UDC 004.93
https://doi.org/10.31891/csit-2022-3-9

ROMAN DIACHOK, HALYNA KLYM
National University "Lviv Polytechnic", Lviv, Ukraine

DYNAMIC SEARCH FOR ERRORS IN INDUSTRIAL INTERNET PROTOCOLS FOR APPLICATION IN MULTISENSOR CONTROL SYSTEMS

Based on the considered theory of dynamic damage analysis of industrial Internet protocols and determination of the necessary data of dynamic multimodal communication of the sensor, the method of fuzzy tests in combination with dynamic multimodal data transmission for jamming in multisensor control systems is proposed. The proposed method traces program execution, finds input fields affecting conditional branches through dynamic damage analysis, and captures the dependency of conditional branches to appropriately control test case grammar generation, increasing deep-level code execution. The results of the comparative experiment confirm that the method to some extent improves the validity of test cases and the speed of code coverage, as well as increases the probability of detecting anomalies in the implementation of the protocol.

To evaluate multi-sensor computerized systems from the point of view of "fuzzy intelligence", studies were conducted to solve the problem of low code coverage caused by repeated execution of test sequences on the same path, starting from the system program level in the implementation of industrial Internet protocols and the prerequisite of obtaining affordable weekends program codes, or an executable binary file. The paper proposes a method that is combined with dynamic multimodal transmission of sensor data in a fuzzy processing program. It provides program execution, protocol implementation, finds input fields that affect conditional branches using dynamic inconsistency analysis, and captures dependency relationships between conditional branches to control test generation.

Key words: industrial Internet of Things, fuzz-testing, corruption of industrial Internet protocols, inconsistency of industrial Internet protocols

РОМАН ДЯЧОК, ГАЛИНА КЛИМ
Національний університет «Львівська політехніка»

ДИНАМІЧНИЙ ПОШУК ПОМИЛОК У ПРОМИСЛОВИХ ПРОТОКОЛАХ ІНТЕРНЕТУ ДЛЯ ЗАСТОСУВАННЯ У МУЛЬТИСЕНСОРНИХ СИСТЕМАХ КЕРУВАННЯ

У роботі на основі розглянутої теорії динамічного аналізу пошкоджень промислових протоколів інтернету та визначення необхідних даних динамічного мультимодального зв'язку датчика запропоновано метод нечітких тестів у поєднанні з динамічною мультимодальною передачею даними для застування у мультисенсорних системах керування. Запропонований метод відстежує виконання програми, знаходить поля введення, що впливають на умовні гілки, за допомогою динамічного аналізу пошкоджень і фіксує залежність умовних гілок, щоб відповідним чином керувати генерацією граматики тестового прикладу, збільшуючи можливість виконання кодів на глибокому рівні. Результати порівняльного експерименту підтверджують, що метод певною мірою покращує валідність тестових випадків і швидкість охоплення кодів, а також підвищує ймовірність виявлення аномалій у реалізації протоколу.

Для оцінки мультисенсорних комп'ютеризованих систем з точки зору «нечіткого інтелекту» були проведені дослідження з метою вирішення проблеми низького рівня покриття коду, спричиненої повторним виконанням тестових послідовностей на одному шляху, починаючи з рівня системної програми при реалізації промислових протоколів Інтернету та передумови отримання доступних вихідних програмних кодів, або виконуваного двійкового файлу. У роботі запропоновано метод, який поєднаний з динамічною мультимодальною передачею даних датчиків у програмі нечіткої обробки. Він забезпечує виконання програми, реалізацію протоколу, знаходить поля введення, які впливають на умовні розгалуження, за допомогою динамічного аналізу невідповідностей, а також фіксує зв'язок залежності між умовними розгалуженнями для керування створенням тестів.

Ключові слова: промисловий Інтернет речей, fuzz-тестування, пошкодження промислових протоколів інтернету, невідповідність промислових протоколів Інтернету

Introduction

The Industrial Internet of Things (IIoT) or Industrial Internet of Things (IoT) has been implemented using a variety of automation components for data acquisition, control, monitoring, and a number of other functions [1]. It is known that a typical industrial architecture of Internet communications is a three-level structure from high to low, respectively, an enterprise network, a monitoring network, and a control system network [1, 2]. PSI refers to a system consisting of computer equipment and an industrial production control unit, which includes SCADA. [3].

In a typical electrical IoT, each link (from electricity generation to electricity use) has a corresponding electrical terminal of the IoT, such as a programmable logic controller (PLC), intelligent substation equipment, monitoring and control devices, and other types of equipment for data collection, instruction generation, remote control, etc. , which can be combined into a multi-sensor system. The transfer of data flow and control between the IoT terminal and the main station via industrial Internet networks is implemented. The application component, which runs in the terminal, is responsible for the analysis and processing of industrial Internet protocols.

In recent years, in conditions of rapid development of various information technologies, industrialization and informatization have been closely integrated. More modern information technologies were applied to traditional IoT. At the same time, various standardized communication protocols and network switching architectures are being

popularized in IoT. With the addition of advanced information technology and communication network technology (such as Ethernet), the openness of industrial Internet control systems has been greatly expanded, and exposed to greater security risks.

In 2015, the Ukrainian electric power industry was attacked by the Black Energy malware [4], which gained remote access to system control, which caused the failure of the power grid's SCADA host system and then a large-scale blackout. In 2017, security manufacturer ESET announced a win32/Industroyer tool that directly attacks electrical IoT. The tool can cause the tripping of the transformer substation by operating the circuit breaker. In 2018, there were many warning messages about illegal Internet access on the IoT remote electrical monitoring platform in China. After analysis, the reason was found to be that the manufacturer ran and maintained the product server remotely by opening the file sharing function, and thus provided access to the system on the public network for a long time, causing serious hidden dangers. Over the past 10 years, many security incidents, potential and possible dangers have made the IoT security situation much more serious [5–7].

