ChinaRxiv

AI translation · View original & related papers at
chinarxiv.org/items/chinaxiv-202009.00100

A Fuzz Testing Method Combining Genetic Algorithm for Industrial Control Protocols (Postprint)

Authors: Zhang Guanyu, Shang Wenli, Zhang Bowen, Chen Chunyu, Liu Zhoubin, Zhang Rui

Date: 2020-09-28T00:00:00+00:00

Abstract

Fuzzing demonstrates excellent applicability in vulnerability discovery for industrial control protocols; however, traditional fuzzing suffers from significant drawbacks, including high overhead in test case generation and elevated failure rates. To address these challenges, we designed GA-fuzzer, an industrial control protocol fuzzer that integrates genetic algorithm with fuzzing, and introduced a dimension-transformation-based test case space model along with the concept of danger points. Within GA-fuzzer, we constructed a more effective dynamic fitness function and designed dynamic mutation and crossover operators to optimize test cases. Under identical experimental conditions, we evaluated the target using both the open-source fuzzing tool Peach and GA-Fuzzer. The results demonstrate that GA-fuzzer effectively mitigates the premature convergence issue inherent in traditional genetic algorithms. Furthermore, compared with Peach, GA-fuzzer reduces the number of test cases required to achieve equivalent testing objectives by 27.20% and decreases testing time by 34.82%.

Full Text

Preamble

Vol. 38 No. 3 *Application Research of Computers* ChinaXiv Partner Journal

A Fuzzy Testing Method for Industrial Control Protocol Combining Genetic Algorithm

Zhang Guanyu[1,2,3], *Shang Wenli*[1,3,4,5]†, *Zhang Bowen*[1,3,4,5], *Chen Chunyu*[1,3,4,5], *Liu Zhoubin*[6], *Zhang Rui*[2]

(1. Key Laboratory of Networked Control Systems, Chinese Academy of Sciences, Shenyang 110016, China; 2. School of Information & Control Engineering,

Shenyang Jianzhu University, Shenyang 110168, China; 3. Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China; 4. Institutes for Robotics & Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110169, China; 5. University of Chinese Academy of Sciences, Beijing 1000396, China; 6. Electric Power Research Institute of State Grid Zhejiang Electric Power Co., Ltd., Hangzhou 310014, China)

Abstract: Fuzzy testing demonstrates strong applicability in vulnerability mining for industrial control protocols, yet traditional approaches suffer from high test case generation workload and elevated failure rates. To address these limitations, we designed GA-fuzzer, an industrial control protocol fuzzer that integrates genetic algorithms with fuzzy testing, introducing a case space model based on dimensional transformation and the concept of dangerous points. Within GA-fuzzer, we constructed a more effective dynamic fitness function and designed dynamic mutation and crossover operators to optimize test cases. Under identical experimental conditions, we compared our approach against the open-source fuzzer Peach. Results show that GA-fuzzer effectively mitigates premature convergence issues inherent in traditional genetic algorithms. Compared to Peach, it reduces the number of test cases required to achieve equivalent test coverage by 27.20% and decreases testing time by 34.82%.

Keywords: industrial control protocol testing; genetic algorithm; fuzzy testing; vulnerability mining

0 Introduction

Industrial Control Systems (ICS) serve as the "control brain" of industrial internet systems, encompassing supervisory control and data acquisition (SCADA) systems, distributed control systems (DCS), process control systems (PCS), programmable logic controllers (PLC), and remote terminal units (RTU) [1]. These systems are widely deployed in critical industries that support modern society, including petrochemicals, machinery manufacturing, electric power, water conservancy, financial services, commercial facilities, transportation, and communications [2]. Broadly speaking, any application scenario involving industrial control equipment relies on ICS support, making it one of the fundamental conditions for societal operation.

Traditional ICS designs prioritized functionality over security considerations and typically employed proprietary systems. Such systems were internally closed with limited external connectivity, resulting in minimal robust defensive measures for many critical infrastructure control systems [3]. However, a series of industrial control security incidents have demonstrated that ICS security requires continuous improvement and strengthening, necessitating comprehensive protection measures that address all potential attack vectors [4,5].

