IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 26 February 2024, accepted 5 March 2024, date of publication 19 March 2024, date of current version 25 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3378533

RESEARCH ARTICLE

Firmware Vulnerability Detection Algorithm Based on Matching Pattern-Specific Numerical Features With Structural Features

PENG LIU, YASU CAO, YUJUN YAN, AND YONG WANG
State Grid Zhejiang Electric Power Company Ltd., Ningbo Power Supply Company, Ningbo 315000, China
Corresponding author: Peng Liu (liupeng8888855@163.com)

ABSTRACT With the continuous improvement of Internet of Things technology, Internet of Things devices are gradually popularized in people's lives and work, bringing convenience to people, but also there are many security risks. There are more and more types of attacks on IoT devices, and the security of their firmware has become a focus of attention. Aiming at firmware vulnerabilities in devices, a firmware vulnerability detection algorithm based on pattern-specific features and structural features is proposed in this study. The algorithm uses the two-stage method to filter and match the functions precisely, so as to find the functions matching the vulnerability functions. By reducing the local call graph from five layers to three layers, the algorithm operation and detection efficiency are improved, and the accurate matching method of weighted three-layer local call graph is implemented. The experimental results showed that the Top1 index value of the five-layer local call graph ranges from 81.99 to 90.19. The indexes of control flow chart and attribute control flow chart fluctuated greatly, ranging from 61.57 to 91.08 and 54.62 to 87.55, respectively. The Top1 index value of the weighted three-layer local call graph increased by 3.73%, and the average increased by 1.47%, indicating a significant improvement in the whole. It can be concluded that the firmware vulnerability detection algorithm based on the matching of pattern-specific numerical features and structural features can effectively find the function that actually matches the vulnerability function, and improve the efficiency of firmware vulnerability detection.

INDEX TERMS Vulnerability detection, pattern-specific features, structural feature matching, firmware security.

I. INTRODUCTION

The Internet of Things (IoT) has become an important tool in daily life and employment due to the expansion and advancement of the industrial revolution. It is widely used in various fields, including medical applications [1]. The integration of IoT technology has made life, work, and education more convenient and efficient. However, it has also made it easier for network intruders to exploit device vulnerabilities and spread malicious information throughout the network, infecting healthy devices and creating botnets [2]. The IoT security scenario is getting worse as IoT technology continues to advance, and Firmware Security (FS) is becoming more and more important [3]. It is crucial to identify and maintain FS since firmware acts as the device's fundamental controller, enabling the IoT device to carry out the required operations [4]. The known firmware vulnerability (FV) has a more severe and pervasive impact on IoT devices than the unknown FV because the majority of FS vulnerabilities are located in common devices like switches, cameras, etc. [5]. Therefore, in this study, a FV detection algorithm based on pattern-specific numerical features and structural features matching is proposed to detect the same origin vulnerabilities in the firmware of another platform. The algorithm proposed

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

© 2024 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

in this study can detect the same-origin vulnerabilities in the firmware of other platforms in the FVs of known platforms, which not only reduces the risk of the device system, but also improves the security of the system. Although firmware security is gaining increasing attention in IoT security research, effectively detecting and preventing FVs remains a challenge. Especially for the detection of homologous vulnerabilities, the existing research often lacks effective solutions. This study fills the knowledge gap in this field and provides a new vulnerability detection strategy. The research's innovation lies in the algorithm's ability to efficiently detect same-origin vulnerabilities, breaking through the limitations of firmware platforms. This provides a new protection mechanism for the firmware security of IoT devices. The goal of the research is to reduce the risk of known vulnerabilities attacking IoT devices and improve the security performance of the entire IoT system. The aim is to enhance security measures for IoT firmware and promote the development of IoT security technology.

This study presents an algorithm that can detect homologous vulnerabilities in firmware of known platforms. The algorithm reduces the risk of device systems and improves system security. While firmware security is receiving attention in IoT security research, detecting and preventing FVs remains a challenge. Existing research often lacks effective solutions for detecting homologous vulnerabilities. This study addresses a gap in the current knowledge of the field and presents a novel strategy for detecting vulnerabilities.

The study is broken into five main sections. The introduction provides an overview of the paper's structure and covers the present status of research on vulnerability detection (VD) algorithms. The literature review, which is the second section, gives an overview of the uses of the FS and VD approaches in important sectors and the present state of academic research on the two techniques.The FV problem in IoT devices is examined in the third part, where it is suggested that the FVD algorithm based on MSNC and SFM be investigated. The first subsection examines the function filtering method based on MSNC, the second subsection examines the FVD algorithm based on MSNC with structural features, and the third subsection examines the exact matching method based on the weighted triple-layer local call graph. The effectiveness of detecting actual FVs by merging MSNC and weighted three-layer local call graphs through comparative experimental studies is assessed in the fourth part. A summary and overview of the research's methodology and findings are provided in the fifth part. To better illustrate the connections between the research work, a flowchart for the content and methods of this research has been established. Please refer to Figure 1 for the framework of the research content.

