IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON URBAN COMPUTING AND INTELLIGENCE

Received February 16, 2019, accepted March 4, 2019, date of publication March 14, 2019, date of current version April 1, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2904047

CAD: Command-Level Anomaly Detection for Vehicle-Road Collaborative Charging Network

QIANMU LI[1,2], SHUNMEI MENG[2,3], SHUO WANG[1,2], JING ZHANG[2], AND JUN HOU[1,2,4]

[1]Intelligent Manufacturing Department, Wuyi University, Jiangmen 529020, China
[2]School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China
[3]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[4]Zijin College, Nanjing University of Science and Technology, Nanjing 210094, China

Corresponding author: Jun Hou (houjunnjust@163.com)

This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFE0108000, in part by the Fundamental Research Funds for the Central Universities under Grant 30918012204, in part by the Jiangsu Province Key Research and Development Program under Grant BE2017739, in part by the 2018 Jiangsu Province Major Technical Research Project "Information Security Simulation System"(electric power and energy) through the National Science Foundation of China under Grant 61702264, Grant 61872219, and Grant 91846104, and in part by the Open Research Project of State Key Laboratory of Novel Software Technology (Nanjing University) under Grant KFKT2017B07.

ABSTRACT A large number of charging piles installed on roadside parking spaces and smart poles on the roadside of the Internet have become essential substation infrastructure (roadside) for building a vehicle-road coordinated charging network for electric vehicles. The management system of China National Grid's network-load interaction includes the interaction between these main stations (traffic control stations) and substations (roadsides). The Internet-friendly interactive communication protocol for China's vehicle-road coordination is IEC 60870-5-104 (104 protocol). The control network of the vehicle-road collaborative charging network has many characteristics, such as multiple levels, multiple types, and frequent information exchange for monitoring and control. Various types of operational information and control commands are subject to eavesdropping, tampering, and interruption during collection, transmission, and triggering. This paper proposes a command-level anomaly detection (CAD) method for a vehicle-road collaborative charging network. The CAD method analyzes the protocol for the specification format and business command characteristics of the 104 protocol. This paper uses the dynamic analysis protocol fuzzy test to realize the dynamic information in the program to guide the generation of test cases and pass the Markov state transition diagram. We describe the state transition and abnormality identification of protocol messages. We also design a long-term memory network to implement instruction-level anomaly feature mining. The experiment proved the validity of CAD. If we adopt other new protocols for the vehicle-road coordinated network in different countries or regions, the analysis of the new protocol can be completed in the same way, which has strong application value and prospect.

INDEX TERMS Protocol analysis, command-level anomaly detection, vehicle-road collaborative charging network.

I. INTRODUCTION

As electric vehicles take the lead in realizing the connectivity of the vehicle network control and the vehicle road coordination system, they become the best carrier for intelligent network connection and automatic driving. In 2016, Norway, Sweden, and other countries announced that they would stop selling fuel vehicles in 2030. The United States, Japan, India, and other countries have announced a road map of electrification and intelligence. The 8th Clean Energy Ministerial Conference in 2017 proposed the "EV30@30 Target Initiative", which means "30% of cars should be new energy vehicles by 2030". Electrification and intelligence are leading the new direction of the development of the global automotive industry.

At present, the State Grid of China has built a fast electric vehicles network with 240,000 electric vehicle charging stations, covering more than 150 cities. The communication

The associate editor coordinating the review of this manuscript and approving it for publication was Liang Hu.

protocol of network-load Interactive in vehicle-road collaborative charging network is IEC 60870-5-104 (104 protocol). Although the 104 protocol has solved the problem of monitoring data transmission of the interactive terminal, it does not propose a practical solution to the security description and the abnormal identification, which seriously threatens the security of the transportation infrastructure. At present, from the charging equipment, communication system to information service, there are different forms of industrial control terminals for data acquisition, command scheduling, remote control and so on. The 104 protocol achieves the flow of control flow and data flow between these industrial control terminals and the primary station. In the implementation process of the industrial control terminal code pair 104 protocol, the protocol field is trusted, but when the attacker modifies the data values of these fields by using the protocol defect, the program can be controlled to run. For example, the destination address parameter of the jump instruction usually comes from a trusted data source inside the program. However, if the attacker rewrites the destination address of the jump instruction, it can illegally control the operation process of the industrial control system. The mainstream solution to such problems is the protocol format analysis [1]–[5]. However, the traditional protocol parsing method is complicated to be universal, and it is difficult to construct a context-free grammar representation of the protocol. Also, the traditional parsing process is mostly based on stack, script-based, or compiler-based reasoning, so the resolution speed is limited. Therefore, it is urgent to research the abnormal feature detection method of instruction level.

Most of the Intelligent Vehicle-Infrastructure Cooperative Systems and V2X have a similar problem. These interactive control systems are complex nonlinear systems that are subject to a variety of external factors. The macro-instruction behavior of interactive control is often complex and changeable. The data contains a variety of periodic fluctuations, and there are nonlinear rising and falling trends, and uncertain random factors also interfere it. Therefore, how to select and optimize the nonlinear model has become the research focus of the analysis of the instruction-level anomaly characteristics in the interactive control system in recent years [6]– [9].

The Long Short Term Memory (LSTM) is mainly used to describe complex and interactive systems for abnormal state analysis. It has been used in large-scale systems such as Real-Time systems, electronic circuits, and communication networks [10]– [16]. This paper proposes a command-level anomaly detection method (CAD) for vehicle-road collaborative charging network. CAD firstly analyzes the protocol according to the specification format and business instruction characteristics of the communication protocol. So that we can implement the program execution through the tracking protocol, find the input field that affects the execution of the conditional branch through dynamic pollution analysis and capture the dependency relationship between the conditional branches. In this way, the use case generation is guided to achieve the instruction level feature extraction. Secondly, the Markov time-varying model (MTM) is used to design the state transition diagram to describe the protocol message state transition and abnormality recognition. Finally, through the training and learning of Long Short Term Memory Networks (LSTM), we realize the discrimination of command anomalies in the network-load interaction control of the vehicle-road collaborative charging network.

II. RELATED RESEARCH

The instruction exception of the network control system mainly comes from the instruction vulnerability. In the field of command vulnerability detection, there have been analysis techniques such as dynamic analysis, symbolic execution and fuzzy testing [6], [17]. Compared with other technologies, fuzzy testing requires only a small amount of target knowledge and is easy to expand and reusable. Therefore, it has become the most popular instruction vulnerability discriminating scheme. The representative fuzzy is SPIKE [18]. The principle of SPIKE is to describe the protocol as a block sequence model, by dividing the data structure of the message and automatically counting the length of the field after the mutation. SPIKE improves the validity of test cases, but the ability to describe constraint relationships in protocol messages is not sufficient. Sulley [19] and Peach [20] extended the data model based on SPIKE and added more descriptions of the dependencies between data blocks. In order to provide a more flexible and accurate fuzzy framework, AFL [21] tracks the path coverage of each input by lightweight instrumentation of the source program and uses a hash mechanism to assign IDs to the basic blocks on the path randomly. Thus we can judge whether the input generates a new path and use it as a seed. This hashing method is prone to collisions, and there is a problem that the input arrives at the new path, but it is not reported. The CollAFL proposed by Gan *et al.* [22] assigns ID values to each basic block employing greedy algorithms to ensure that the hash values of each side are different, avoiding hash collisions, and making path coverage judgment more accurate. Moreover, they use the number of path neighbor branches as weights to sort the seeds, leading the fuzzy to explore paths that have not arrived. Rawat *et al.* [23] proposed VUzzer without prior knowledge of any application or input format. They mainly use static and dynamic analysis of control flow and data flow features to aid mutation and selection of seed files. In this progress, they assign each basic block specific weights to drive the program to a deeper level. However, these methods are difficult to apply to the 104 protocol with complex input syntax.

III. COMMAND-LEVEL ANOMALY DETECTION METHOD

The flow of the command-level anomaly detection method designed in this paper is shown in Figure 1. The method mainly includes some modules, such as packet capture, packet depth analysis, MTM-based protocol feature modeling, protocol feature tracking and identification based on LSTM, and instruction extraction of 104 protocol. The packet capture module provides necessary input for

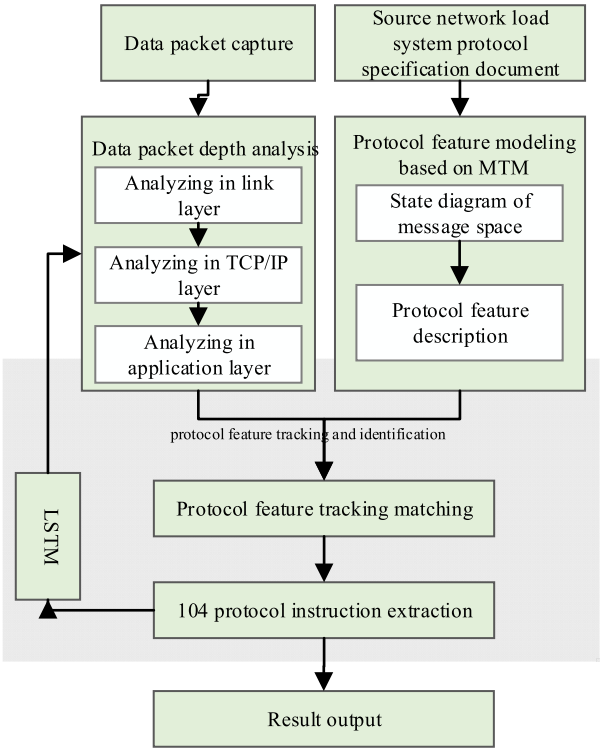

FIGURE 1. Command-level anomaly detection method of vehicle-road collaborative charging networks.

subsequent modules. The packet depth analysis module performs expert analysis of the link layer, the network layer and the transport layer of the data packet. The MTM-based protocol feature modeling implements the construction of the spatial state diagram of the protocol message features and the protocol feature description. The protocol feature tracking and identification module perform instruction level analysis of the application layer data stream according to the protocol feature model. The LSTM network realizes the feature processing of the measured object through self-learning.