In industrial Internet protocol implementations, terminal codes, protocol fields, are generally trustworthy, but an attacker can control program execution by changing the values of data fields using protocol flaws, which then affect the entire system. For example, the destination address parameters of a bypass command usually come from trusted data sources in the application, not from external untrusted input such as Industrial Internet Protocol input. However, an attacker can overwrite the destination address of an instruction due to a system vulnerability and then control the running IoT process. Professor Jonathan M. Garibaldi proposed a non-differentiated conceptual framework as a key component in the evaluation of computerized decision support systems. Practical studies show that human experts are not perfect and there are techniques that allow fuzzy systems to model human-level performance, including variability.

Therefore, the need to use "fuzzy intelligence" to evaluate multi-sensor computerized systems is relevant from two aspects: (i) fuzzy methodology is necessary as a knowledge-based system to identify the cause of uncertainty; (ii) when evaluating intelligent multisensory systems, ambiguity is necessary, which is recognized as an imperfect implementation [8].

Purpose is to improve the validation method and code coverage of test cases, as well as to increase the probability of detecting anomalies in the implementation of the IoT protocol for use in multi-sensor control system

Proposed technique

Security analysis of industrial control protocols.

An industrial control system communicates directly with the underlying equipment or data acquisition transducer using a protocol agreed upon by the communicating parties. Initially, most industrial control protocols were implemented only between a closed network and trusted software via a dedicated serial port. But in order to meet the increasingly complex requirements of industrial control systems, dedicated lines were gradually replaced. At the beginning of their design, industrial control protocols did not fully consider the conditions necessary to protect user security, such as encryption and authentication, and many protocols currently rely on TCP/IP. In this case, the data transmitted through the protocol cannot be protected from danger. An attacker only needs to master the protocol specifications and penetrate the industrial control network to falsify any data of the target device.

Common security flaws. Industrial control protocols are analyzed as follows:

1. The Modbus protocol does not have an authentication mechanism; communication is established on the basis of TCP/IP. Therefore, as long as the attacker obtains the device's network IP address, he can successfully connect directly using port 502. If the function code transmitted by the application data block is supported by the device's Modbus, a legitimate Modbus session can be established. Also, there is no message to check the Modbus/TCP protocol. Since the checksum is generated at the transport level, not the program level, it is easy to forge the command. At the same time, attackers who have connected to a target Modbus device can perform the functions of this device without permission. In addition, data encapsulated in Modbus is transmitted in clear text, and an attacker can obtain the message data using a network capture tool packet. However, the most dangerous aspect of Modbus is its programmability. Attackers can inject malicious code into the RTU or PLC to gain control.

2. DNP3 is a standard developed by IEEE PES based on IEC. Its security is expressed in the absence of authorization and encryption mechanisms. The function codes and data types are clearly defined in it, which makes it easier for attackers who intend to tamper with a DNP3 session.

3. Ethernet/IP is more modern compared to Modbus, but security issues remain. For example, when using the UDP protocol for real-time data transmission, it lacks a built-in network layer mechanism to ensure communication reliability and data integrity. An attacker can easily enter false data or use IGMP control messages to manipulate transmission paths.

The method of fuzz-testing of the industrial Internet protocol based on dynamic analysis

In view of the severe damage caused by IoT vulnerabilities, researchers have proposed many vulnerability detection techniques, such as dynamic analysis, symbolic execution, and fuzz test [9]. Compared to other techniques, fuzz testing requires little knowledge about the target, but can easily be scaled up to a large, reusable application. It has become the most popular solution for detecting vulnerabilities at the moment, especially in IIS. Regarding network

protocols, SPIKE [10] is the most representative phaser. The principle is to describe the protocol as a block sequence model and create change data in blocks, as well as split the message data structure and automatically create field length statistics after the change, which greatly improves the efficiency of test cases. However, such a principle is vague and indicates insufficient descriptiveness for limited communication in protocol messages. Later, Sally [11] and Peach [12] extended the SPIKE-based data model and added more descriptions of the dependency relationship between data blocks. To provide a more flexible and precise fuzzy structure, AFL [13] tracked the path coverage of each input using a lightweight toolkit in the source program and randomly assigned an ID to the basic blocks along the path, using a hashing mechanism to determine where any new path generated was located, and then used the input that generates the new path as the initial one. Combined with detailed information in the program, the method improves the speed of code coverage, but the Hash method can easily have collisions and thus cause the problem of false negatives, even if the input has reached a new path. Gan et al. proposed CollAFL [14], where the ID value of each basic block was allocated using a greedy algorithm and other methods to ensure the difference of the hash value on each side, and thus avoid the hash collision and realize a more accurate representation of the path coverage.

From the above studies, it can be concluded that there are the following problems of applying fuzzy testing methods to industrial Internet protocols: a) Low code coverage: Many errors can be caused only in the case of large path coverage. Although some methods use dynamic multimodal sensor communication, the data processed in the program is not reflected in the formation of the test cases directly, because of which most cases are executed multiple times along the same path as the input data and cover only a few paths. b) There are no unified models of description. Protocols even of the same type have different message forms, but current methods require the creation of different data models, which increases the model design burden; c) Insufficient matching of test cases. Test cases make it difficult to validate the application, resulting in too many invalid tests. The variational design of the strategy, not taking into account the characteristics of industrial Internet protocols, full consideration may lead to case redundancy for a large number of data samples, which will affect the testing efficiency.