Communication protocols, as the carriers for data collection and control command transmission in ICS, are critical to security. Traditional industrial control protocols often lack adequate security considerations, such as insufficient

information encryption, absence of protocol authentication, and public protocol documentation, making systems employing these protocols vulnerable to exploitation [6]. Such attacks are low-cost yet high-yield; attackers who master protocol specifications can forge malicious data packets targeting protocol vulnerabilities, thereby compromising the entire ICS. Consequently, vulnerability mining for industrial control protocols constitutes an indispensable component of ICS security assurance.

International research on communication protocol security began earlier and has matured rapidly, yielding various testing tools for practical application. Domestic research on industrial control protocol fuzzing has also progressed steadily in recent years. Artemios G. Voyiatzis et al. [7] designed a fuzzer for identifying security vulnerabilities in Modbus/TCP, incorporating a detection phase to enhance predictive capability and enable timely adjustment of attack vectors, thereby significantly reducing fuzzing randomness. Reference [8] proposed utilizing genetic algorithm mutation operators to process multi-dimensional inputs, revealing relationships between input elements and unsafe functions in target applications to trigger potential vulnerabilities. Banks G et al. [9] addressed limitations in applying fuzzing to protocol security testing by constructing SNOOZE, a network protocol fuzzer that enables testers to describe protocol states and corresponding message formats, facilitating targeted testing of specific protocol states. Kang J et al. [10] presented a hybrid fuzzing system combining black-box and white-box testing, employing weakness analysis methods to accurately detect security vulnerabilities and uncover potential threats. Domestically, Lai Yingxu et al. [11] introduced the concept of mutation factors—characteristics of industrial control system vulnerabilities—to generate test case data for Modbus/TCP protocol testing, validating effectiveness through bypass monitoring. Tu Ling et al. [12] proposed a parallel hybrid test case generation method combining protocol deformation and dynamic features to improve coverage while reducing false positive rates. Zhang Yafeng et al. [13] developed a state-based industrial control protocol fuzzing technique, designing a protocol state machine-based test sequence generation algorithm that achieved higher vulnerability hit rates and target coverage by addressing previous limitations in considering protocol interaction states.

1 Background

1.1 Modbus Communication Protocol

First published by Schneider Electric in 1979, the Modbus communication protocol has become the most common data communication method in industrial control systems due to its simple deployment and easy maintenance. It is essentially a master-slave protocol that defines how a control request device accesses other service devices [14].

Based on differences in data format and physical interface, Modbus can be categorized into Modbus RTU and Modbus ASCII for serial links, and Mod-

bus/TCP for TCP/IP transmission. To facilitate data transfer across different modes, Modbus defines a communication-layer-independent Protocol Data Unit (PDU). In various transmission modes, data can be properly transmitted by simply adding corresponding additional fields to the PDU.

Figure 1 illustrates the data frame structure in TCP mode. The Application Data Unit (ADU) comprises six components: a transaction identifier that numbers each communication event (identical for request and response), a protocol identifier indicating the protocol type (0x0000 for Modbus), a length field storing the number of bytes including the unit identifier and PDU, a unit identifier serving as system-internal routing (i.e., serial link device address encoding), and finally the PDU storing communication data. In this mode, the PDU from the serial protocol is extended with the transaction identifier, protocol identifier, length information, and unit identifier at its head, then encapsulated with a TCP header to enable Modbus transmission over Ethernet. All tests in this paper were conducted in this transmission mode.

[Figure 1: see original paper] Modbus TCP/IP data frame structure

1.2 Fuzzy Testing

First proposed by Professor Barton Miller in 1989, fuzzy testing constructs large volumes of malformed data as input to test targets and analyzes vulnerabilities by observing responses. Its strong adaptability and portability stem from requiring no internal target knowledge—only input data format understanding—to construct and transmit test cases.