A. DEEP ANALYSIS OF DATA PACKETS

The steps for deep packet parsing are designed as follows:

STEP (a) Protocol segmentation. Further subdividing the ''detection flow'' by the session. For example, one interaction of 104, one user login behavior of FTP, and one mail transmission/reception of SMTP/POP3, which can be abstracted into one ''detection flow'';

STEP (b) Protocol domain segmentation. The message is segmented at the smallest granularity, dividing the detection stream into the Header and Body parts, and the Header is subdivided into individual Fields. Protocol domain segmentation helps to determine whether the header field needs to be detected. It can determine whether the feature of the header field hits its definition, and identify the key location for extracting audit log information;

STEP (c) Analyze the interaction process of remote signaling, telemetry, and remote control. Decode the header of the protocol, and send the decoded field to the fuzzy test algorithm engine.

STEP c.1. After receiving the 104 message, parse the message, obtain the frame header by looking for 68H, and then obtain the frame length.

STEP c.2. Determine which type of information frame the message is based on the type flag.

For example, a network frame of data is received as 68 14 2E 00 04 00 1E 01 03 00 01 00 0B 00 01 31 25 16 15 12 04 0B. It is a remote burst frame, its data length is 20, the type identifier is 30, and the single point information with time stamp CP56Time2a. 0B 00 01 indicates that the information object address is 65547. 31 is the code value description (bit IV is 0 for valid, bit NT is 0 for current value, bit SB is 1 for replaced, bit BL A value of 1 indicates that it is blocked, and a bit PI of 1 indicates that the switch position is ON, that is, the switch is closed.)

Another example, the telemetry frame 68 10 2C 00 04 00 09 01 03 00 01 00 01 40 03 3E 00 00 is received. The type identification is 09 for normalized measurement. 01 40 03 for information object address is 212993. 3E 00 indicates that the telemetry value is 62, where 00 is the code value description (bit IV is 0 means valid, bit NT is 0 means the current value, bit SB is 0, the table is not replaced, bit BL is 0 means unblocked, bit OV is 0 means no overflow).

STEP c.3. Analyze the data according to the frame structure to get relevant information. The values of the obtained remote signal, telemetry, remote control, and remote adjustment are assigned corresponding numbers and placed in the memory.

STEP (d) Negotiation protocol identification. Negotiation is used for data transmission. The corresponding protocol parser can identify the quintuple features inherent in the negotiation protocol through the parsing of session packets, and identify the session by negotiating the matching of the association table.

In the processing phase of ''deep analysis of data packets,'' the fuzzy test algorithm engine discovers potential vulnerabilities by injecting a large amount of malformed data into the target system and monitoring the execution state of the system [25-38]. The advantage of the fuzzy test algorithm engine is that it is easy to deploy, and is not limited to the internal implementation details and complexity of the system under test. It can be executed with or without source code and has good scalability and practicability. In CAD, the processing of the fuzzy test algorithm engine is shown in Figure 2. For the industrial control protocol implementation program, dynamic information is obtained through dynamic stain analysis to guide the generation of test cases. In order to avoid a situation where the number of test cases is insufficient due to single sample data, the method uses multiple protocol messages as input.

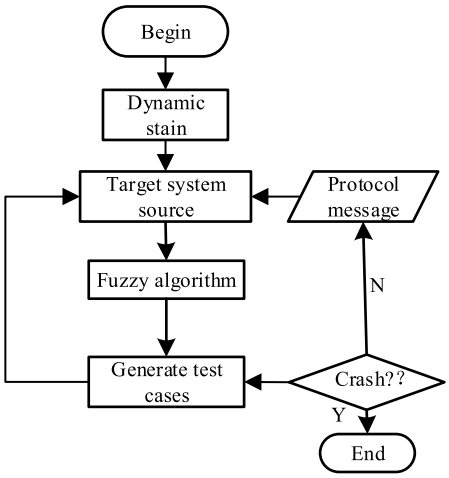

FIGURE 2. 104 protocol fuzzy test flow based on dynamic analysis.

For a conditional branch in the control program, its dynamic information contains two parts: dynamic interaction field and dependency control relationship.

Definition 1 (Dynamic Interaction Field) In the given program execution, for conditional branchx_i (the i execution of conditional branch x), there is

$DIF(x_i)$={Fj|Fj is the protocol field that affects x_i execution.} Where $DIF(x_i)$is the set of protocol fields.

Definition 2 (Dependency Control Relationship) In a given program execution, if there is a true or false conditional branch y_i directly determines whether x_i is executed, then x_i is said to be dependent on the dynamic control of y_i, denoted as $CDC(x_i) = \{y_j, T|F\}$, where $CDC(x_i) = \{y_j, T|F\}$ is the set of qualified branches, and $Constraint(x_i)$ is the conditional branch x_i. Constraint.

Definition 3 (Dynamic Control Flow Graph) For each program input, its execution path can be represented by a dynamic control flow graph, where conditional branch x_i is a node, $DIF(x_i)$ is a node dynamic interaction field, and $CDC(x_i)$ is an edge, indicating a dependency relationship between conditional branches.

CAD uses dynamic stain analysis to determine which input protocol fields affect the conditional branch, pollute the token for each protocol input field, and track the contaminated data stream during program execution. Since the checksum (such as CRC) is common in the 104 protocol field, if the propagation in the control information flow leads to the flood of pollution data, when using the dynamic stain analysis method, we only pay attention to the propagation pollution in the data stream. Do not track pollution in the control flow. Algorithm 1 is the specific description.

In Algorithm 1, the main function dynamicFuzz takes the executable program P, the protocol syntax G, and the protocol message I as input, processes the multiple protocol inputs, and outputs the triggered exception information. Lines 1∼2 of Algorithm 1 initialize the data structure used for storage. Since the generated test case is also used as new input, it is placed in a queue for recursive execution.

Algorithm 1 FUZZY TEST algorithm

Input: Program P, protocol syntax G, protocol message I
Output: Exception information
Function dynamicFuzz (P, G, I)

```
1.  probedPath, inputQueue, checklist ← empty()
2.  inputQueue.push(I)
3.   while inputQueue.notEmpty() do
4.      input ← inputQueue.pop()
5.      dcfg ← executionAnalysis (P, I)
6.      node ← dcfg.start(probedPath)
7.      while node ≠ NULL do
8.         c ← dcfg.getConstraint (node.CDC)
9.         if checklist.find(node.DIF, c) &&
10.         node.true, node.false ≠ NULL then
11.         node ← dcfg.next()
12.         continue
13.        end
14.        tcList ← makeTestCases (node.DIF, c, G)
15.        for each tc in tcList do
16.           input'← makeInput (input, tc.value, c)
17.           res ← executionMonitor(P, input')
18.           if (res.execption) then
19.             return getExceptionInfo(res)
20.           end
21.           if(probedPath.isNew(res.pathInfo)) then
22.             inputQueue.push(input')
23.           end
24.        end
25.        checklist.add(node.DIF, c)
26.        node ← dcfg.next()
27.     end
28. end
29. probedPath.add(dcfg)
```

Line 5 combines the two parameters of the program and protocol, and uses the dynamic stain analysis method to find the $DIF(x_i)$ and $CDC(x_i)$ sets of each conditional branch in each program, and constructs the corresponding dynamic control flow with $DIF(x_i)$ as the node and $CDC(x_i)$ as the edge. Line 6 begins with the blurring operation from the start of the executable path in the control flow graph. Line 8 infers the constraint $Constraint(x_i)$ corresponding to the node based on the dependency control relationship as a parameter of the test case generation algorithm. Line 16 of Algorithm 1 indicates that when the test case is generated, we need to use it as new input, replacing it with the value of the protocol field in the $DIF(x_i)$ set in the original input, and modifying all the fields associated with $Constraint(x_i)$ to meet the constraints. In order to ensure that the test cases can be executed to a deeper level in the program, the remaining protocol fields that are not related to $DIF(x_i)$ and $Constraint(x_i)$ should be valid according to the protocol syntax. For example, the values of the two fields of the start and end addresses in the protocol input are 1 and 2 become 3 and 6, respectively. Even if there

