is the word vector dimension. Set a convolution kernel as $W_c \in \mathbb{R}^{d_w \times L}$, and set the first dimension of the kernel to be the same length as the word vector of the sentence vector, thus a feature mapping is obtained:

$$c = f(x \otimes W_c + b)$$

(5)

Where $\otimes$ is the convolution operation, function $f$ is the activation function, $b$ is the bias, and then the maximum pooling layer is used, namely:

$$\tilde{c} = \text{max}_{c} \text{pool}[c]$$

(6)

In this paper, the sizes of $a_1$ detectors are set, and each detector altogether uses $a_2$ convolution kernel. Then, the final generated sentence feature vector can be expressed as:

$$f = F(x) \in \mathbb{R}^{a}$$

(7)

Where $a = a_1 \times a_2$, given the feature vector representation of test cases, the feature space of test cases is constructed, and both the real test case set $X$ and the generated test case set $Y$ satisfy a certain distribution in the feature space.

$$c(x, y) = 1 - \frac{F(x, \theta), F(y, \theta)}{\|F(x, \theta)\|_2 \cdot \|F(y, \theta)\|_2}$$

(8)
Where \( F(\theta) \) represents a fully connected network. The function of using the fully connected neural network is to keep the length of the transformed vector output in the output layer consistent with the length of the input test case feature vector.

\[
prox_{g_f}(v) = \arg \min_{x \in X} \left( f(x) + \frac{1}{2\lambda}d(x, v) \right)
\]

Through the obtained core function, the main iterative formula of the optimization algorithm using the IPOT near endpoint method can be obtained as follows:

\[
x^{(t+1)} = \arg \min_{x \in X} f(x) + \beta'd(x, x')
\]

Iteration formula of IPOT with cost change distance is shown as follows:

\[
T^{(t+1)} = \arg \min_{T \in \prod (P_y, P_x)} <C, T> + \beta D_h(T, T^{(t)})
\]

Where \( \prod (P_y, P_x) \) represents the set of all transmission matrices transmitted from the probability distribution of generated test cases \( P_y \) to the probability distribution of real test cases \( P_x \), and \( \beta \) represents the penalty factor of near-end optimization.

The final cost variation distance is shown as 12, which is used to represent the distribution distance between a mini-batch generated test case set and the real test case set in the feature space, so as to optimize the generation network and the discriminant network.

\[
D_{CVD}(P_z, P_{real}) = \min_{T \geq 0} \sum_{i=1}^{m} \sum_{j=1}^{n} T_{ij} c(x_i, y_j)
\]

### 3.3. Data sets and model training results

In this paper, 1000 pieces of data obtained from Modbus protocol capture package were constructed by Wireshark. 80% of the data set was used as the test set, and 20% as the verification set. The model training was implemented on Windows10 system and Tensorflow framework and its Python binding. The result of model training is shown in Figure 2. It can be seen from the figure that the generated test cases and the original test cases have the same structure.
4. Simulation and performance analysis

In order to verify the effectiveness of the proposed method and test its performance, open-source fuzzy testing tools Peach, AFL, Sulley and the method proposed in this paper were used to test Modbus. Firstly, Modbus Poll and Modbus Slave are used to establish normal data communication. The experiment mainly verifies the performance of the method proposed in this paper and the traditional tools Peach, Sulley and AFL in the generation of variation data, and uses the test cases generated in the third section to attack the protocol system to detect vulnerabilities.

The method proposed in this paper, Peach, Sulley and AFL are compared with the acceptance rate of generated test cases. The test case acceptance rate refers to the ratio of the number of test cases accepted by the test target to the total number of test cases sent. The higher the value, the higher the rate of validity of the generated data. As shown in Figure 3, the acceptance rate of test cases of the method proposed in this paper reached 91.5% within one hour, and the acceptance rate of test cases of Peach, Sulley and AFL was 47.1%, 36.5% and 56.7% respectively. The acceptance rate of test cases of the method proposed in this paper was significantly higher than that of the traditional method.
As can be seen from Figure 4, the vulnerability detection rate gradually increases over time, indicating that the number of errors caused by test cases is increasing. In the experiment, we counted the number of exceptions thrown, including the same exception being thrown more than once. Compared with traditional methods, the method proposed in this paper has a higher vulnerability detection rate.

![Figure 4. Vulnerability detection rate](image)

5. conclusion
This paper aims at the problem of low diversity and usability of test cases generated by existing fuzzy testing tools. In order to improve the test efficiency of the protocol system and the vulnerability detection rate, this paper use a test case generation method based on the generative confrontation network. The method was compared with existing vulnerability detection tools AFL, Peach and Sulley on Modbus simulation software. The effectiveness of the method in this paper is verified by the acceptance rate of test cases and the ability to discover vulnerabilities. The test case rate was increased to 91.5%, and the vulnerability detection efficiency was also higher than that of traditional tools. The experimental results show that the test cases generated by the method proposed in this paper have high acceptance rate of test cases and vulnerability detection rate.

Acknowledgments
This work was financially supported by 2020 Yubei District Big Data Intelligent Technology Special Key Project: Research and Development and Application of Multi-sensor Intelligent Interactive Terminal of the Internet of Things, Project Number: 2020-02.

References
[1] B. Zhang, J. Ye, C. Feng and C. Tang, "S2F: Discover Hard-to-Reach Vulnerabilities by Semi-Symbolic Fuzz Testing," 2017 13th International Conference on Computational Intelligence and Security (CIS), Hong Kong, China, 2017, pp. 548-552, doi: 10.1109/CIS.2017.00127.
[2] F. Jingqi, C. Yifan and Y. Mubai, "Research on Fuzzing Test Case Generation Method Based on Gaussian Mixture Model," 2019 Chinese Control And Decision Conference (CCDC), Nanchang, China, 2019, pp. 2674-2678, doi: 10.1109/CCDC.2019.8833034.
[3] M. Li, L. He, Y. Teng, X. Wang, J. Zhang and S. Qing, "Research on network protocol vulnerability discovery based on fuzz testing," 2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, 2017, pp. 1354-1358, doi: 10.1109/ITNEC.2017.8285016.
[4] M. Arjovsky, S. Chintala, L. Bottou, "Wasserstein GAN," arXiv preprint, arXiv: 1701.07875, 2017.
[5] A. Akande and F. M. Salem. Simplified long short-term memory recurrent neural networks: part ii. The 16th Int’l Conference on Information & Knowledge Engineering, pages 78-81, 2017.