While fuzzy testing processes may vary depending on test targets, test case generation methods, target characteristics, researcher expertise, and data formats, all approaches share six fundamental steps: (a) identify test target information including target type and historical vulnerability data; (b) determine test data inputs, encoding formats, and all required data types; (c) design case generation methods and construct test case sets based on input information; (d) transmit test cases to targets according to data transmission specifications; (e) establish monitors to detect test execution results; and (f) determine exploitable vulnerabilities based on final results.

Current research on fuzzy testing for industrial control protocol vulnerability mining primarily focuses on developing more effective case generation methods and reducing fuzzing randomness to decrease test case quantities and improve efficiency. Building upon existing research, this paper proposes a fuzzy testing method for industrial control protocol vulnerability mining that combines genetic algorithms. The method designs a dynamic fitness function to control case generation and introduces concepts of case space and dangerous points to evaluate case quality. By calculating similarity and importance characteristics between cases and integrating them with the dynamic fitness function, the approach effectively reduces test case failure rates and minimizes test case quantities.

2 GA-fuzzer Design and Implementation

This paper employs fuzzy testing methods tailored to Modbus protocol characteristics. Traditional fuzzing suffers from high test case failure rates, generating numerous invalid packets when mining industrial control system protocol vulnerabilities, resulting in low efficiency [16]. To address this issue while automating and optimizing the process, we integrate genetic algorithms into fuzzy testing, adapting genetic strategies for test case generation and optimization.

Genetic algorithms represent one of the earliest global optimization algorithms, mimicking natural selection to search for optimal individuals within each generation and ultimately finding optimal solutions through iterative selection, crossover, and mutation operations [20]. However, traditional genetic algorithms gradually converge during population iteration, increasing similarity among generated cases—making them unsuitable for direct test case generation. Consequently, genetic strategies require appropriate modifications.

2.1 Improvements and Optimizations

Our improvements target premature convergence in traditional genetic algorithms and address high failure rates and low efficiency in fuzzy testing. The main contributions include:

Test Case Encoding: Industrial control protocol communication messages are essentially ordered sequences of symbols, representable as one-dimensional vectors where each element corresponds to a message field containing integers, characters, separators, or formatting characters. A genetic algorithm generation can thus be represented as a matrix (Equation 1). Initial population individuals employ binary encoding, with each protocol byte represented by 8 binary bits. After byte stream conversion by the test server, cases can be transmitted to Modbus clients.

Selection Operator: Selection represents a crucial component where the fitness function guides algorithm direction. An appropriate fitness function is essential for Modbus protocol fuzzing. We first introduce two key concepts: case space and dangerous points.

Each Modbus/TCP message field has fixed functions and specific ranges (e.g., slave device address codes range from 0-255). By normalizing message fields and mapping them to corresponding dimensional space, each test case corresponds to a unique point in case space. For any test case, the coordinate value of its ith field is calculated using Equation 2, where max(c_i) and min(c_i) represent the maximum and minimum values of the ith field within the population.

The test case mapping to case space is expressed as . After mapping all cases, each possesses a unique spatial coordinate, enabling Euclidean distance between spatial points to represent similarity between two cases and , described by Equation 3 (where m is the smaller dimension between and).

High-dimensional test cases undergo dimensionality reduction, which sacrifices some data information. However, since this data primarily consists of Modbus protocol message storage fields, negative impacts on similarity and experimental results can be mitigated.

Dangerous Points: Response status determines whether a case qualifies as a dangerous point when sent to the test server. Four response states exist: normal response (case functions properly, server responds normally), abnormal response (case fields are abnormal but server recognizes and responds), request timeout (unknown exception prevents server response), and other (unknown exception causes server to close communication or deny access).

Cases with normal or abnormal responses constitute valid Modbus messages recognizable by the server. Request timeouts indicate non-compliant fields that the server cannot recognize, classifying them as invalid cases. The "other" state indicates severe server errors (e.g., unresponsive to subsequent cases, closed or denied service), with high probability of triggering server anomalies—such cases are marked as dangerous points.