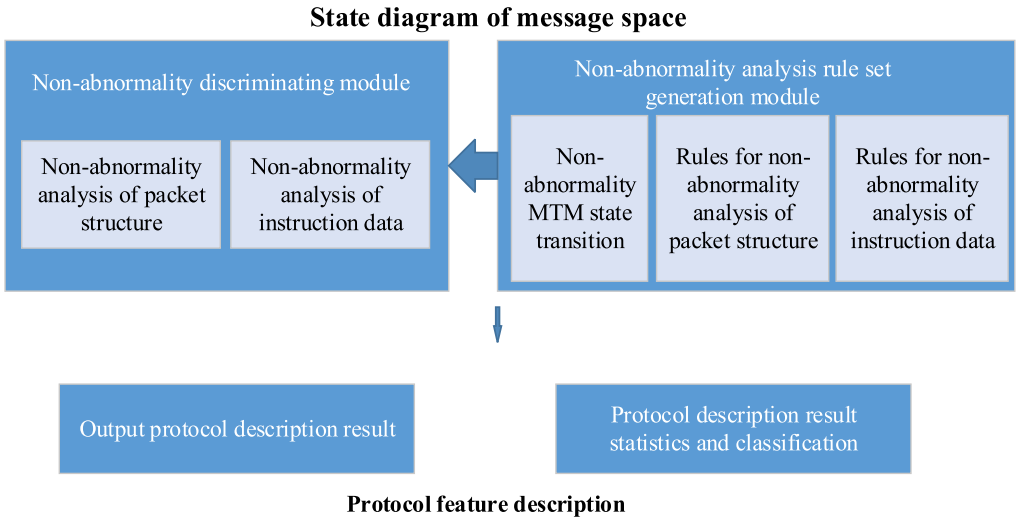

FIGURE 3. Flow of MTM-based protocol feature modeling.

are no constraints, the corresponding address data field will be regenerated according to the original standard size of its protocol (increasing from 1 to 3). But if the two addresses are blurred after the invalid test case of 6 and 3 (start address is less than the end address), the fields unrelated to $DIF(x_i)$ and $Constraint(x_i)$ will remain at reasonable values.

Lines 17-20 of Algorithm 1 monitor for abnormalities such as crashes or memory leaks during the execution of the test case. Although the test case is used as a new input, the method does not repeatedly acquire the dynamic information of each conditional branch, considering the overhead incurred by program execution. Lines 21-24 store the traversed code path during the execution of the test case. If a conditional branch that has not been resolved is encountered, it will be placed in the input queue, and the number of new branches determines the priority. Line 25 saves the dynamic information sequence of each node, which facilitates the judgment of whether the current node generates test cases in lines 9∼13. When a dynamic information sequence already exists in the list and test cases are generated and correspond to $DIF(x_i)$ and $Constraint(x_i)$ of the current node x_i, then the test case generation of the node can be skipped directly. Line 29 is to store the dynamic control flow graph as a probed path in the queue after the program code path has been completed.

In order to be able to directly embody the dynamic information extracted from the program in the construction of the test case, the method selects the generated technology as the main, the mutation as the supplementary. The test case syntax is generated for the node x_i in combination with its corresponding dynamic information. Algorithm 2 describes the specific process, the test case generation function makeTestCases, with the protocol field *fields*, that is, the element of the $DIF(x_i)$ set in the definition 1, the rule constraint C and the protocol syntax G are the input, and the output test case set *tcList*. The key to the function is to generate test cases by getting the valid syntax of each node and the opposite syntax. According to Definition 2, the valid syntax under $Constraint(x_i)$ can be applied to each protocol field in $DIF(x_i)$. Line 3 indicates that when there are multiple valid grammars in the protocol field, multiple grammars are stored in the test case set by the combination of them. For example, for node x_i, there is $DIF(x_i) = \{F_a, F_b\}$, and the correct grammar derived at $Constraint(x_i)$ is $F_a = (0|1), F_b = 2$, then the combined syntax is $(F_a = 0, F_b = 2)$ and $(F_a = 1, F_b = 2)$. Then, in lines 4∼9 of the algorithm, for each field in the correct grammar, the corresponding valid grammar is negated under the condition that the $Constraint(x_i)$ constraint is satisfied, and the grammar of the other grammar is added to the test case. In the collection.

When the applicable test syntax is not found in $DIF(x_i)$, the use case data is generated by mutation, such as using random bit flips, replacing with extreme values or boundary values.

Algorithm 2 Test Case Generation Algorithm

Input: protocol field *fields*, rule constraint c, protocol syntax G
Output: test case set *tcList*
Function makeTestCases(fields, c, G)

```
1) tcList ← empty()
2) validGrams ← extractGrams(fields, c, G)
3) tcList ← combination (validGrams) ^c
4) for each g in validGrams do
5)    fg ←~g ^c
6)   fuzzyGram ← fg ^(other g'in validGrams)
7) tcList.add(fuzzyGram);
8) end
9) return tcList
```

B. MTM-BASED PROTOCOL FEATURE MODELING

The MTM-based protocol feature model is to track and analyze the instruction frame of the historical protocol message and match with the rules in the rule set to judge the

non-abnormality of the instruction. It realizes the initial screening of the instruction abnormality, and improve interaction security of the control master station and Roadside substation unit. Interactive commands are mostly used to change the state of running terminal devices, such as remote control and remote adjustment. There are two types of instruction transfer: direct instructions, select and execute instructions.

The flow of MTM-based protocol feature modeling is shown in Figure 3. The non-abnormality analysis of protocol packets is mainly divided into the message space state diagram and the protocol feature description module. The framework of the exception state of message space is mainly divided into two parts: the abnormality recognition rule set generation module and the non-anomaly discrimination module. The non-abnormality analysis rule set generation module analyzes the interaction process of the grid-load unit communication protocol message. It uses the Markov time-varying theory to model the interaction process. Thereby it generates a non-abnormality analysis rule set. The non-abnormality discrimination module performs matching in the non-abnormality analysis rule set according to the data packet provided by the data acquisition layer. In this way, the non-abnormality of the protocol message interaction process is discriminated, to analyze the non-abnormality of the grid-load unit protocol message. The protocol feature description module provides unified visual non-abnormality real-time analysis results display, non-abnormality overall situation display, and other functions.

In the abnormal state diagram of the message space, the exception detection rule set generation module is the core component. It plays a role in the overall data flow of the model. The message space anomaly state diagram can be described by the Markov time-varying model as shown below:

$$M = (S, s_0, V, I, P, A, T) \tag{1}$$

where, $S = \{s_0, s_1, \ldots, s_n\}$: A non-empty finite state set, the set describes various steps experienced by the security protocol message in performing the non-abnormality analysis process.

S_0: Initial state, describing the starting state of the non-abnormality analysis process.

$V = \{v_1, v_2, \ldots, v_n\}$: A set of variables that describe variables such as data fields or counters that need to be provided during non- abnormality analysis. The set of variables in V that can be used as transition input parameters is called the input variable set V_i, such as various data fields in the data packet; The set of remaining variables is collectively referred to as the context variable set V_c, then $V = V_i \cup V_c$; The variable value space of V is represented as $\overline{V}$, and the value of the current variable consists of a variable value vector $\overline{V_i} = (\overline{v_1}, \overline{v_2}, \cdots, \overline{v_n})$.

I: A non-empty input set, $\forall i \in I$ with a form $i = (v_1, v_2, \cdots, v_k)$, where $v_k \in V_i, k \geq 0$, $(v_1, v_2, \cdots, v_k)$ is a list of input parameters that describe the parameters that need to be provided at each step during the non- abnormality analysis.

P: A set of assertions that operate on variables. It describes what decisions the non- abnormality analysis process needs to make at each step.

A: An action set that operates on a variable. It describes which variables need to be manipulated when an assertion is established, such as an execution counter plus one or zero.

T: Non-empty transition set, which describes the state transition of Markov time-varying during non- abnormality analysis. It describes the conditions and effects when the transfer occurs.

$$T = S \times V \times I \times P \to S \times V \times A;$$
$$\forall t_i \in T, t_i = (s_{start}, i, p, a, s_{end}) \tag{2}$$

where s_{start} and s_{end} indicate the start and end states of the transition t_i. $i \in I$ indicates the input primitive of the transition t_i. $p \in P$ indicates the predicate conditional expression that must be satisfied when the transition t_i is performed. $a \in A$ Indicates the action that must be completed when the transition t_i is executed.