This paper describes the current state of VD technology and its applications in related fields. It proposes and verifies the FVD algorithm, which combines MSNC and SFM to address the FV problem in the IoT. The algorithm includes a precise matching method that utilizes the structural features of MSNC and a weighted three-layer local call graph.

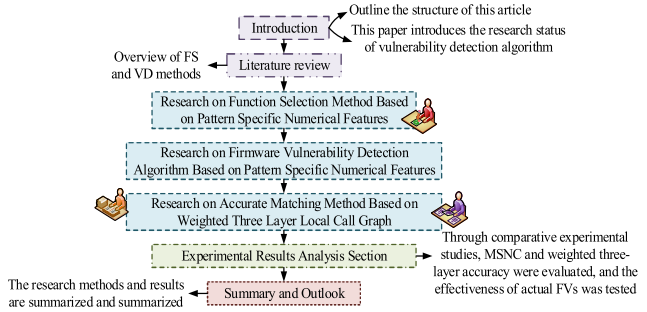


FIGURE 1. Pattern specific numerical feature filtering algorithm.

The proposed algorithm's advantages in actual FV detection have been verified, providing a clear reference for follow-up research.

II. RELATED WORKS

Due to the increased use of IoT devices, vulnerabilities in the firmware have emerged as a severe danger to the system's functionality. As a result, FS has gained more attention and has developed into a popular area of study for many academics. Vitale et al. proposed the concept of conformance testing to test firmware elements, i.e., intangible objects, i.e., the network. According to the experimental findings, version 3 of the test and test control notation took into account both the benefits and drawbacks of a scalable and adaptable network test environment [6]. Greis et al. proposed two deep learning methods, cyclic network architecture and convolutional network architecture, to improve the accuracy of IoT device type identification. The results showed that the accuracy of these two methods reaches 97% and 98%, respectively, which is significantly better than the traditional manual feature fingerprint recognition technology (accuracy of 82%). In addition, they offered a three-order-of-magnitude improvement in runtime performance over manual methods. The study explained the effectiveness of two methods in processing and analyzing traffic data flow through importance indicators [7]. Fala et al. proposed a secure firmware update framework for embedded systems that relies on hardware primitives and encryption modules to ensure the security of the update process in potentially insecure communication environments. The research results indicated that the framework can effectively resist a variety of attack vectors. Its adaptability and performance was verified by multiple test cases on FPGA, achieving the ability to securely update 1183kB of firmware images within 1.73 seconds [8]. Li et al. proposed a deep 3D blood vessel segmentation network based on edge profiles. The network architecture included a shared encoder and a dual decoder that jointly learns segmentation maps and edge profiles. Additionally, a bidirectional convolutional long short-term memory module was used to enhance 3D context awareness. The results indicated that the proposed method can significantly improve performance when training data is limited. Additionally, the computational efficiency of network reasoning was better than that of prior art [9].

Qasem et al. proposed various analysis techniques, such as static analysis, dynamic analysis, symbolic execution, and hybrid methods, to detect security vulnerabilities in the firmware of embedded systems in the IoT era. The results demonstrated that these techniques have both quantitative and qualitative advantages and disadvantages. Additionally, the authors identified the limitations of existing methods and discussed future research objectives [10].

There are many vulnerabilities in the device firmware, which makes some viruses use the security holes in the device firmware to spread and proliferate in the network, causing great threats and losses to the device, and many scholars have carried out many researches on FV. Cao et al. proposed a VD method called BGNN4VD, which is based on bidirectional graph neural networks. This method captures both syntactic and semantic information of code by integrating abstract syntax tree, control flow graph, and data flow graph. Then, it used the bidirectional edge enhanced by graph neural network to learn the distinguishing features of vulnerable and non-vulnerable code. Finally, it combines convolutional neural network to identify vulnerabilities. The study found that BGNN4VD significantly improved F1 measurement, accuracy, and precision compared to the other four advanced methods, with increases of 4.9%, 11.0%, and 8.4%, respectively. The method achieved an accuracy of 45.1% when dealing with the latest vulnerabilities in CVE reports, confirming its effectiveness in real-world scenarios [11]. Mcdaid et al. monitored device signals in home IoT networks using tools for interference analysis and wireless spectrum monitoring. According to experimental findings, RF spectrum analysis was a useful technique for tracking network activity using radio waves, but the level of expertise needed to interpret these patterns is limited [12]. Based on static or dynamic code analysis, Nong et al. weighed the benefits and drawbacks of cutting-edge memory error VD. Five cutting-edge memory error VD were empirically assessed and contrasted, respectively, with a benchmark dataset of C++ programme. Rational selection of the right tool with various trade-offs in accuracy and recall was inevitable [13]. Alrabaee et al. proposed using binary fingerprint identification for malware detection and vulnerability analysis. The results demonstrate the effectiveness of this method in identifying the properties of functions, authors, and libraries used in binary code. The limitations of the fingerprint recognition process, existing method application environments, and captured information types are discussed. Additionally, future research directions in this field are clarified [14].