Dynamic analysis

Dynamic pollution analysis proposed by Newsom is a method of tracking the information flow during program execution, i.e. tracking the transfer of data slots that are subject to analysis in the system and obtaining a detailed process of data processing by the target program. In the proposed method, we used the technology of obtaining dynamic data of a multimodal sensor in the program to provide the basis for further testing. The method of dynamic analysis of inconsistencies includes two parts, namely, identification of contamination data and monitoring of damage transmission path [15]. A key part of identifying damage data is identifying non-conformity data. If the data source is suspicious, the data it generates is corrupted, which, like the input data, must be monitored and analyzed in the running binary process, and then the message data constructed in the fuzz test can be treated as suspicious.

During program execution, the transfer of damage data is completed, instructions and data may participate in the operation as output operands of an arithmetic operation instruction or parameters of a data movement instruction whose output data is typically associated with damage tracking data and thus must be identified as a mismatch attribute. Figure 1 shows a simple example of the dynamic analysis process, a1 and a2 are the damage data, the arrow indicates the operation; the leading end represents the output operand, the tail end shows the output of the operation [16]. When executing a program on a variable V_1~V_8 damage data a_1 and a_2 are affected.. If A, as the corruption source set for each variable, is empty, we can assume that the variable is not corrupted. It can be seen in Figure 1 that the corruption source set V_8, which is the last output, is the union of the corruption source sets of the operands V_3 and V_6.

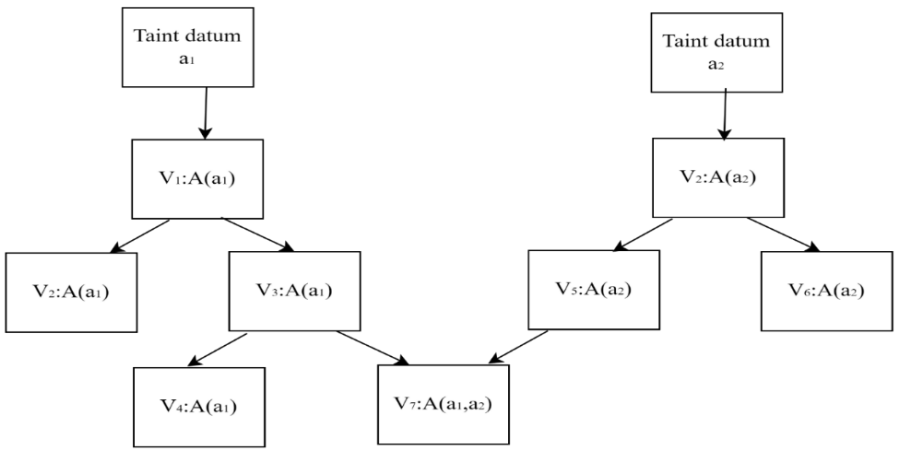

Fig. 1. An example of dynamic pollution

Using the dynamic corruption method, we aim to analyze the data flow and control flow, also known as explicit flow and implicit flow, during binary code execution. The first concerns the data dependency relationship.

The corruption information of the V_1 variable is sent to V_2 irectly by assignment or arithmetic operation, as shown in Figure 1. This corresponds to the control and dependency relationship of the conditional branches, that is, the error information of the V_1 variable is sent to V_2 indirectly through the associated condition expression. For the purpose of understanding, we made a simple analysis using the sample code shown in Figure 2 [17].

```
1.  begin
2. L1[ = get_input( ):
3. L2msg =uri = `  `:
4. L2 if (x ==`a`) {
5. L4  uri = `post` :
6.L5 msg = `a` :
7. L6}
8. L7 else if (x == `b`){
9.L8 urs = `post`:
10. L9  msg =`b` :
11. L10}
12. L11 send(uri, msg):
13. end
```

Fig. 2. Listing of dynamic analysis code

In the example above, the code implementation function involves inputting the variable *x*, generating the *msg* message, and passing it via message. In this process, *x* is identified as a reference point, and according to the data flow analysis method, «*msg*» is directly assigned as a constant «a» or «b», «*msg*» is identified as an *untainted* (untainted) attribute. However, the value of «*msg*» also depends on whether the conditional branches *x*==a and *x*==b are true or false, meaning that "msg" and x have a control-dependency relationship that belongs to an implicit flow defect. In practical applications, such codes have a security risk when they are used to implement network protocol communication. When a message is sent, an attacker can intercept the message and then output the value of the input x. Therefore, the message is definitely a reference point, so it should be captured and monitored. At the same time, false negatives may occur without taking into account the relationship of control and the dependence of the message. In contrast, the url value is independent of the L_3 and L_7, branches, so there is no harm even if it is identified as untainted data.

Method and principle

According to the Industrial Internet Protocol implementation program, relevant dynamic multimodal communication sensor data is obtained using dynamic discrepancy analysis to guide the creation of test cases. Figure 3 shows the specific process of testing the method. The proposed method uses many protocol messages as input to avoid the problem of insufficient test cases caused by a single data sample.

Definition 1. Dynamic interactive fields: When executing a given program for a conditional branch x_i (i-th execution of a conditional branch x), there exists $DIF(x_i) = \{F_j|F_j$ - is a protocol field affecting the execution of $x_i\}$, in which $DIF(x_i)$ - is a protocol field set..

Definition 2. Dependency and Control Relationship: During the execution of a given program, if there is a conditional branch y_i, that decides whether to execute x_i, we can assume that x_i depends on the dynamic control y_i and express it as $CDC(x_i) = \{y_i,T|F\}$, where $CDC(x_i)$ - is the set of branches satisfying the condition and the constraint (x_i) represents the constraint of the conditional branch x_i.