The construction process of the message space state diagram is as follows:

STEP (a). Constructing a set of states S which have been experienced during the operation of the state diagram according to the non-abnormality analysis process described in the 104 Statute. Constructing a variable set V containing input variables and context variables according to the non- abnormality analysis process described in the 104 protocol and the data fields provided in the protocol message data package;

STEP (b). Constructing the input set I according to the variables that need to be provided when the states of the non-abnormality analysis are transferred;

STEP (c). Constructing a set of assertions P that operate on the variables according to the conditions that need to be made during the transition of each state in the non- abnormality analysis process;

STEP (d). Constructing an action set A that operates on the variable according to the influence of each state on the variable during the non- abnormality analysis process;

STEP (e). Constructing a transition set T based on the state transition in the process of non- abnormality analysis;

When constructing the message space state diagram, it is necessary to note that the smart unit terminal that has not undergone security enhancement may participate in the interaction process. Therefore, when performing non-abnormality analysis, it is necessary to consider that the protocol packet message does not adopt the security enhancement mechanism. The workflow of the message space state is shown in Figure 4. After obtaining the data packet, it will send the data packet to its corresponding Markov time-varying for analysis according to the type of the protocol message. After the model obtains the data packet, it will send the data packet to its corresponding Markov time-varying for analysis according to the type of the protocol message. When analyzing the data in the packet, the model first analyzes the non-abnormality of the data packet structure. For example,

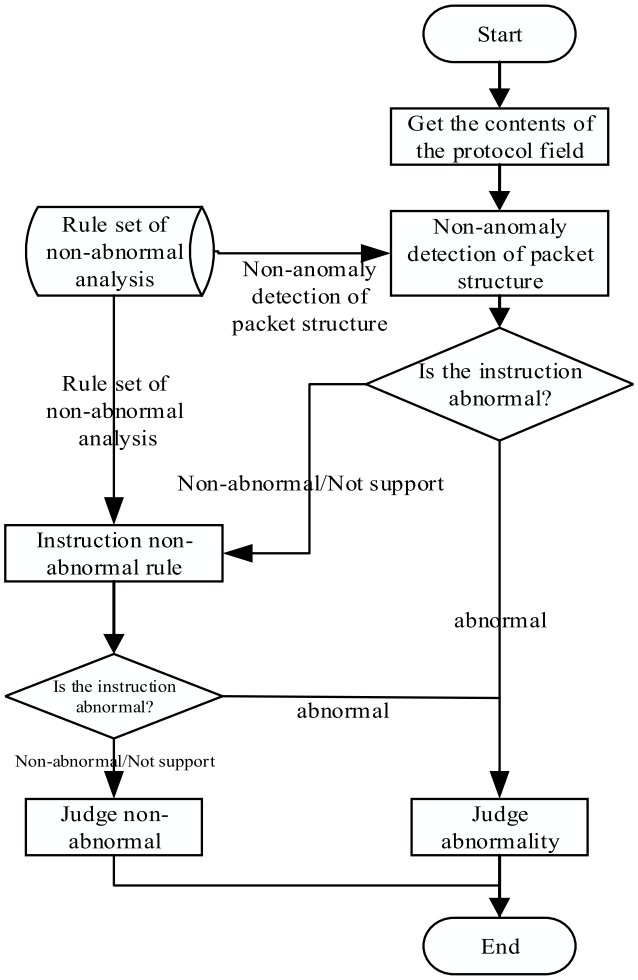

FIGURE 4. Flow of the message space state diagram.

it will analyze whether the port number used by the data is correct or whether the content of some data fields meets the requirements. If the structure of the packet does not meet the specifications, it is directly determined that the packet is not compliant. If the structure of the data packet conforms to the specification requirements or the terminal device that generated the data packet does not perform security enhancement, then it will analyze the non-abnormality of the data in the application layer of the packet again. It determines whether the consistency of the data content is correct or whether there is a replay attack on the content of the packet. If the content of the application layer data does not meet the specification requirements, it is determined that the data packet is not compliant. If the content of the packet application layer data meets the specification requirements or because the terminal device that generated the data packet does not perform security enhancement, the data packet is determined to be compliant.

Typical instruction exceptions include three types of exceptions: protocol structure exceptions (malformed packets), protocol interaction exceptions, and instruction function code exceptions (including frequency, range, and other

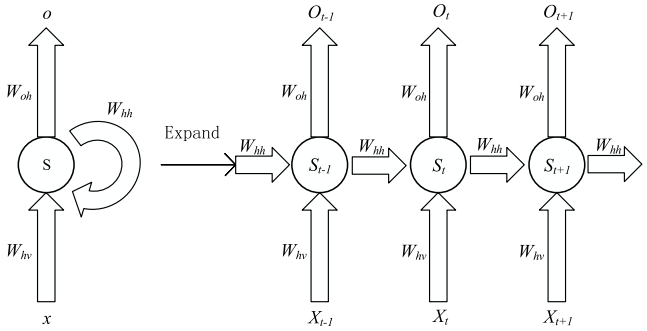

FIGURE 5. Recursive neural network structure.

feature exceptions). In the monitoring process of the interactive network terminal, when it reaches the preset calculation monitoring window period, the significant characteristic indicators of the command abnormality are calculated, such as service flow, remote adjustment uplink/downlink message ratio, remote control timeout interval, and the number of connected messages.

After packet depth parsing, combined with the MTM-based protocol feature modeling, the tracking matching related abnormal instruction identification can also be sent to the subsequent module for processing.

C. LSTM BASED ON AUTOCORRELATION COEFFICIENTS

Recurrent neural network (RNN) is a popular learning method in the field of deep learning in recent years. As shown in Figure 5, the RNN uses a looping structure in the network, establishing the connection of neurons to themselves. Through this structure, the neuron can ''memorize'' the input information from the previous moment in the neural network and affect the output of the current moment. Therefore, RNN can reflect the timing relationship of data, and often has a good performance in the analysis of the abnormal state.

LSTM is a variant of RNN. It replaces the neurons in the hidden layer of the RNN with the cell state. The information remains on the cells. As shown in Figure 6, cellular states are transmitted over the time chain with only a small amount of linear interaction. Each memory contains one or more memory cells and three nonlinear summation units. The nonlinear summation unit is also called ''Gate'' and is divided into three types: ''Input gate,'' ''Output gate'' and ''Forget gate,'' respectively. Matrix multiplication controls the input and output of memory cells [10].

The LSTM first calculates the partial derivative corresponding to the memory cell output value. Then calculate the output gate partial derivative. Then calculate the memory cell state, the forgetting gate, and the partial derivative corresponding to the input gate. Finally, the gradient weighting method is used to update the connection weight of the model.

The instruction state timing sequence is a collection of temporally ordered data. The time series model assumes that the law of past state data changes over time is also applicable to the future. We can judge the command anomaly by learning the information of historical state data, and further perform

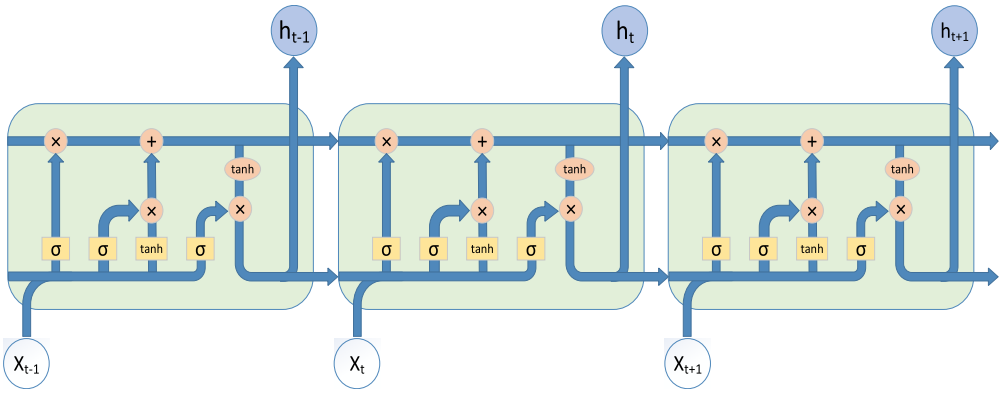

FIGURE 6. LSTM neural network model structure.

real-time abnormal state analysis and even short-term abnormal state analysis.

Assume that the time series is (y_1, y_2, $y_3 \ldots y_t$), where y_t represents the data value at time t. We analyze the value of y_{m+1} by the abnormal state by (y_{m-k+1}, $y_{m-k+2} \ldots y_m$), where k represents the number of data steps used for each abnormal state analysis. The autocorrelation coefficient r_k is used to indicate the degree of correlation between the time series itself and the lag k period data [10]:

$$\gamma_k = \frac{\sum_{t=1}^{T-k}(y_t - \bar{y})(y_{t+k} - \bar{y})}{\sum_{t=1}^{T}(y_t - \bar{y})} \tag{3}$$

where y_1, $y_2 \ldots y_T$ represent time series data; T represents the size of the data set; represents the average number of sequences. The higher the r_k value, the better the autocorrelation of the state data over the period k.

The RNN timing sequence abnormal state analysis model analyzes the trend of the series and analyzes the next moment according to the abnormal data state of the previous T time steps. According to experience, the instruction frame data has a positive autocorrelation in the period, which means that the data before a particular time can competently represent the characteristics of the frame data at the current time. However, since the previous frame data and the frame data of the first T time steps do not constitute a continuous timing relationship, the conventional RNN neural network is not suitable for modeling by adding autocorrelation features.

This paper proposes an improved LSTM network based on autocorrelation coefficients. As shown in Figure 7, the original LSTM network is unchanged, and the value analyzed by the abnormal state is combined with several values before a self-correlation cycle to form the input of the DNN network. By training the DNN network, the autocorrelation feature and the timing feature of the state data traffic are combined to achieve the purpose of discriminating the abnormal message. In the memory learning process of long-short-term memory networks based on autocorrelation coefficients, if a specific state is detected in the regular event set without this event, the event is an abnormal event. It is necessary to discriminate whether it is an error response of error event or information security of the abnormal event. Then it will record the feature

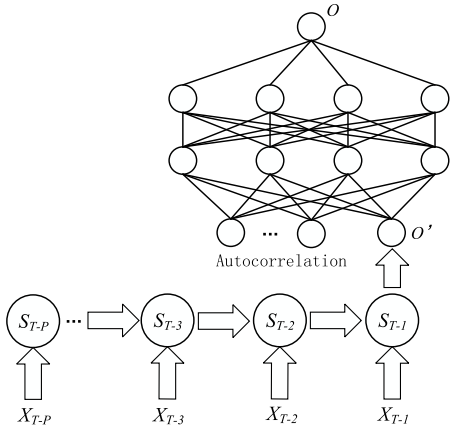

FIGURE 7. Improved LSTM structure.

of the unknown event, associated with the abnormal state set, construct the abnormal command state spanning tree (as shown as figure.8), and perform state expansion.