Dangerous point coordinates are marked in case space. During genetic algorithm iteration, distances between individuals and dangerous points determine importance. With q dangerous points in space, each case is assigned to its nearest dangerous point, dividing the population into q subpopulations corresponding one-to-one with dangerous points.

Average Case Similarity: Maximizing coverage is essential in test case generation. Average case similarity serves as a diversity indicator, traditionally described through average inter-individual distances. However, this requires computing distances between each individual and all others, resulting in extensive redundant calculations and low efficiency. We propose a new method that accurately reflects population distribution while reducing computational overhead.

First, summing and averaging all four fields across the population yields a central case , with each field calculated using Equation 4 (where m is the current population size and x represents case fields). Each case' s average similarity can then be expressed as its distance to this central case using Equation 3. This approach reduces time complexity from $O(n^2)$ to $O(n)$, providing significant efficiency improvements for large n.

Dynamic Fitness Function: Based on case space, dangerous points, and average similarity, we propose a novel dynamic fitness function to evaluate case quality, influenced by dangerous point presence in case space:

(a) When no dangerous points exist in population case space (q=0), similarity dominates selection. The fitness function for the kth test case equals its average similarity, calculated using Equation 5.

(b) When dangerous points appear (q>0), the population first divides into q subpopulations based on nearest dangerous point assignment. Indepen-

dent selection then occurs within each subpopulation using importance as the fitness function—specifically, the distance between individuals and their subpopulation' s dangerous point. The fitness function for the jth individual in the subpopulation corresponding to the tth dangerous point is given by Equation 6, where d_{tj} is the distance to the dangerous point and d_{tmax} is the maximum distance within that subpopulation.

Since dangerous points may not represent exploitable vulnerabilities, each is assigned a 20-generation survival cycle to improve testing efficiency. Upon reaching maximum survival, dangerous points are eliminated after recording test information.

Crossover and Mutation Operators: Crossover and mutation probabilities (P_c, P_m) are critical for maintaining population diversity. Excessive similarity during iteration negatively impacts diversity. Reference [20]' s adaptive crossover and mutation operator design effectively adjusts probabilities based on population state in real-time. We combine this with our case population state to construct adaptive probability functions suitable for Modbus fuzzing, as shown in Equation 7:

where d_{ave} is the average distance among all individuals in the current (sub)population, d_{max} and d_{min} are maximum and minimum distances, and parameters a, b judge excessive similarity (empirically set to a=0.72, b=0.25). These constructed probabilities vary with population state: when d_{ave} is small (high similarity), larger P_m values increase mutation probability to enhance diversity; when d_{ave} is large, the opposite occurs.

Based on these theoretical methods, the genetic algorithm flow is illustrated in Figure 2, comprising: (a) constructing initial case populations for generation and optimization; (b) selecting fitness functions based on case space state and dangerous point count; (c) using average similarity as fitness function and conducting similarity checks when no dangerous points exist; (d) dividing population into subpopulations via K-means clustering and using individual importance as fitness function when dangerous points exist; (e) performing genetic operations within populations or subpopulations; (f) checking termination conditions—if unsatisfied, updating dangerous point information and returning to step (b), otherwise ending; (g) terminating algorithm and outputting test logs.

Compared to traditional genetic algorithms, our primary enhancement is the dynamic fitness function that selects different functions based on dangerous point presence, preventing high similarity and rapid failure rate increases after convergence. Dynamic mutation and selection probabilities adjust population diversity according to state, improving vulnerability hit probability while maintaining coverage.

2.2 GA-fuzzer Design

Integrating improved genetic algorithms with fuzzy testing, GA-fuzzer' s architecture is shown in Figure 3:

(a) **Protocol Analysis:** Analyze Modbus message characteristics, marking field data types, functional properties, and value ranges while capturing normal communication messages to construct high-quality initial test populations.

(b) **Case Generation:** The initial population enters the case generation unit, which handles dangerous point marking, dynamic fitness function calculation, genetic operations, and case space updates.