Definition 3. Dynamic control flowchart: For each program input, its execution path can be expressed using a dynamic control flowchart, in which the conditional branch x_i is the node, $DIF(x_i)$ is the dynamic interactive field of the node, and $CDC(x_i)$ is the side representing the dependency relationship of conditional branches.

We identify input protocol fields affecting conditional branches through dynamic damage analysis, perform damage identification processing for each field protocol input, and monitor for inconsistent data flow during program execution [17].

As for the dependence and relationship of program management, we fix it using the algorithm proposed in [17]. It should be noted that for industrial Internet protocols, fields typically have a checksum (eg CRC) that will cause a corrupted data stream if transmitted in the control stream. Since all fields are used as checksums, most conditional branches will be identified as corrupted data, in which case it is difficult to directly find the specific field that affects the conditional branches. In addition, due to the control dependency and relationship, the proposed fuzzy method indirectly considers the transmission disturbance in the control flow. Therefore, when using the dynamic mismatch analysis method, we only focus on corrupt transmissions in the data flow, rather than tracking transmission corruption in the control flow.

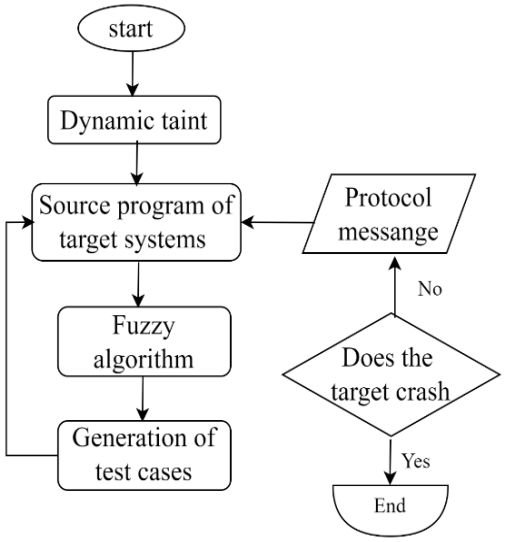

Fig. 3. Industrial Internet Protocol testing process

For conducting experimental research, must be crated program based on two algorithms for dynamic searching error in multisensor control systems. These algorithms are:

Algorithm 1. Fuzz test Algorithm combined with dynamic multi-modal sensor communication data.

Algorithm 2. Test case generation algorithm combined with dynamic multi-modal sensor communication data.

Algorithm 1 depicts the introduction of the fuzzy method in detail. The core function of dynamic Fuzz can execute program P, protocol G and protocol message I as input to process a large amount of protocol input, and the output will be abnormal information. First step of the algorithm is initializing the data structure that will be stored. The test cases generated, which will also be used as new input, are queued for storage and ease of recursion. Two parameters of program and input protocol obtains the *DIF(xi)* and *CDC(xi)* sets of each conditional partition in each program using the dynamic spot analysis method, and then constructs the corresponding dynamic control flowchart using *DIF(xi)* as a node and *CDC(xi)* as a side. Next step is performs a fuzzy operation from the very beginning point of the executable path in the control flow diagram. Then algorithm outputs *Constraint (xi)*, he corresponding constraint condition of node xi, as a test case parameter of the generation algorithm.

After the test case is generated, we need to use the test case as a new input to replace the value of the protocol field in the set *DIF* (x_i), and also enter and change all fields related to *Constraint* (x_i) o match the constraint condition. To ensure that the test case will be executed at a deeper level in the program and to improve the probability of finding new execution paths, the rest of the protocol fields unrelated to *DIF*(x_i) i *Constraint* (x_i), must also remain valid according to the grammar protocol. For example, in a two-field value input protocol, the start address and end address must change from the initial 1 and 2 to 3 and 6. Even if there is no restriction condition, the corresponding address data fields must be restored to the original normal size (increment from 1 to 3) , but if the two addresses change to invalid test cases 6 and 3 after fuzzification (start address < end address), the fields unrelated to *DIF*(x_i) and *Constraint* (x_i), still retain their normal values.

Next step is monitor if there is any abnormal condition such as a crash or memory leak during the execution of the test case. Although the test case is used as a new input, due to the overhead caused by the program execution, the proposed method does not obtain the dynamic multimodal sensor communication data of each conditional branch repeatedly. During the execution of the test case, the passed code paths are saving , put conditional branches without parsing into the input queue, and then determine the priority ranking according to the number of new branches. Algorithm stores the dynamic multimodal sequence of sensor communication data of each node, to evaluate whether the current node is generating a test case. If there is a dynamic multimodal sensor data sequence in the list and a generated test case that matches the *DIF(xi)*, i *Constraint (xi)* of the current node xi, respectively, then the algorithm will skip generating the test case of the node directly. Also the algorithm stores the dynamic control flow diagrams as paths that are explored in the queue after the program code paths are completed [17].

To display dynamic multimodal sensory data received from the test creation program. The proposed method focuses on the generation and creation of tests for a specific node x_i, by combining with the corresponding dynamic multimodal communication sensor data. Algorithm 2, which describes a specific process [19], namely the test case generation function *makeTestCases* takes the protocol fields, i.e., the element in the set *DIF*(x_i) in definition 1, the rule constraint *c*, and the protocol *G* s input, and the output is the test case set *tcList*. The key to the function is to generate test cases by obtaining the efficient grammar of each node and the reverse grammar Algorithm 2 start with line which contains only one valid grammar for the protocol fields, i.e. outputs the grammar of each protocol field in the constraint condition according to the parameters as the valid grammar of the node.