The construction algorithm of the abnormal instruction state generation tree takes the abnormal message types DM and SMA as input and derives the analytical SMA from the start field of the abnormal message type. First, create a root node (line 1) for the start field (DM.Z). Then, find the generator corresponding to the start field in the generation set (DM.P) of the exception message type (line 2). The production processing function, Production Disposal, is called to process it. Finally, the generation tree of the abnormal instruction state corresponding to the SMA is output.

Production Disposal implements the core function of SMA parsing. That is, the SMA is sequentially compared with the constituent elements on the right side of the generator to obtain the grammatic structure of the SMA. Before processing each component, we calculate the value of its inherited property (line 7). Because its value may affect the resolution of the post-order field. When dealing with grammar elements to identify SMA, Algorithm 3 is divided into three cases according to the type of field:

1) If it is an atomic field (lines 8-10), we extract the value of the field in the SMA according to the attribute value of the field (get-data function), and create a node (the Node function) to add the exception instruction state spanning tree;

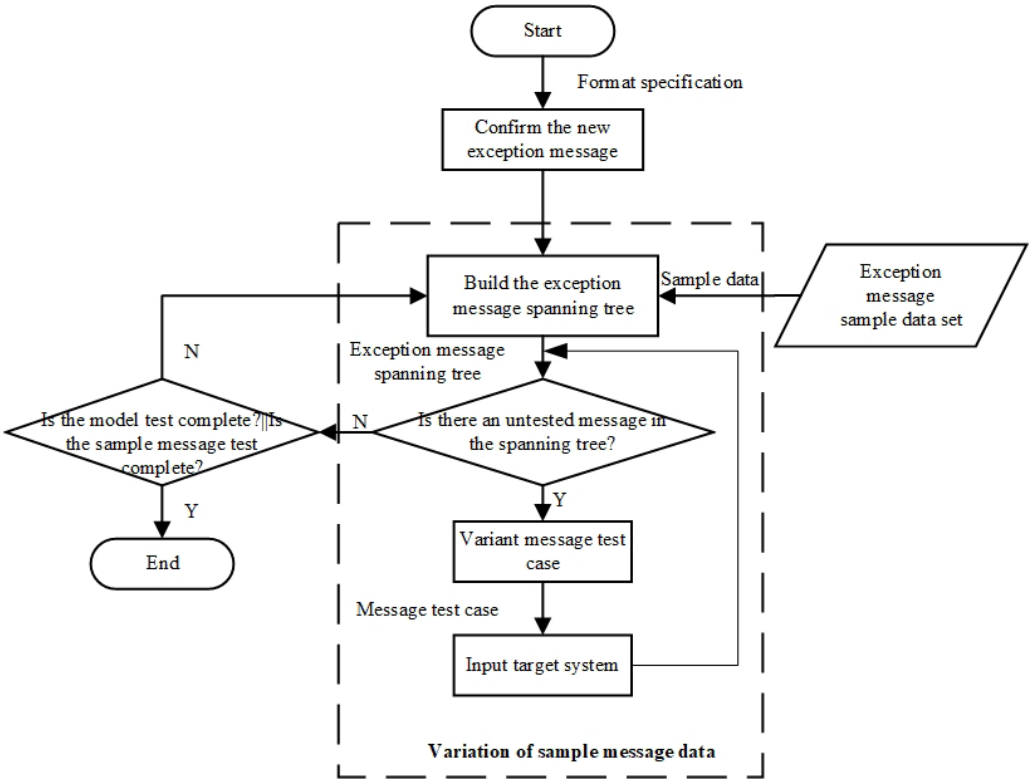

FIGURE 8. Abnormal packet status learning process of LSTM.

2) If it is a compound field containing structure attributes (lines 12-13), we calculate the structure attribute value corresponding to SMA according to the attribute rule first. Then we construct the analysis tree according to the general compound field processing method;

3) If it is a general compound field (lines 14-18), firstly, we create a node for the composite field to join the tree and find the generator of the composite field. Then we recursively call the production processing function (Production Disposal), until the SMA analysis is completed or the model derivation is terminated. At the same time, when the profiling of the composite field is ultimately resolved, the significant properties need to be calculated (line 18).

After the abnormal command state spanning tree is constructed, it is post-ordered and traversed, and the newly determined abnormal exception message is selected for automatic mutation. In the case where the mutated element is not manually specified, all fields are mutated by default. In order to avoid repeated sample learning of common fields between samples, the method introduces a set of measured feature elements for recording feature elements of the memory network that have been long and short. When looking for variability features, duplicate long-term memory networks are avoided by comparison with the set feature elements. At the same time, it can also be sent to subsequent module processing, such as Permit/deny, Drop to drop subsequent messages, Redirect, generate attack log alarms, and so on.

IV. EXPERIMENT AND ANALYSIS

Take the common subscriber/publisher model in the vehicle-road collaborative charging networks platform of State Grid Corporation as an example. Subscribers/publishers are primarily used to provide a one-way exchange of information for time-critical and periodic-based services. Moreover, subscriber/publisher can be used for the transmission of multi-party information, that is, the exchange of information between the publisher and multiple subscribers. This paper aims at the abnormality of the subscriber/publisher mode in the 104 protocol. First, we analyze the consistency of the message and determine whether the data packet structure meets the requirements of 104 specifications. Second, we apply the message. The layer data is analyzed to determine whether the data packet has a replay attack behavior.

The working process of this part of the 104 protocol is shown in Figure 9. In Figure 9, Stop represents the stop state; Retransmission pending represents the retransmission pending state; Retransmission represents the retransmission state.

The part of the non-abnormality analysis process in the 104 protocol.

Constructing a state set $S = \{s_1, s_2, \ldots, s_8\}$, where s_1 indicates that the message has been received, waiting for the status of the signature value to be extracted; s_2 indicates that the signature value is successfully extracted, waiting for the status of the verification signature value; s_3 indicates that

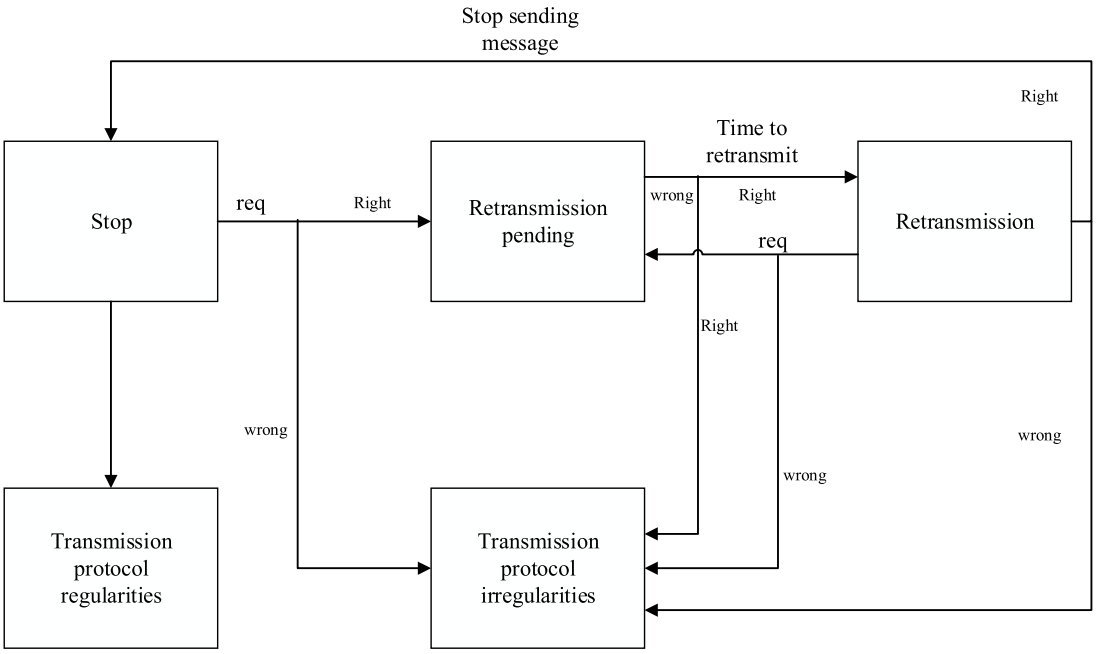

FIGURE 9. Subscriber/publisher process of IEC104.

the verification of the signature value is successful, waiting for the state of the reserved field 2; s_4 indicates that the reserved field 2 is extracted successfully, waiting for the status of the verification CRC value; s_5 indicates that the CRC value verification success, waiting to verify the status of the APDU time value is correct; s_6 indicates that the time value verification in the APDU is successful, waiting to verify the status of the status number in the APDU; s_7 indicates that the status number verification in the APDU is successful, the status of the data packet compliance; s_8 indicates the status of the packet is not compliant; s_1 is the initial state, s_7 and s_8 are the termination states.