In summary, as an important branch of IoT security, FV detection research has become relatively mature. Although early and current methods can widely detect FVs, these detection methods are often limited to the functional level of firmware code, and accuracy needs to be improved. In response to this issue, research innovatively adopts static detection methods, mainly focusing on FV scenarios on the platform, and then combines numerical and structural features that match specific patterns to detect functions that actually match vulnerability functions. This method has opened up new research directions and avenues for FV detection.

III. FVD ALGORITHM BASED ON MSNC AND SFM

Most of the entry points of cyber-attacks are focused on insecure IoT devices, and malicious attacks are carried out by exploiting the hidden vulnerabilities in the device firmware to achieve the purpose of destructive activities. Therefore, it is important to ensure the safe operation of the device system and to improve the security of the device firmware. Although FS is a crucial component of IoT security, FV is mostly present in firmware code functions, necessitating the matching of function features in order to achieve FVD.

A. NUMERICAL FEATURE SCREENING METHOD FOR FUNCTIONS BASED ON PATTERN SPECIFICITY

In the dynamic detection method, the VD of the firmware needs to be implemented according to the specific operating environment, but the cost of its implementation is too high and it is not applicable to all detection scenarios. For the consideration of factors such as algorithmic time cost and scalability, a static detection method is more suitable for FVD [15]. By extracting the relevant features of the function, the vulnerabilities that may be contained in the firmware are then detected. Based on the coarse-grained matching of the numerical features of the functions, the functions matching the VFs are filtered out. The Relief algorithm is used to calculate the importance of each numerical feature, and the larger the weight, the better the ability of the numerical feature to distinguish between different functions. The specific flow of the pattern-specific function numerical feature screening algorithm is shown in Figure 2.

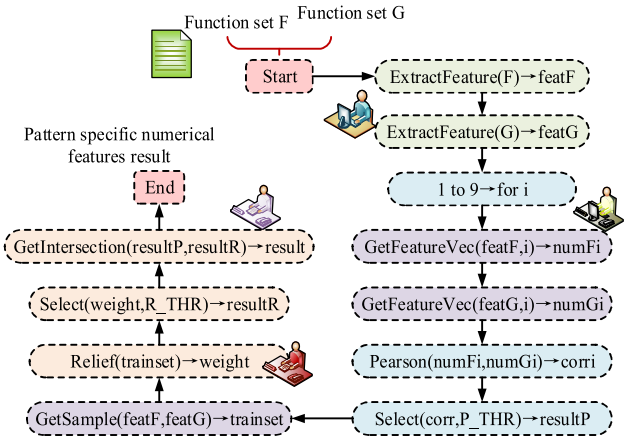


FIGURE 2. Pattern specific numerical feature filtering algorithm.

The algorithm for selecting numerical features specific to a pattern begins with data pre-processing, including normalization and denoising. Next, the F function set and G function set are applied to extract features that are compatible

with the algorithm and adaptable to it. The core of this algorithm is feature selection. Using statistical principles and machine learning techniques, this approach evaluates the discrimination and information content of the features generated by the F and G function sets. It then eliminates redundant and irrelevant features, optimizing the feature set for model training. This results in improved accuracy and efficiency for specific pattern recognition. Cross platform correlation analysis of numerical features, the numerical features should have high similarity on different platforms, the Pearson correlation coefficient is used for quantitative assessment, and its calculation equation is shown in Equation (1).

$$corr(x, y) = \frac{\sum (x - \mu_x)(y - \mu_y)}{\sqrt{\sum (x - \mu_x)^2 (y - \mu_y)^2}} \tag{1}$$

In Equation (1), x and y represent the original numerical characteristics, and μ_y and μ_x represent the mean values of y and x, respectively. Since the calculation of correlation coefficient has a certain symmetry, the results of correlation coefficient under correlation modes "$ARM \rightarrow MIPS$" and "$MIPS \rightarrow ARM$" are also equivalent. The Relief algorithm is used to determine the relative weights of various numerical features in order to objectively examine how well different functions can be differentiated from one another. The bigger the numerical feature weight, the better the effect of differentiation on the various functions. The results of the correlation coefficient calculation of each numerical feature under different correlation modes are shown in Table 1.

TABLE 1. Cross platform correlation analysis of numerical features.