A possibility is that the valid logic applied to node x_i is definitely a subset of the protocol grammar G, because by Definition 3.2, i.e. under *Constraint* (x_i), he valid logic can be applied to all protocol fields in $DIF(x_i)$. For example, for node x_i protocol field contains many correct grammars, constructing the many grammars using combinations of correct grammars, and storing the grammar in a set of test cases. For example, for node $DIF(x_i) = \{F_a,F_b\}$, and the actual logic derived in C*onstrain*t (x_i), є F_a = (0|1), F_b = 2, and then the aggregate logic is (F_a = 0, F_b = 2) i (F_a =1, F_b=2). Next, algorithm change the correct logic corresponding to each of the fields of the valid grammar in the premise of fulfilling the constraint condition *Constraint* (x_i), and then combine them with the correct grammar of the other fields for the fuzzy grammar and add the grammar to the set. test cases. If no matching test grammar is found in $DIF(x_i)$ the register data must be generated by changing methods such as random bit shifting, extreme swapping, and extreme value swapping.

Methods and experiment

In this work, the Modbus protocol is taken as an example. Modbus protocol fields include identification, information length, function code field, start address, end address. The data field is defined according to the function code and CRC (Cyclic Redundancy Check). Value of CRC calculation and other checksum algorithms as protocol termination mechanisms are usually built into the protocol specification to detect potential corrupted data. Although the Modbus/TCP validation mechanism is implemented in the TCP transmission frame and protocol formats do not include CRC, CRC validation typically exists in Modbus RTU and other industrial Internet protocols. If the CRC obtained by the program does not match the CRC obtained by calculation, the corresponding case data can be ignored directly. This is a very important mechanism for security and functionality, but if the CRC value is not updated when the protocol fields change, the fuzz test should be blocked, so the CRC is also considered an important field.

Figure 4 shows the control block diagram and the ways to perform the first login. This is an adynamic control block diagram constructed with a valid input performing a write function, the right half of which shows the paths actually taken by the input during the execution of the application program, and the dashed part shows the new execution paths discovered at each node. According to Definition 3, in the figure, the start node 3_1 represents the first execution of the conditional branch, the elements in parentheses represent $DIF(3_1)$, the field that affects the conditional branch; the side with the arrow represents the dependency and control of the $CDC(3_1)$ relation.

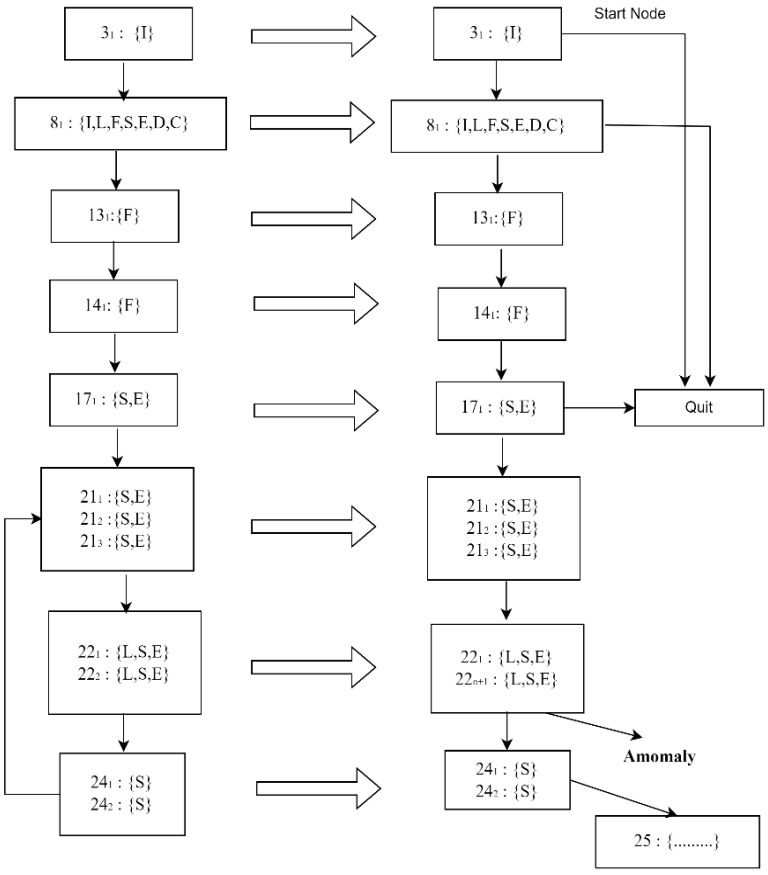

Fig. 4. Block diagram of management and methods of implementation of the first case

The control block diagram shown in Figure 5 is a test case generated by selecting a combination of function code 0x05 and n data fields in case node 24_1 that is true during the execution of the first input and is treated as the second input.

Assume that the initial node 3_1 is independent of any node and considers the protocol identification field *I* as a dynamic interactive field and has a single true value of *0x0000*. Then the corresponding two test grammars are obtained after fuzzification with the algorithm. Node 3_1 is actually a conditional branch to determine whether the field is true or false according to the protocol, so according to each test grammar it is easy to conclude that the constraint condition that makes node3_1 true is $I \neq 0x0000$.

The condition that makes the conditional branch executable is $I \neq$ 0x0000, while the constraint that makes the conditional branch false is $I = 0x0000$. In this case, for the next node 8_1 according to its $CDC(8_1)$, it is necessary to output *Constraint* (8_1), which is the constraint that makes the conditional branch 3_1 false, i.e $I =$ 0x0000. Given cyclic redundancy as a single field grammar check applied to the entire field generates two test grammars $I = 0x000 \wedge CRC(I, L, F, S, E, D) = C$ and $I = 0x000 \wedge CRC(I, L, F, S, E, D) \neq C$ a in a similar way. Based on this, we obtain the constraint condition of node 8_1, and then similarly solve each node.