Construct a input variables set $Vi=\{v_{i1},v_{i2},\ldots,v_{i16}\}$, where v_{i1} is the value of the TPID field in the message; v_{i2} is the value of the TCI in the message; v_{i3} is the value of the EtherType in the message; v_{i4} is the value of the APPID in the message; v_{i5} is the value of the Length in the message; v_{i6} is the value of reserved field 1 in the message; v_{i7} is the value of reserved field 2 in the message; v_{i8} is the content of the APDU in the message; v_{i9} is the content of the Reversed field in the message; v_{i10} is the content of the Private field in the message; v_{i11} is the content of the signature value in the message; v_{i12} is the value of the message time in the APDU; v_{i13} is the value of the message status number in the APDU; v_{i14} is the value of the allowable lifetime in the APDU; v_{i15} is the system time; v_{i16} is the maximum status number.

Construct a set of context variables $Vc=\{v_{c1}\}$, where v_{c1} is the most recent state number.

Construct an input set $I=\{i_1,i_2,..,i_6\}$, where $i_1 = (v_{i11})$ indicates the signature value in the input message; $i_2 = (v_{i3}, v_{i4}, \ldots, v_{i11})$ indicates the content from the EtherType field to the signature value field in the input message; $i_3 = (v_{i7})$ indicates that the content of the reserved field 2 in the input message; $i_4 = (v_{i1}, v_{i2}, v_{i3}, v_{i4})$ indicates the content from the TPID field to the APPID field in the input message; $i_5 = (v_{i12}, v_{i15})$, indicates the message time and system time in the input APDU; $i_6 = (v_{i13}, v_{i14}, v_{i16}, v_{c1})$ indicates the message status number, the allowable survival time, the maximum status number and the most recent status number in the input APDU.

In this paper, we construct an assertion set that operates on a variable $P = \{p_1, p_2, \ldots, p_{12}\}$, where p_1 indicates that the signature value is not empty; p_2 indicates that the signature value is empty; p_3 indicates that the signature value extracted in the message is the same as the calculated signature value; p_4 indicates that the signature value extracted in the message is different from the calculated signature value; p_5 indicates that the value of reserved field 2 is not empty; p_6 indicates that the value of reserved field 2 is empty; p_7 indicates that the value of reserved field 2 is the same as the calculated CRC check value; p_8 indicates that the value of reserved field 2 is different from the calculated CRC check value; p_9 indicates that the difference between the time obtained from the APDU and the system time is less than 2 minutes; p_{10} indicates that the difference between the time obtained from the APDU and the system time is greater than 2 minutes; p_{11} indicates that the status number obtained from the APDU is greater than or equal to the latest status number and the allowable lifetime does not expire; p_{12} indicates that the status number obtained from the APDU is less than the latest status number or the allowable lifetime timeout.

In this paper, the protocol fields are merely referred to as I, L, F, S, E, D, and C. Figure. 10 is a control flow graph and an execution path input for the first time, which

Algorithm 3 Abnormal Instruction State Generation Tree Construction Algorithm

Input: Exception packet type *DM*, **single data sample** *Sample*

Output: SMA abnormal command state generation tree *GT*

Gratree_Build (*DM,Sample*)

{

1) *GT.root*←*Node*(*DM.Z*);
2) *p* ←*Search*(*DM.Z,DM.P*);
3) *ProductionDisposal*(*p, GT.root, Sample*)
4) **return** *GT*;
5) }

ProductionDisposal(*production*, *treenode,Sample*)

{

6. **for each** *x* **in right of** *p*
7. *x* ←*RuleCompute_inherit*(*DM.R, x*);
8. **if**(*x* ∈ *F*)
9. *x.v* ←*getData*(*x, Sample*);
10. *treenode.child*←*Node*(*x*);
11. **else**
12. **if**(*x* ∈*SA*)
13. *x* ←*RuleCompute*(*DM.R, x*);
14. *childnode*←*Node*(*x*);
15. *treenode.child* ← *childnode*;
16. *p'* ← *Search*(*x, DM.P*);
17. *ProductionDisposal*(*p'*, *childnode*)
18. *x* ←*RuleCompute_ synthetize*(*DM.R, x*);

}

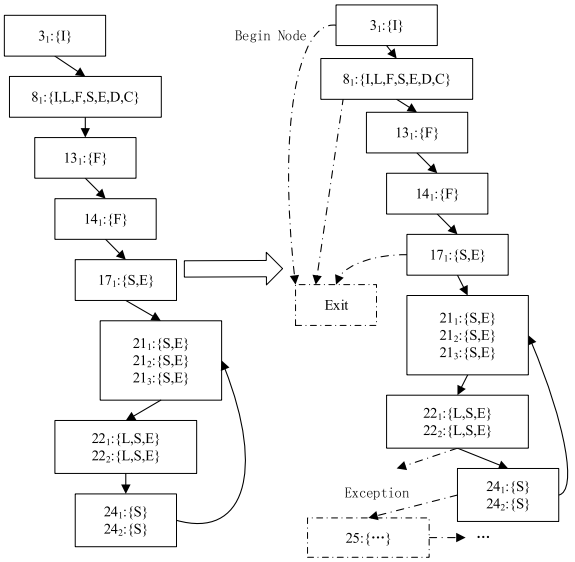

FIGURE 10. Control flow graph and execution path entered for the first time.

is constructed by an active input performing a write function. The dynamic control flow graph, the right half is the input path that is covered in the sample program run, and the dotted line is the newly discovered execution path at each node.

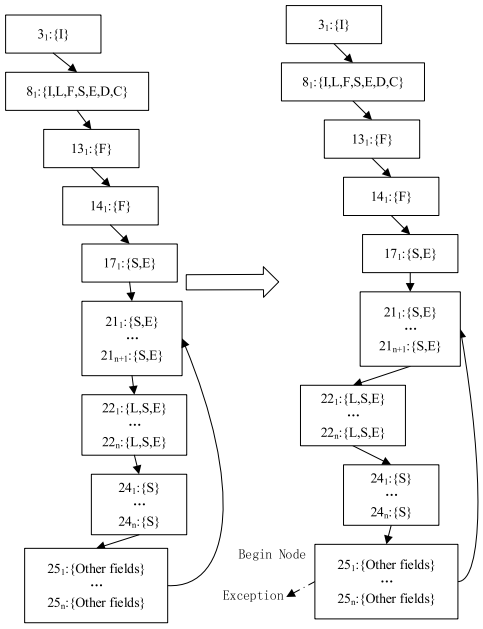

FIGURE 11. Control flow graph and execution path for the second input.

According to Definition 3, the starting node 3_1 in Figure.10 represents the first execution of the conditional branch corresponding to the third row of the sample code, the elements in the curly braces represent the field $DIF(3_1)$ that affects the conditional branch of the third row, and the edge with the arrow indicates dependence control relationship $CDC(3_1)$. The control flow shown in Figure. 11 is a test case generated by selecting an input as the second input, with a function code of 0x05 and n data field combinations in the case where the node 24_1 is true during the first input execution.

The Markov time-varying for this process part of the 104 protocol is shown in Figure 12. Construct the action set $A = \{a_1\}$ of the operation, where a_1 indicates that the most recent state number is assigned the current state number.

Construct a transition set T={t1,t2,...,t12}, where t1=(s1,i1,p1,s2), indicates the transition from s1 to s2; t2=(s1,i1,p2,s3), indicates the transition from s1 to s3; t3=(s2,i2,p3,s3), indicates the transition from s2 to s3; t4=(s2,i2,p4,s8), indicates the transition from s2 to s8; t5=(s3,i3,p5,s4), indicates the transition from s3 to s4; t6=(s3,i3,p6,s5), indicates the transition from s3 to s5; t7=(s4,i4,p7,s5), indicates the transition from s4 to s5; t8=(s4,i4,p8,s8), indicates the transition from s4 to s8; t9=(s5,i5,p9,s6), indicates the transition from s5 to s6; t10=(s5,i5,p10,s8), indicates the transition from s5 to s8; t11=(s6,i6,p11,s7), indicates the transition from s6 to s7; t12=(s6,i6,p12,s8), indicates the transition from s6 to s8.