FEATURE NAME	ARM→MIPS MIPS→ARM	ARM→x86 x86→ARM	MIPS→x86 x86→MIPS
Number of calls to function	0.081	0.091	0.056
Number of logical instructions	0.025	0.027	0.038
Number of data transmission instructions	0.007	0.017	0.003
Number of redirect instructions	0.045	0.039	0.063
Total Instructions	0.003	0.008	0.002
Number of Basic block	0.051	0.034	0.047
Local variable size	0.026	0.028	0.035
Edge number of Control-flow graph	0.042	0.065	0.054
Number of times a function has been called	0.082	0.094	0.096

The correlation coefficient of a numerical feature indicates its strength of correlation with the platform. A higher coefficient implies a stronger correlation and better differentiation ability. According to the table, numerical characteristics with correlation coefficients higher than 0.8 are deemed to be highly strongly correlated, and on the basis of this threshold, numerical features with improved discriminative abilities under various correlation modes are selected. The Relief algorithm is used to derive the degree of importance of different numerical features, which quantitatively describes the size of the distinguishing ability of numerical features under different functions. The larger the weight coefficient of the numerical features, the better the discriminative effect it produces. The numerical weight is measured by the distance of the vector, and its calculation formula is shown in Equation (2).

$$\cos(Vec(G_1), Vec(G_2)) = \frac{Vec(G_1) \cdot Vec(G_2)}{\|Vec(G_1)\| \, \|Vec(G_2)\|} \tag{2}$$

In Equation (2), $Vec(G_1)$ is the vector of ACFG1 and $Vec(G_2)$ is the vector of ACFG2. Its loss function definition Equation is shown in Equation (3).

$$L = (Label - \cos(Vec_1, Vec_2))^2 \tag{3}$$

According to Equation (3), the higher the similarity between $Vec(G_1)$ and $Vec(G_2)$, the closer the cos similarity value will be to 1, the smaller the loss value L will be, and vice versa. However, in the pattern-specific function numerical feature screening stage, kNN algorithm is used to filter the detection function, and Z-Score standardization is adopted to process the value of the function numerical feature. The calculation formula is shown in Equation (4).

$$z = \frac{x - \mu}{\sigma} \tag{4}$$

In Equation (4), x denotes the original numerical feature, μ denotes the overall mean of the numerical feature, and σ denotes the overall standard deviation of the numerical feature. With Z-Score normalization, the numerical features of the function are transformed into unitless Z-Score scores in order to achieve the elimination of the variability in the range of values of the numerical features [16]. The Euclidean distance obtained between the function to be matched and the numerical vector is calculated as its numerical characteristic distance, and its calculation formula is shown in Equation (5).

$$d(f, g) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{5}$$

In Equation (5), x_i and y_i denote the ith dimension normalized numerical features of the functions f and g to be matched, respectively, and n denotes the dimensions of the function numerical feature vectors in that association mode of the slot. The robustness analysis of the numerical features of the function with the generated training dataset is used to determine the numerical features under different association modes. Through the robustness analysis of the numerical features of the function, the numerical features under different association modes are determined, so that the features are screened in the face of the function to be detected. The screening process is shown in Figure 3.

By analyzing the numerical features of the function in a specific mode, the features that have a critical impact on the analysis of a specific mode are effectively selected, thus providing a higher quality input for the subsequent FV detection. On the basis of the completion of the numerical feature

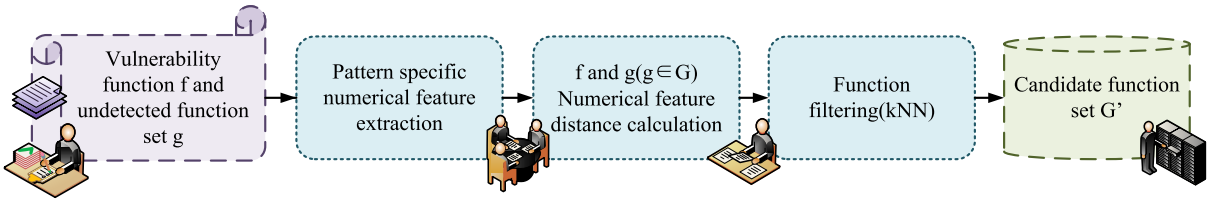


FIGURE 3. Function screening based on pattern specific numerical features.

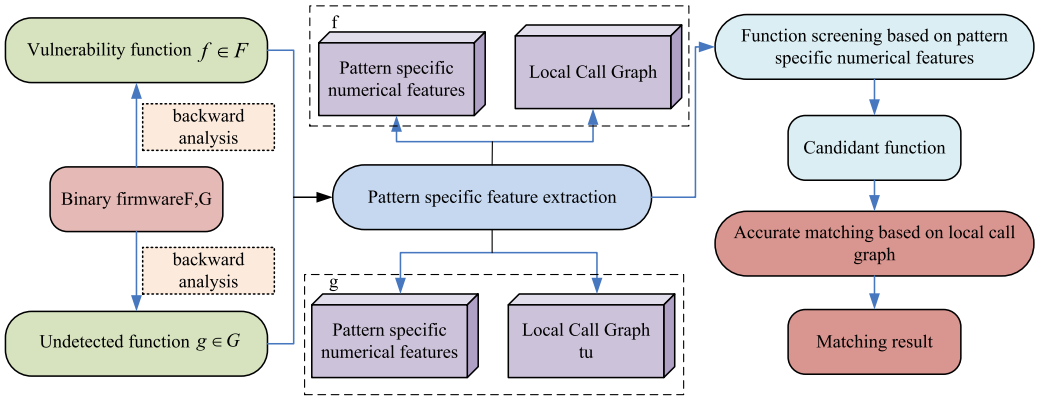


FIGURE 4. Overall framework of two-stage firmware vulnerability detection algorithm.