It should be noted that according to the description of the algorithm, the test logic generated at node 13_1, decides whether node 14_1 is true or false. Therefore, there is no need for the test logic generated at node 14_1, Similarly, skip nodes 21_1, 21_2, 22_2, 24_2 i 21_3. According to Algorithm 1, when the test logic is generated, the dynamic block diagram is stored in the queue as the examined path. In this case, if there is a researched path, when searching for paths in the control flow diagram, we can get a partition node that generates different paths by comparing the paths and considering the node as the start node. All nodes before this node have the same dynamic multimodal sensor communication data and thus can be extracted directly to avoid re-generating test cases. Table 1 shows an example of the grammar of the generation of the test example of the first entry, obtained using Algorithm 2.

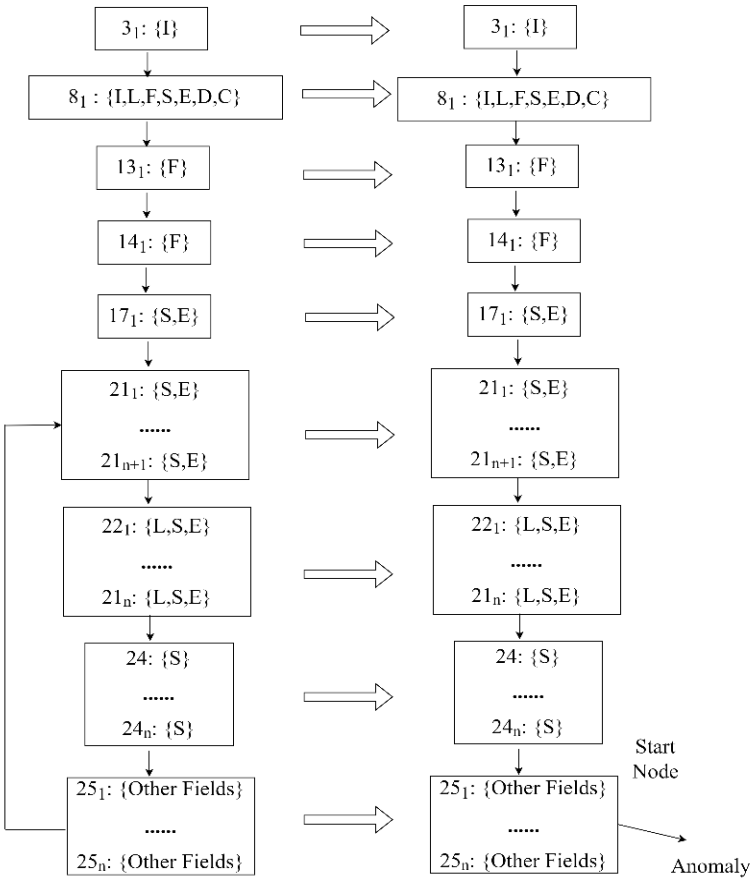

Fig. 5. Block diagram of control and methods of implementation of the second case

An experiment on the example of the most popular Modbus/TCP protocol. The experiment was carried out with Ubuntu 14.04, so we chose the open source library "*libmodbus*" to implement Modbus. Communication of the TCP protocol in the Linux system. Implemented the proposed fuzzy method with Symfuzz, an open source binary analysis tool based on BAP [19] that can convert executables into an intermediate language applicable to program analysis, and combined with PIN to executing the dynamic binary tools against the target application to obtain the dynamic interactive fields and dependency and control relationships required by the method to generate the next test case. We choose AFL-fuzzer as a fuzzy tool to compare with experimental results.

A Modbus slave station waits for request messages from other master stations. In the experiment, we found that the master station sends 25, 30, and 35 request messages as test input to the slave station, respectively, and performs fuzzy processing to the Modbus slave program and checks the whole program for any abnormalities.

To evaluate the effectiveness of the proposed fuzz-testing method, we compared the number of test cases, the total number of execution scenarios, the speed of code coverage, and the testing time with the same number of samples.

The experiment is based on the statistics of the number of test cases created using two methods of fuzzy testing with 25, 30, and 35 data samples. The number of test cases refers to the total number of samples generated after the program crashes or the examples run completely.

Table 1

Conditions for generating test cases

Node x_i	DIF(x_i)	CDC (x_i)	Limitation (x_i)	Valid logic	Logic after fuzzification	Logic test
3_1	*I*	*Ø*	*Ø*	$I = 0x0000$	$I \neq 0x0000$	$I = 0x0000$ $I \neq 0x0000$
8_1	*I~C*	(3_1, *F*)	$I = 0x0000$	$CRC\ (I \sim D) = C$	$CRC\ (I \sim D) \neq C$	$CRC\ (I \sim D) = C$ $CRC\ (I \sim D) \neq C$
13_1	*F*	(8_1, *F*)	$I = 0x0000$ $CRC\ (I \sim D) = C$	$F = 0\ x\ 01$	$F \neq 0\ x\ 01$	$F = 0\ x\ 01$ $F \neq 0\ x\ 01$
14_1	*F*	(13_1, *F*)	$I = 0x0000$ $CRC\ (I \sim D) = C$ $F \neq 0\ x\ 01$	*Skip*	*Skip*	*Skip*
17_1	*S, E*	(14_1, *T*)	$I = 0x0000$ $CRC\ (I \sim D) = C$ $F = 0\ x\ 05$	$C \leq E$	$C > E$	$S \leq E\ C > E$
21_1	*S, E*	(17_1, *F*)	$I = 0x0000$ $CRC\ (I \sim D) = C$ $F = 0\ x\ 05$ $C \leq E$	*Skip*	*Skip*	*Skip*
21_1	*L, S, E*	(21_1, *T*)	$I = 0x0000$ $CRC\ (I \sim D) = C$ $F3 = \{1\}, F4 \leq F5$	*Incoming data. The size is valid*	*Incoming data. The size is valid*	*Incoming data. The size is valid*

From Figure 6, it can be seen that with a larger number of samples, the test data obtained using the proposed method is significantly smaller than the test data generated using the AFL-fuzzer, because the proposed method constructs test cases using technology based on data generation. In this case, the generated test data is smaller than that of the AFL-fuzzer using the variation strategy.