During the abnormal test of the 104 protocol subscriber/publisher mode command of the State Grid from June 2017 to December 2017, a total of 76,324 exceptions occurred in 59 cases. By locating the exception code, merging exceptions with the same key code, we get four types

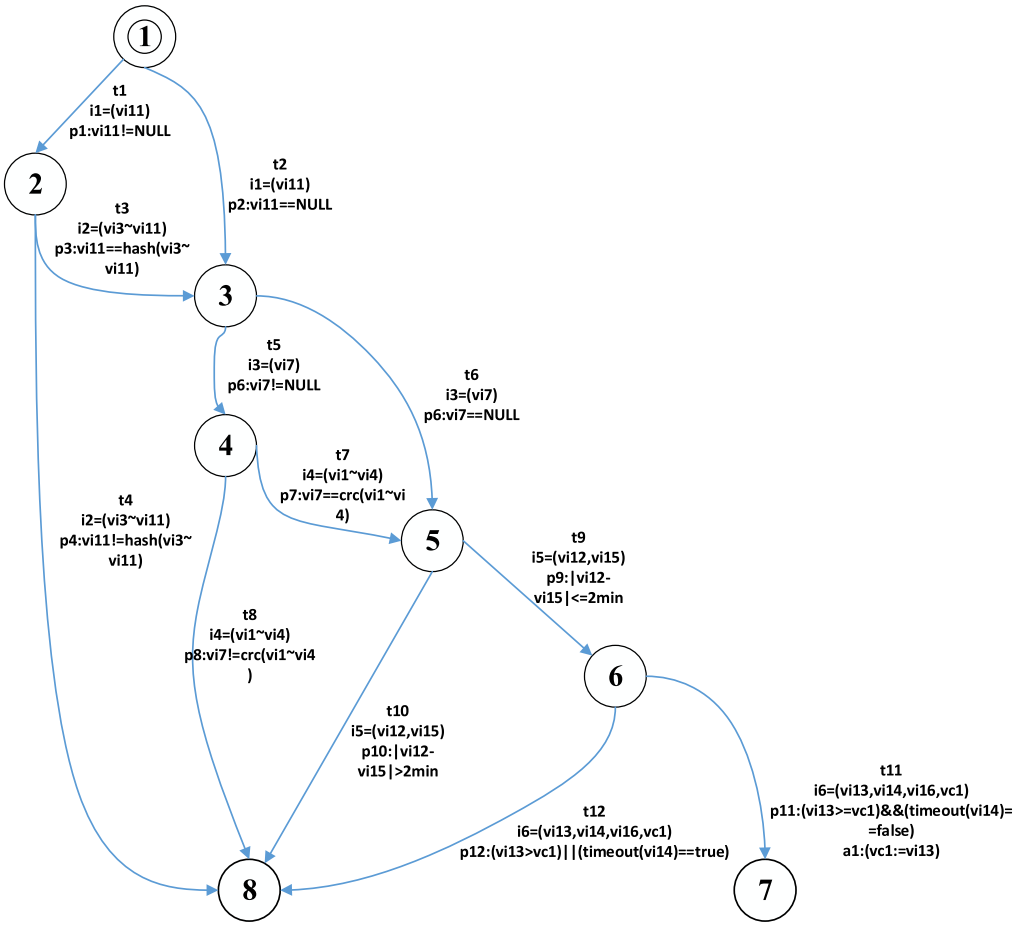

FIGURE 12. Instruction state transition of subscriber/publisher mode in 104 protocol based on MTM.

of exceptions. Then, we compare the vulnerability analysis and the published protocol vulnerability database. Three of them are verified as released vulnerabilities and the other is unknown vulnerability. Among them, two well known vulnerabilities (BAGTRAQ-8445, BAGTRAQ-8440) are caused by improper handling of the server name in the message returned by the host server to the network side server. As a result, a string overflow vulnerability and another well known vulnerability (BAGTRAQ - 8443) is a format string processing vulnerability that is triggered when a terminal processes a server information using a format string from a server. The unknown vulnerability is a stack overflow vulnerability in ACPI packets. Figure 13 shows the debugging information for this vulnerability.

As shown in Figure 13, the SEH chain in the stack is overwritten by the extra long message data, causing an exception in the execution of the program. If we construct the overlay data carefully, the attacker can execute arbitrary code with system privileges. The discovery of the vulnerability illustrates the effectiveness of the instruction-level anomaly feature extraction and parsing method for the industrial network with grid-load interaction.

This paper compares learning results of 9 types of IEC104 protocol messages by learning LSTM-DNN

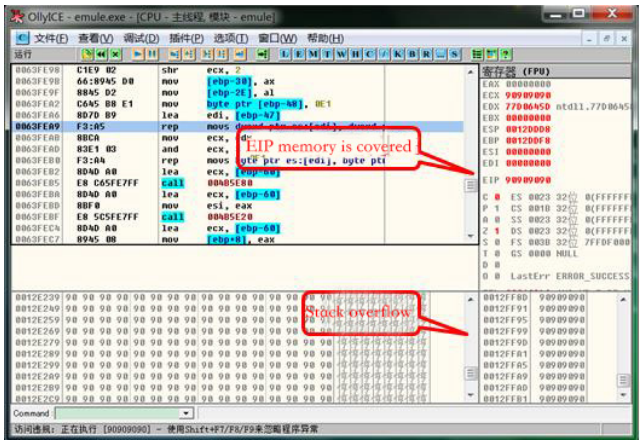

FIGURE 13. Unknown vulnerability debug in abnormal test of the 104 protocol subscriber/publisher mode command.

completely shown in Table 1. It can be seen from the data in Table 1 that although SPIKE and Peach generate the same total number of LSTM-DNN learning cases, the efficiency of LSTM-DNN learning cases is much lower than that of Peach. Because SPIKE adopts a flat block sequence structure, which has insufficient expression ability for the structure and constraint relationship in the protocol message. Thus it

TABLE 1. Comparison of the Effects of IEC104 Protocol Using LSTM-DNN.

LSTM-DNN learning tools	Number of learning cases for LSTM-DNN	Number of valid LSTM-DNN learning cases	Efficiency ratio of LSTM-DNN learning case	Code coverage ratio	LSTM-DNN learning time (h)	Number of vulnerabilities
SPIKE	225639	38053	10.7%	38%	58.5	2
PEACH	186324	91112	48.9%	52%	43.1	4
OUR METHOD	138017	76324	55.3%	65%	35.3	7

TABLE 2. Experimental Comparison Results.

Test method	Number of test case	Number of execution paths	Code coverage ratio	Test time/h	abnormal
CAD	16247	2896	50.90%	4.19	1
AFL-fuzzer	38652	1057	41.85%	3.35	1

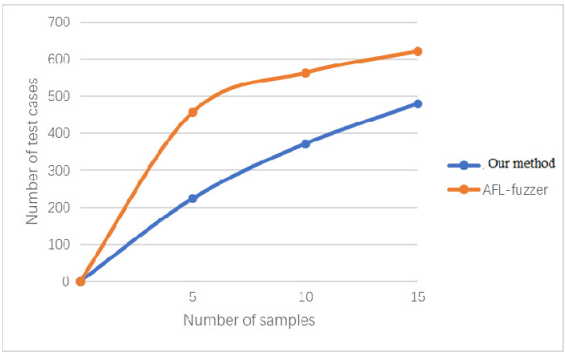

FIGURE 14. Number of test cases under different sample sizes.

generates too many invalid LSTM-DNN learning cases, and only two overflow vulnerability due to improper processing of the server name are found. Compared with Peach and this method, this method finds more vulnerabilities. It uses a smaller LSTM-DNN learning case to achieve higher code coverage. In addition, it should be noted that when defining the LSTM-DNN learning script, SPIKE and Peach need to write LSTM-DNN learning scripts for each type of IEC104 message format. This work is redundant and the script work is heavy. We only need to define a single exception packet type to parse all types of packet formats. The format is more concise.

We choose AFL-fuzzer as the tool for the comparison experiment. The slave station waits for a request message from its master station. In the experiment, the master station sends 5, 10, 15 request messages as test inputs to the slave station. Then the slave program is blurred and the target program is monitored to detect whether an exception has occurred. In order to evaluate the performance of themethod in this paper, we compare the number of test cases, the total number of execution paths, code coverage, and test time with the same number of samples.

The experiment counts the number of test cases generated by the two fuzzy test methods under 5, 10, and 15 sample data. The number of test cases here refers to the total number of samples generated after the program crashes or the sample is executed completely. It can be seen from Fig. 14 that with the increase of the number of samples, the test data generated by this method is significantly less than the test data generated by AFL-fuzzer. This is because the method uses the generated technology to construct test cases, compared with the use of mutation strategy. The AFL-fuzzer is naturally much less in number. In addition, we can see that the number of test cases generated by AFL-fuzzer increases more smoothly as the number of samples increases. This is because AFL-fuzzer considers that users may provide low-quality initial samples, resulting in certain types of Variations may result in data redundancy. So the input samples are pruned. While the methods in this chapter are more dependent on the number of samples, but the resulting strategy is more targeted.

Table 2 shows the statistics of the crash of the target program. When combining the program dynamic information to guide the test case generation, the test time is longer than using AFL-fuzzer. But there is a significant increase in the number of execution paths and code coverage. This indicates that during the running of the program, the more the number of execution paths found in the traversed conditional branch, the greater the possibility that the test case reaches the program to trigger the exception deeper, and its pertinence is also strengthened. Therefore, in the case of abnormality in the program, except for the test time, the method is superior to AFL-fuzzer in implementing 104 protocol test overall.

V. CONCLUSION

To execute command-level anomaly detection for vehicle-road collaborative charging network, this paper proposes a long-short-period neural network method for autocorrelation sequence processing combined with dynamic information. The method performs the tracking program execution, finds the input fields that affect the execution of the conditional branch through dynamic pollution analysis, and captures the dependencies between the conditional branches, to guide the grammar generation of the test cases and increase the chances of executing the deep code. The comparison experiment proves that the method improves the validity and code coverage of the test cases to a certain extent. It also dramatically increases the abnormal probability in the discovery protocol implementation.

REFERENCES

[1] A. Y. Javaid, W. Sun, V. K. Devabhaktuni, and M. Alam, "Cyber security threat analysis and modeling of an unmanned aerial vehicle system," in *Proc. IEEE Conf. Technol. Homeland Secur. (HST)*, Nov. 2012, pp. 585–590.

[2] H. Onishi, "Paradigm change of vehicle cyber security," in *Proc. 4th Int. Conf. Cyber Conflict (CYCON)*, Jun. 2012, pp. 1–11.