screening, the numerical feature and the structural feature are fully integrated to improve the accuracy and efficiency of FV detection, and realize the comprehensive and effective evaluation of firmware security. After obtaining the information of the VF and the function to be detected, the association pattern of the detection scenario can be determined. The association pattern is then used as the basis for extracting the dedicated numerical features of the VF and the function to be detected in the current association pattern, in combination with the numerical feature results of the function. The MSNC of the function to be matched is normalized using the Z-Score standard method, and the Euclidean distance between the numerical features of the function is calculated on the basis of this feature vector. Finally, the kNN algorithm is used to obtain the to-be-detected function as a candidate function.

B. FVD ALGORITHM BASED ON MSNC WITH STRUCTURAL CHARACTERISATION

Representation of functions using numerical features of different association patterns can effectively eliminate functions with low similarity to the VF and retain the functions to be detected that match the VF [10]. However, just using the numerical features of the function is not enough to complete the FVD task, therefore, a two-stage FVD algorithm is proposed. The detection process involves two steps. Firstly, the numerical features of the pattern are used to filter the functions to be detected, narrowing down the detection range of the target function. Secondly, the local call graphs of the functions are used to accurately match the candidate functions, enabling the detection of the function that the VF actually matches. The overall framework of this two-stage FVD algorithm is shown in Figure4.

The two-stage FV detection algorithm uses the reverse analysis tool to assemble the binary of the firmware, and the obtained assembly code is used as the research object. The numerical and structural features of firmware functions are extracted by writing corresponding plug-ins. The numerical feature is the statistical information generated under the analysis function syntax, while the structural feature is the local call graph related to the function to be matched generated from the call graph [17], [18]. Its numerical and structural characteristics are obtained by analyzing the function, and its calculation formula is shown in Equation (6).

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{6}$$

In Equation (6), $[h_{t-1}, x_t]$ denotes the splice of the output vector of the previous moment and the current vector, W_f denotes the matrix coefficients of the forgetting gate, b_f denotes the bias term and σ denotes the sigmoid function. Finally for the current state C_t output to the output value h_t, the portion of the output that is wanted is controlled by tanh, which is calculated as shown in Equation (7).

$$\begin{cases} O_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t = O_t \cdot \tanh(C_t) \end{cases} \tag{7}$$

Equation (7) is controlled by executing a sigmoid layer and in this way controlling its output. The basic block contains the instruction semantic embedding vector and statistical features, then the expression of the basic block is shown

in Equation (8).

$$B_{fea} = W_{b1}B_{emb} + W_{b2}B_{sta} \tag{8}$$

In Equation (8), B_{emb} and B_{sta} denote the instruction semantics of the basic block, and W_{b1} and W_{b2} denote the instruction semantics embedding vectors of the basic block and the weight matrices of the statistical features, respectively. Using the local call graph to analyze and calculate the distance on function structure and features, the function that actually matches the VF is derived on the basis of the structural distance between functions. Then, the main operating process of the FV detection algorithm is realized through the vulnerability function, as shown in Figure 5.

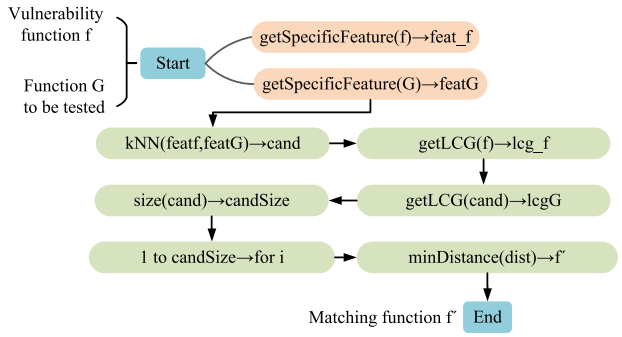


FIGURE 5. The main operation process of FV detection algorithm.