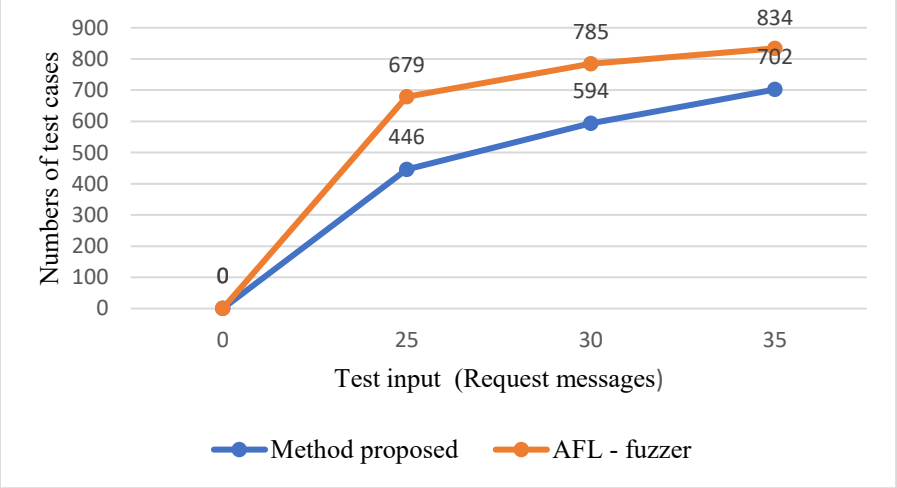

Fig. 6. Number of tests with different number of samples

Furthermore, we can see that the number of test cases generated by AFL-fuzzer increases steadily with the number of samples, because Afl-fuzzer reduces the input samples given that users may offer low-quality initial samples and thus cause a possible data redundancy in some types of variation. Although the proposed method depends more significantly on the number of samples, it uses a more appropriate generation strategy [19,20].

Figure 7 shows the code coverage rate. The principle of the calculation is that the devices can help fix the rate of coverage of the branch and detect gross damage by calculating the performance of functions.

The speed of code coverage is not necessarily related to the probability of finding anomalies, but surely for a test case an execution path that does not reach a conditional deep-level program branch will not cause any potential anomalies.

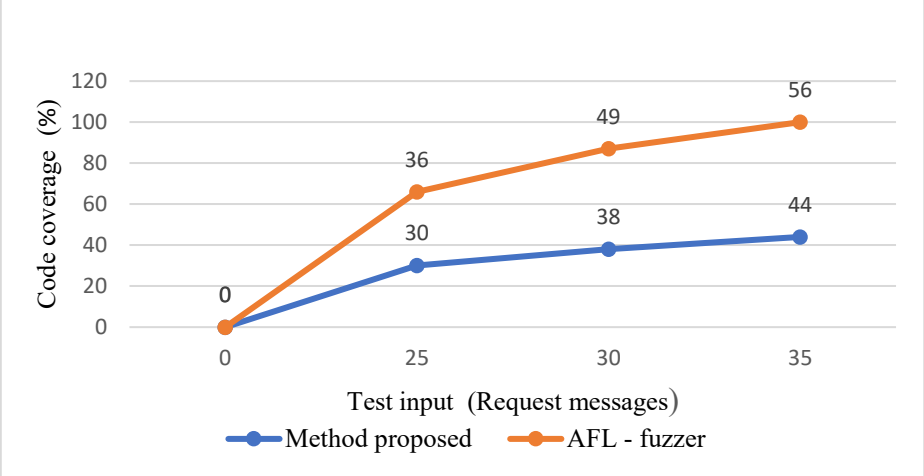

Fig. 7. Coefficients of code coverage with different number of samples

So, we can see that the proposed method realizes a higher code coverage rate compared to AFL-fuzzer. Table 2 shows the statistics when the target program crashes.

Generating a test case by combining with software dynamic multimodal communication sensor data pays the price of test time to solve the conditional constraint of branching and test grammar generation compared to the variation strategy. However, this greatly increases the number of execution paths and the speed of code coverage, indicating that the more execution paths found by conditional branches during program execution, the more likely it is that test cases will reach a deep level and trigger anomalies. , and the relevance is stronger. Thus, in case the program has anomalies, although it shortens some testing time, the proposed method is superior to AFL-fuzzer in terms of Modbus-TCP protocol test implementation.

Table 2

Comparison of experimental results

Test method	**Test number**	**The number of completed paths**	**Code coverage**	**Test time/hour**	**Anomaly**
Proposed Method	18492	3862	55,90%	4.19	1
AFL – fuzzer	46931	1787	39,85%	3.35	1

Conclusions

1. The article proposes a fuzzy processing method in combination with dynamic multimodal sensor data at the application system level in the Industrial Internet for application in multisensor systems. The proposed method traces program execution, finds input fields affecting conditional branches through dynamic patch analysis, and captures conditional branch dependencies to properly control the generation of test case grammars, thereby increasing code execution capability at a deep level. The results of the comparative experiment confirm that the method to some extent improves the validity of test cases and the speed of code coverage, and also increases the probability of detecting anomalies in the implementation of the protocol. The proposed method to some extent improves the validity and code coverage of test cases and significantly increases the probability of detecting anomalies in the implementation of the IoT protocol

2. The method proposed in the paper is a dynamic and targeted test from the point of view of the program being tested. Dynamic analysis of the program is also expensive. So, in the next step, we consider combining the technology of static analysis. The program aims to reduce the load on dynamic analysis and at the same time use the obtained dynamic information to adjust the blur position and strategy combination.