(c) **Simulation Testing:** After population optimization, all individuals are transmitted to the test target via a Modbus simulation server.

(d) **Anomaly Detection:** Each test case' s execution results are recorded in test logs, with dangerous points identified to update dangerous point information, fitness functions, and mutation/crossover probabilities.

(e) **Result Analysis:** Final results are analyzed to determine exploitable security vulnerabilities.

3 Experiments and Results Analysis

To validate effectiveness and performance, we compared our method against the open-source fuzzer Peach. Developed by Michael Eddington in 2004, Peach tests file formats, communication protocols, and APIs. Through Peach Pit file configuration—including data types, mutation strategies, and data lengths—it enables industrial control protocol fuzzing [19].

Experimental Environment: Hardware platform: Intel(R) Core™ I5-3470 CPU @ 3.20GHz; 3.79GB RAM; Windows 10. Development language: Python 3.6. Testing employed Modbus communication simulation software. Normal Modbus Poll-to-Slave communication was established, with Wireshark capturing normal messages for feature analysis to construct initial populations as inputs for all test methods.

Key identical parameters across both test groups: population size = 10; standard mutation probability = 0.03; standard crossover probability = 0.76.

The termination condition was vulnerability discovery. To evaluate our case optimization method' s performance, we statistically analyzed average case similarity per generation for both methods, shown in Figure 4 (calculated using Equation 8).

[Figure 4: see original paper] Trend of Case Average Similarity

Analysis reveals that Peach' s mutation probability, fixed by configuration, cannot adjust based on case generation state. While maintaining low average simi-

larity, its weak targeting yields random case generation. GA-fuzzer dynamically adjusts population state—when dangerous points increase, genetic algorithm convergence causes cases to converge toward their dangerous points, and appropriate mutation probability adjustment improves vulnerability hit probability near dangerous points.

Some populations exhibit similarity exceeding 0.7 (e.g., around generation 770). Data analysis shows this occurs when dangerous points exist and convergence approaches. Per our dynamic fitness function, case generation then targets dangerous points to increase vulnerability hit probability, naturally elevating inter-case similarity.

After multiple tests in identical environments, statistical analysis of generated cases yielded average metrics shown in Table 1.

Comparison of Experimental Results

We also selected representative consecutive 100-generation populations from three test groups, normalized and abstracted into three-dimensional case space to visualize case distribution, shown in Figure 5.

[Figure 5: see original paper] Distribution of Cases of Each Method in the Case Space

Open-source fuzzing frameworks generate uniformly distributed cases (Figure 5a) due to lack of control strategy, prioritizing coverage over targeting. While maintaining individual differences, this approach lacks focus, resulting in lengthy testing times and excessive test cases. Figure 5b shows GA-fuzzer generated cases. In the sample data containing two dangerous points, population individuals first cluster according to nearest dangerous points, then genetically converge within each class, enabling simultaneous targeted testing of multiple suspected vulnerabilities to improve efficiency. When no dangerous points exist or all survival cycles end, case generation trends toward high coverage and low similarity, ensuring comprehensive testing. Dynamically updating fitness functions and genetic operators based on dangerous point presence achieves both high coverage and strong targeting, effectively reducing test case failure rates.

Table 1 demonstrates that in identical environments, GA-fuzzer achieves successful test completion with approximately 34.82% shorter testing time and 27.20% fewer test cases compared to Peach.

4 Conclusion

This paper applies genetic algorithms to industrial control protocol fuzzing, designing a fuzzer that dynamically adjusts genetic algorithm fitness functions, crossover, and mutation probabilities using case space. This provides a novel approach for protocol vulnerability mining: the case space and dangerous point concepts mitigate fuzzing' s strong randomness while solving traditional genetic algorithm premature convergence. Dynamic fitness functions enable real-time

genetic operation adjustment based on population state, generating more targeted test cases that reduce failure rates and improve efficiency. Future work will further refine the case space model, such as more carefully handling data loss from dimensionality reduction and optimizing dynamic fitness function adjustment strategies to further enhance test case quality.