[3] J. Liu, Y. Xiao, S. Li, W. Liang, and C. L. P. Chen, "Cyber security and privacy issues in smart grids," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 981–997, 4th Quart., 2012.

[4] E. Schoitsch, C. Schmittner, Z. Ma, and T. Gruber, "The need for safety and cyber-security co-engineering and standardization for highly automated automotive vehicles," in *Advanced Microsystems for Automotive Applications*. Cham, Switzerland: Springer, 2016, pp. 251–261.

[5] A. Taeihagh and H. S. M. Lim, "Governing autonomous vehicles: Emerging responses for safety, liability, privacy, cybersecurity, and industry risks," *Transp. Rev.*, vol. 39, no. 1, pp. 103–128, Jul. 2018.

[6] M. S. Haque and M. U. Chowdhury, "A new cyber security framework towards secure data communication for unmanned aerial vehicle (UAV)," in *Security Privacy Communication Networks*. Springer, 2018, pp. 113–122.

[7] Q. Li, J. Hou, and Y. Qi, "A classification matching and conflict resolution method on meteorological disaster monitoring information," *Disaster Adv.*, vol. 6, pp. 415–421, Mar. 2013.

[8] L. Qm and H. Zhang, "Information security risk assessment technology of cyberspace: A review," *Information*, vol. 15, pp. 677–683, 2012.

[9] L. Qm and J. Li, "Rough outlier detection based security risk analysis methodology," *China Commun.*, vol. 9, pp. 14–21, Jul. 2012.

[10] K. Greff, R. K. Srivastava, J. Koutnìk, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[11] X. Liu, R. Zhu, B. Jalaian, and Y. Sun, "Dynamic spectrum access algorithm based on game theory in cognitive radio networks," *Mobile Netw. Appl.*, vol. 20, no. 6, pp. 817–827, Dec. 2015.

[12] S. Wan *et al.*, "Multi-dimensional data indexing and range query processing via voronoi diagram for Internet of things," *Future Gener. Comput. Syst.*, vol. 91, pp. 382–391, Feb. 2019.

[13] Z. Gao, D. Y. Wang, S. H. Wan, H. Zhang, and Y. L. Wang, "Cognitive-inspired class-statistic matching with triple-constrain for camera free 3D object retrieval," *Future Gener. Comput. Syst.*, vol. 94, pp. 641–653, May 2019.

[14] R. Zhu, X. Zhang, X. Liu, W. Shu, T. Mao, and B. Jalaian, "ERDT: Energy-efficient reliable decision transmission for intelligent cooperative spectrum sensing in industrial IoT," *IEEE Access*, vol. 3, pp. 2366–2378, 2015.

[15] S. Ding, S. Qu, Y. Xi, and S. Wan, "A long video caption generation algorithm for big video data retrieval," *Future Gener. Comput. Syst.*, vol. 93, pp. 583–595, 2019.

[16] B. Jalaeian, R. Zhu, H. Samani, and M. Motani, "An optimal cross-layer framework for cognitive radio network under interference temperature model," *IEEE Syst. J.*, vol. 10, no. 1, pp. 293–301, Mar. 2016.

[17] B. Liu, L. Shi, Z. Cai, and M. Li, "Software vulnerability discovery techniques: A survey," in *Proc. 4th Int. Conf. Multimedia Inf. Netw. Secur.*, Nov. 2012, pp. 152–156.

[18] D. Aitel, *An Introduction to SPIKE,the Fuzzer Creation Kit*. Accessed: Jun. 2018. [Online]. Available: http://www.blackhat.com/presentations/bh-usa-02/bh-us-02-aitel-spike.ppt

[19] G. Devarajan, *Unraveling SCADA Protocols: Using Sulley Fuzzier*. Accessed: 21, 2015. [Online]. Available: http://www.defcon.org/html/defcon-15/dc15-speakers.html

[20] *Peach*. Accessed: 21, 2018. http://www.peachFuzzer.com

[21] Zalewski. *American Fuzzy Lop*. Accessed: 25, 2017. [Online]. Available: http://lcamtuf.coredump.cx/afl/

[22] S. Gan *et al.*, "CollAFL: Path sensitive fuzzing," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Fransisco, CA, USA, May 2018, pp. 660–677.

[23] S. Rawat, V. Jain, A. J. S. Kumar, L. Cojocar, C. Giuffrida, and H. Bos, "Vuzzer: Application-aware evolutionary fuzzing," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, CA, USA, 2017, pp. 1–14. doi: 10.14722/ndss.2017.23404.

[24] X. Qi *et al.*, "Overview of industrial control network protocol fuzzing test technology," *Small Mirco Comput. Syst.*, vol. 36, no. 3, pp. 497–502, 2015.

[25] S. Aamir *et al.*, "Real time MODBUS transmissions and cryptography security designs and enhancements of protocol sensitive information," *Symmetry*, vol. 7, no. 3, pp. 1176–1210, Jul. 2015.

[26] T. Kwon, J. Lee, and O. Yi, "Vulnerability analysis and security modeling of MODBUS," *Adv. Sci. Lett.*, vol. 22, no. 9, pp. 2246–2251, Sep. 2016.

[27] C. Krushna and S. Magesh, "Analysis of vulnerabilities in the protocols used in SCADA systems," *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 4, no. 3, pp. 1014–1019, Mar. 2015.

[28] S. Bagaria, S. B. Prabhakar, and Z. Saquib, "Flexi-DNP3: Flexible distributed network protocol version 3 (DNP3) for SCADA security," in *Proc. Int. Conf. Recent Trends Inf. Syst.*, Dec. 2012, pp. 293–296.

[29] Z. Yang, Q. Bing, C. Lin, N. Yang, and D. Mei, "Research on short-term traffic flow prediction method based on similarity search of time series," *Math. Problems Eng.*, vol. 2014, Aug. 2014, Art. no. 18463.

[30] A. Bechtsoudis and N. Sklavos, "Aiming at higher network security through extensive penetration tests," *IEEE Latin Amer. Trans.*, vol. 10, no. 3, pp. 1752–1756, Apr. 2012.

[31] H. Shah. *Writing NASL Scripts*. [Online]. Available: http://www.infosecwriters.com/text_resources/pdf/NASL_H Shah.pdf

[32] T. Bartman and K. Carson, "Securing communications for SCADA and critical industrial systems," in *Proc. 69th Annu. Conf. Protective Relay Eng. (CPRE)*, Apr. 2016, pp. 1–10.

[33] R. W. Ritchey and P. Ammann, "Using model checking to analyze network vulnerabilities," in *Proc. IEEE Symp. Secur. Privacy SP*, May 2000, pp. 156–165. doi: 10.1109/SECPRI.2000.848453.

[34] X. Xu *et al.*, "An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles," *Future Gener. Comput. Syst.*, vol. 96, pp. 89–100, Jul. 2019.

[35] Y. Cheng, X. Zhou, S. Wan, and K. R. Choo,, "Deep belief network for meteorological time series prediction in the internet of things," *IEEE Internet Things J.*, to be published.

[36] S. Wan, Y. Zhang, and J. Chen, "On the construction of data aggregation tree with maximizing lifetime in large-scale wireless sensor networks," *IEEE Sensors J.*, vol. 16, no. 20, pp. 7433–7440, Oct. 2016.

[37] F. U. Din *et al.*, "Efficient sizing and placement of distributed generators in cyber-Physical power systems," *J. Syst. Archit.*, to be published.

QIANMU LI received the B.Sc. and Ph.D. degrees from the Nanjing University of Science and Technology, China, in 2001 and 2005, respectively, where he is currently a Full Professor with the School of Computer Science and Engineering. His research interests include information security, computing system management, and data mining. He was a recipient of the China Network and Information Security Outstanding Talent Award, in 2016, and the Education Ministry Science and Technology Awards, in 2012.

SHUNMEI MENG received the Ph.D. degree from the Department of Computer Science and Technology, Nanjing University, China, in 2016. She is currently an Assistant Professor with the School of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing, China. She has published papers in international journals and international conferences, such as TPDS, ICWS, and ICSOC. Her research interests include recommender systems, service computing, and cloud computing.

SHUO WANG was born in Harbin, Heilongjiang, China, in 1994. She received the B.S. degree in network engineering from the Nanjing University of Science and Technology, Nanjing, Jiangsu, China, in 2016, where she is currently pursuing the Ph.D. degree in software engineering. Her research interest includes the information security and intelligent transportation systems. She had received Suzhou Industrial Park Scholarship and First Prize Scholarship from NJUST, in 2017.

JING ZHANG received the Ph.D. degree in computer science from the Hefei University of Technology, Hefei, China, in 2015. He is currently an Associate Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology. He has published several articles in some prestigious journals, such as TNNLS, TKDE, TCYB, JMLR, TMM, and top-tier international conferences, such as AAAI, SIGKDD, SIGIR, and CIKM. His research interests include data mining, machine learning, and their applications in business and industry.

JUN HOU received the Ph.D. degree in computer science from the Nanjing University of Science and Technology, China, in 2019, where she is currently an Assistant Professor with the Zijin College. She has published dozens of articles in prestigious journals and top-tier conferences. Her research interests include data mining and information security. She serves as a PC Member for several international conferences and an external reviewer for many journals.