In the process of VD algorithm, firstly, the information of VD function f and function set G to be detected is obtained by using reverse analysis tool to determine the corresponding association pattern. Secondly, the dedicated numerical features of the association pattern are determined based on the numerical feature results of the functions, and the dedicated numerical features of the to-be-matched functions f and g A are extracted. Additionally, the normalized MSNC is used to calculate the numerical feature distance between the VF and the function to be detected, and the k functions that are closest to the VF are screened as candidate functions using the kNN algorithm. By solving the minimum power matching between two node sets in order to calculate the distance between the node sets, the equation for node set distance is shown in Equation (9).

$$d(s_1, s_2) = \frac{bm(s_1, s_2)}{\max(bm(s_1, \varphi_1), bm(s_2, \varphi_2))} \tag{9}$$

In Equation (9), $bm(s_1, s_2)$ denotes the minimum match between node sets s_1 and s_2 derived under the bipartite graph matching algorithm, φ_1 denotes the node set with the same number of nodes as the node set s_1 and the node numerical features are all zero, and φ_2 denotes the node set with the same number of nodes as the node set s_2 and the node numerical features are all zero. The structural feature distance between the VF and the candidate function is computed using the local call graphs of the two functions. Finally by finding the candidate function with the smallest structural feature distance from the VF as the final matching result.

C. FEATURE MATCHING BASED ON THREE-LAYER LOCAL CALL GRAPHS

The FV detection algorithm improves the accuracy and efficiency of VD by deeply integrating numerical and structural features. However, relying on these characteristics alone does not meet the full needs of complex VD. Therefore, a feature matching method based on three-layer local call graph is further proposed to effectively capture complex interaction behaviors between functions by constructing a fine call relationship graph, so as to improve the depth and breadth of VD and achieve more accurate security assessment. The candidate functions obtained from the screening using five-layer local call graph exact matching, although the exact matching method of this local call graph has a high accuracy rate, it requires too much time cost [19]. Therefore, this study proposes a weighted three-layer local call graph to reduce the time required for function structure feature distance computation. However, reducing the number of local call graph layers reduces the information on function structure features as well, affecting the accuracy of function SFM [20], [21]. To compensate for the loss of information on the function features, the information on the frequency of calls between functions in the three-layer local call graph is extracted to generate a weighted three-layer local call graph of functions. To achieve a better balance between the time efficiency and accuracy of the algorithm, the tuning scheme of the function exact matching method is shown in Figure 6.

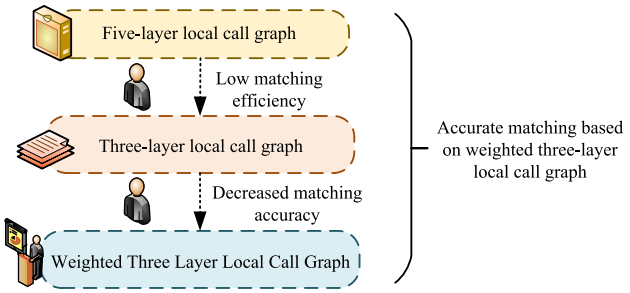


FIGURE 6. The main operation process of firmware vulnerability detection algorithm.

The algorithm reduces the number of layers of function local calls in order to speed up the computation time of the function to be matched in the exact matching phase, which leads to a decrease in the accuracy of exact matching of functions. For the information loss due to the reduction in the number of layers of structural features, the approach is to obtain the call frequency information in the function local call graph and generate a weighted local call graph on the basis of this information [10], [22]. For attribute control flow graph embedding will compute new vectors for each basic block, Gemini by training the interaction between the nodes, the equation is shown in Equation (10).

$$\mu_v^{(t)} = \tanh(\sigma(\sum\nolimits_{u \in N(v)})\mu_u^{(t-1)} + W_1 x_v) \tag{10}$$

In Equation (10), x_v denotes the feature vector of the basic block, μ_u denotes the vector representation of the embedded

network node u, and $N_{(v)}$ denotes the collocated node of the basic block v. Using the complex aggregation function in order to effectively represent the directionality of the graph, this fusion is spliced and its fusion equation is shown in Equation (11).

$$Z = Concat(Z_F, \alpha Z_{Sin}, \beta Z_{Sout}) \tag{11}$$

In Equation (11), α and β denote the different weights of in-degree convolution and out-degree convolution, and Z_F, Z_{Sin} and Z_{Sout} can obtain rich first-order and second-order neighborhood feature information. Compared with the five-layer local call graph, the weighted local call graph not only subtracts the indirect call relationship of the function to be matched, but also adds the call frequency information in the direct call relationship [10]. The call frequency information between functions is extracted, and then the weighted three-layer local call graph is generated from the call frequency information, and the weighted three-layer local call graph of its to-be-matched functions to f and g is shown in Figure 7.