References

[1] Yan Tengfei, Shang Wenli, Zhao Jianming, et al. Anomaly detection algorithm based on OCSVM double contour model of genetic algorithm optimization for industrial control system [J/OL]. *Application Research of Computers*, TP. 20180811: 1-3.

[2] An Gaofeng, Zhu Changming, Lei Xiaofeng, et al. Information security policy and standard system of industrial control in China [J]. *Research on Information Security*, 2018, 4(10): 959-964.

[3] Wan Ming, Shang Wenli, Zeng Peng, et al. Modbus/TCP communication control method based on deep function code inspection [J]. *Information and Control*, 2016, 45(02): 248-256.

[4] Xiong Qi, Peng Yong, Yi Shengwei, et al. Survey on the fuzzing technology in industrial network protocols [J]. *Journal of Chinese Computer Systems*, 2015, 36(3): 497-502.

[5] Edmonds J. Security analysis of multilayer protocols in SCADA networks [D]. Department of Computer Science, University of Tulsa, Tulsa, Oklahoma, 2006.

[6] Huitsing P, Chandia R, Papa M, et al. Attack taxonomies for the Modbus protocols [J]. *International Journal of Critical Infrastructure Protection*, 2008, 1(none): 37-44.

[7] Voyiatzis A G, Katsigiannis K, Koubias S, et al. A Modbus/TCP fuzzer for test internetworked industrial systems [C]. 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA). IEEE, 2015.

[8] Wu Zhiyong, Atwood J W, Zhu Xueyong. A new fuzzing technique for software vulnerability mining [J]. *Journal of Communication and Computer*, 2011(2): 88-95.

[9] Banks G, Cova M, Felmetsger V, et al. SNOOZE: Toward a stateful network protocol fuzzer [C] *Information Security*, 9th International Conference, ISC 2006, Samos Island, Greece, August 30-September 2, 2006, Proceedings. DBLP, 2006.

[10] Kang J, Park J H. A secure-coding and vulnerability check system based on smart-fuzzing and exploit [J]. *Neurocomputing*, 2017, 256(20): 23-30.

[11] Lai Yingxu, Yang Kaixiang, Liu Jing, et al. A vulnerability mining method for industrial control network protocol based on fuzz test [J/OL]. *Computer*

Integrated Manufacturing Systems: TP20180511, 1-22.

[12] Tu Ling, Ma Yue, Cheng Cheng, et al. Hybrid protocol deformation based web security fuzzy test and utility evaluation approach [J]. *Computer Science*, 2017, 44(05): 141-145.

[13] Zhang Yafeng, Hong Zheng, Wu Lifa, et al. Form-syntax based fuzzing method for industrial control protocols [J]. *Computer Science*, 2017, 44(05): 132-140.

[14] Cheng Yang, Liu Xueping, Zhang Tao. Design of an industrial control system based on MODBUS protocol [J]. *Machinery Design & Manufacture*, 2011(01): 1-3.

[15] Cui Xin, Wen Yanlong. Analysis and application of fuzzy test protocol for industrial control systems [J]. *China Information Security*, 2018(9): 73-78.

[16] Zhang Yafeng, Hong Zheng, Wu Lifa, et al. Form-syntax based fuzzing method for industrial control protocols [J]. *Application Research of Computers*, 2016, 33(8): 2433-2439.

[17] Yu Haibin, Zeng Peng, Shang Wenli, et al. CN 105721230, A fuzzy test method for Modbus protocol: China [P/OL]. 2016.06.29.

[18] Li Zhu. Automatic test-case generation based on adaptive genetic algorithm [J]. *Application of Computer System*, 2016, 25(01): 192-196.

[19] Yi Shengwei, Zhang Chongbin, Xie Feng, et al. Security analysis of industrial control network protocols based on Peach [J]. *Journal of Tsinghua University: Science & Technology*, 2017, 57(1): 50-54.

Note: Figure translations are in progress. See original paper for figures.

Source: ChinaXiv —Machine translation. Verify with original.