3. The industrial control protocol model describes only the data format level. In real production, it is necessary to take into account that the protocol has an execution state. Therefore, the description of the industrial state control protocol needs detailed study to establish the relationship between the protocol format and the protocol state transition.

References

1. Q. Li, Y. Tian, Q. Wu, Q. Cao, H. Shen, H. Long, A Cloud-Fog-Edge closed-loop feedback security risk prediction method. IEEE Access 8(1), 2020. P. 29004–29020.
2. Ericd, Knapp et al., Industrial Network Security: Smart Power Grids, SCADA and other IIS Key Infrastructures, CA: NDIP. 2014. P.18-26.
3. Qianmu Li, Shunmei Meng, Shuo Wang, Jing Zhang and Jun Hou. CAD: command-level anomaly detection for vehicle-road collaborative charging network. IEEE Access, Vo.7. 2019. P. 34910–34924.
4. S. Raval, Black Energy a threat to Industrial Control Systems network security, International Journal of Advance Research in Engineering. Sci Technol 2, 2015. P. 31–34.
5. IIS-CERT. Information products [EB/OL], https://IIS-cert.us-cert.gov/, 2018.

6. China National Vulnerability Database. Vulnerability of Industrial Internet Industry [EB/OL], http://IIS.cnvd.org.cn/, 2018.
7. S Wan, M Li, G Liu, C Wang, Recent advances in consensus protocols for blockchain: a survey. Wireless Networks, 2019. P. 1-15.
8. J.M. Garibaldi, The need for fuzzy AI, IEEE/CAA J. Autom. Sinica 6(3), 2019. P. 610–622.
9. Cui Baojiang, Zhang Xiangqian, Zhang Tianxin, Zhang Qin. Embedded system vulnerability mining technology based on in-memory fuzzing test. The 13th International Conference on Broadband, Wireless Computing, Communication and Applications. Taichung, Taiwan. pp.439-449. https://doi.org/10.1007/978-3-319-69811-3_40. October 27-29, 2018. P.439-449.
10. Aitel D., An introduction to SPIKE, the fuzzer creation kit [EB/OL], http://www.blackhat.com/presentations/bh-usa-02/bh-us-02-aitel-spike.ppt, Accessed, 2014-01-06.
11. Devarajan G.Unraveling, SCADA protocols: using Sulleyfuzzer [EB/OL], http://www.defcon.org/html/defcon-15/dc15 speakers.html, (21.06.2015).
12. Peach. [EB/OL]. http://www.peachFuzzer.com, (21.07.2015).
13. Zalewski, American fuzzy lop [EB/OL]. http://lcamtuf.coredump.cx/afl/, 2017-12-25.
14. S. Gan, C. Zhang, X. Qin, et al., CollAFL: path sensitive fuzzing, in 2018 IEEE Symposium on Security and Privacy (SP) (IEEE Computer Society, San Fransisco, CA, USA), 2018. P. 660–677.
15. L. Chen, S. Liu, D. Xiao, et al., A Cisco IOS heuristic fuzz test method. Comput Eng 40, 2014. P. 68–73.
16. Q. Li, Y. Wang, P. Ziyuan, S. Wang, W. Zhang, A time series association state analysis method in Smart Internet of electric vehicle charging network attack. Transport Res Record 2673, 2019. P. 217–228.
17. S.K. Cha, M. Woo, D. Brumley, Program-adaptive mutational fuzzing, in 2015 IEEE Symposium on Security and Privacy, San Jose, CA, 2015.
18. Hanwen Liu, Huaizhen Kou, Chao Yan, Lianyong Qi. Link prediction in paper citation network to construct paper correlated graph. EURASIP Journal on Wireless Communications and Networking, 2019.DOI: https://doi.org/10.1186/ s13638-019-1561-7.
19. Wan, S., Li, X., Xue, Y. et al. Efficient computation offloading for Internet of vehicles in edge computing-assisted 5G networks. J Supercomput, 2019. https://doi.org/https://doi.org/10.1007/s11227-019-03011-4
20. Lianyong Qi, Xuyun Zhang, Shancang Li, Shaohua Wan, Yiping Wen, Wenwen Gong. Spatial-temporal data-driven service recommendation with privacy-preservation. Information Sciences, 2019. DOI: https://doi.org/10.1016/j.ins.2019.11.021.

Halyna Klym **Галина Клим**	Doctor of Technical Sciences, Professor of the Department of Sspecialized Computer Systems, Professor of the Lviv Polytechnic National University, Lviv, Ukraine, e-mail: halyna.i.klym@lpnu.ua, orcid.org/0000-0001-9927-0649	Доктор технічних наук, професор, професор кафедри спеціалізованих комп'ютерних системи, Національного університету «Львівська політехніка», Львів, Україна.
Roman Diachok **Роман Дячок**	Graduate student of the department of specialized computer systems, Professor of the Lviv Polytechnic National University, Lviv, Ukrain , e-mail: rodyachok@gmail.com. https://orcid.org/0000-0002-8652-140X	Аспірант кафедри спеціалізованих комп'ютерних системи, Національного університету «Львівська політехніка», Львів, Україна.