In the three-layer local call graph, layer 0 represents the layer where the function to be matched resides, and the weighted three-layer local call graph can be calculated by the distance between node sets to obtain the nodes that match each other with minimum matching costs [23], [24]. The calculation method of the weighted structural distance is shown in Equation (12).

$$d(s_1, s_2) = \frac{\sum_{i=1}^{m} (\cos t(a_i, b_i) * \min(t(a_i), t(b_i))) + \sum_{j=m}^{n} t(b_j)}{\sum_{i=1}^{m} \max(t(a_i), t(b_i)) + \sum_{j=m}^{n} t(b_j)} \tag{12}$$

In Equation (12), s_1 and s_2 denote two node sets on a particular layer of the weighted three-layer local call graph, where $s_1 = \{a_1, a_2, \ldots, a_m\}$, $s_2 = \{b_1, b_2, \ldots, b_n\}$, where $t(\,)$ denotes the frequency of calls between the function of the current node and the to-be function of the acquisition, and $\cos t(\,)$ denotes the matching cost of the acquisition of the node. In this paper, the weighted three-layer local call graph is used to calculate the distance between node sets, and the matching cost of matching nodes is weighted and summed according to the call frequency information. Finally, it is used as the final distance between node sets, so as to obtain more information of function structure characteristics to a certain extent.

IV. PERFORMANCE TEST OF FVD ALGORITHM BASED ON MSNC AND SFM

This study selects a dedicated set of numerical features based on different association patterns to be tested for function screening. In order to determine the effects of various combinations of numerical features on the screening effect of functions, this study performs comparative experiments on the screening effect of functions on the test set [25]. Three function sets are obtained by compiling the generated binary files on three separate target platforms, namely ARM, MIPS, and x86, and are utilized as the test sets for the experiments in order to confirm the efficacy and validity of the tests. The numerical features of each function in the test set are extracted and their screening effects are compared to verify the effectiveness of MSNC [26]. For the completeness of the experimental tasks in different stages, the same experimental environment is set up, and its environmental parameters are shown in Table 2.

TABLE 2. Experimental environment for reverse analysis and function matching.

Experimental environment for reverse analysis and function matching	
Operating system	Windows
Processor	1.8Hz Intel Core i5
Memory	8GB
Compilation tools	arm-linux-gcc mip-linux-gcc gcc
Reverse analysis tools	IDA Pro
Function matching tool	Matlab

In Table 2, arm-linux-gcc is used to complete the compilation work on ARM platform, mip-linux-gcc is used to complete the compilation work on MIPS platform, and gcc is used to complete the compilation work on x86 platform. In order to assess the effectiveness of function screening, the evaluation index of recall is introduced, the higher the recall of function screening, the higher the probability that the function that actually matches with the VF is identified as a candidate function. The distance between the function's numerical feature vectors and the candidate function's average screening time are both used to determine how comparable the VF and the function to be detected are. Whereas the similarity is inversely correlated with distance between the function to be identified and the VF, and vice versa. The average screening time refers to the average time for a function to obtain the set of functions through screening, thus reflecting the efficiency of the function screening method in screening the set of functions to be detected. Since the combination of numerical features used in the screening effect comparison experiments of this study includes G and S, the corresponding function screening methods are labeled as kNN-G and kNN-S. The average comparison of the results after 50 experiments is shown in Table 3.

In Table 3, the overall performance of the kNN-S method is better than that of the kNN-G method under the six different association modes. kNN-S method under the kNN-S method improves the recall by an average of 5.17% relative to the kNN-G method, and the range of the enhancement is 2.11%∼7.72%. In the evaluation index of average Rank value, the kNN-S method reduces 13.98 on average compared with the kNN-G method, with a range of 3.57-22.53. This MSNC set screening method has a high probability of

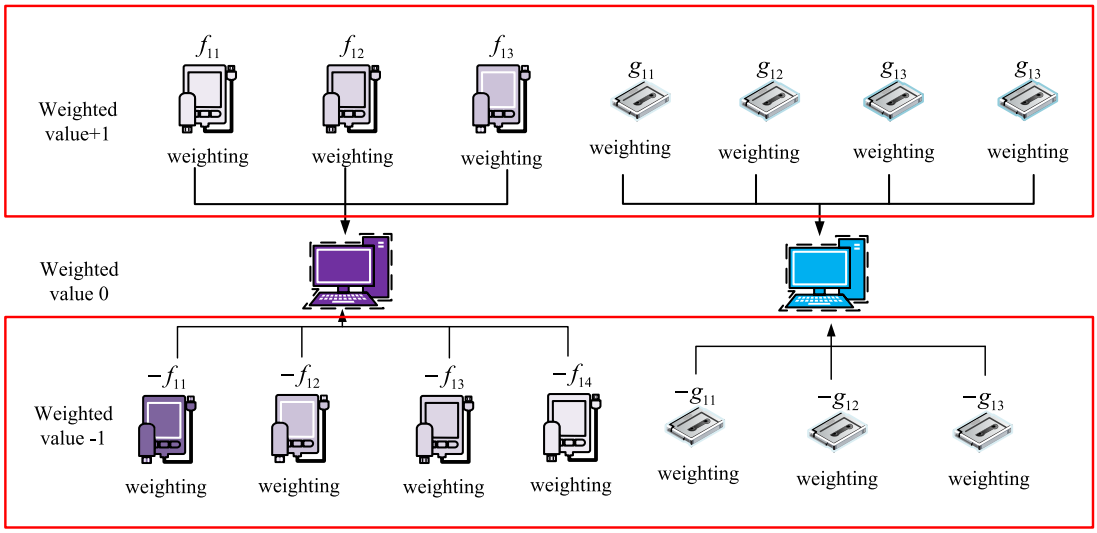


FIGURE 7. Weighted three layer local call graph of matched functions for f and g.

TABLE 3. Comparison of the average results of 50 experiments on NN-G and kNN-S50.

Associated mode	Recall(%)		Average Rank value		Average screening time(ms)	
	kNN-G	kNN-S	kNN-G	kNN-S	kNN-G	kNN-S
ARM→MIPS	89.47 %	93.87 %	51.59	43.81	0.48ms	0.48ms
MIPS→ARM	92.26 %	94.35 %	40.75	37.16	0.46ms	0.46ms
ARM→x86	78.05 %	85.77 %	94.71	72.52	0.48ms	0.46ms
x86→ARM	86.19 %	93.31 %	56.12	33.46	0.46ms	0.45ms
MIPS→x86	86.61 %	93.73 %	61.83	41.98	0.49ms	0.48ms
x86→MIPS	93.07 %	95.41 %	43.91	35.94	0.47ms	0.48ms

filtering the target function into the candidate function set, which is very suitable for rapid screening of the function set to be detected. It can be seen that the function screening efficiency of the kNN-G method and the kNN-S method is basically the same, which is due to the fact that the function to be detected is obtained by the Euclidean distance between the numerical feature vectors of the function, and if the numerical feature dimensions do not change much, the efficiency of the function screening method is more or less the same. In order to describe the matching accuracy in the precise matching stage, TopK indicator is used, the larger the value of this indicator, the better the correlation effect of the function in the precise matching stage, and the higher the precise matching accuracy. The accuracy of the exact matching method is measured as Top1, Top10 and Top100. The legend contains control flow graphs (CFG), attributed control flow graphs (ACFG) and local call graphs (LCG). Among them, the comparison experimental results of Top1 metrics for the three structural features are shown in Figure 8.

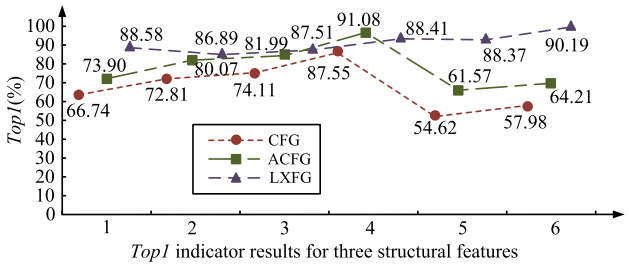


FIGURE 8. Comparison results of Top1 indicators for three structural features.

In Figure 8, the Top1 indicator of LCG5 achieves better results under all the association modes in this study, and is overall better than the two structural features of CFG and ACFG. Among them, the Top1 indicator value of LCG5 shows a better trend under the six correlation modes, and its indicator value is roughly around 85%, with the highest value of 90.19 and the lowest value of 81.99. The indicator of CFG is only second to the Top1 indicator value of LCG5, and it shows a larger fluctuation, with the highest value of 91.08 and the lowest value of 61.57. The indicator of ACFG also shows a large fluctuation, and is one of the three structural features of the study. The indicator of ACFG also shows large fluctuations and is the lowest among the three structural features, with the highest value of 87.55 and the lowest value of 54.62. It can be concluded that the accuracy matching method of LCG5 is more stable under the six association modes, which not only has a high matching accuracy, but also has the smallest change in the matching accuracy under the six association modes. And the comparison experimental results of the Top10 indicators of the three structural features are shown in Figure9.

In Figure 9, the Top10 indicators of LCG5 have achieved better results under all the association modes in this study,

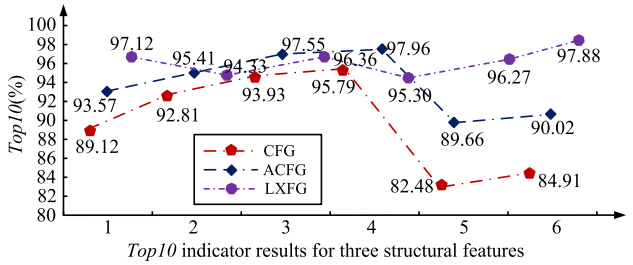


FIGURE 9. Comparison results of Top10 indicators for three structural features.

and are overall better than the two structural features of CFG and ACFG. Among them, LCG5's Top10 indicator values under the six correlation modes show a better trend, and its indicator values are roughly around 87%, with a maximum of 97.88 and a minimum of 94.33. CFG's indicator is second only to LCG5's Top10 indicator values, and it shows larger fluctuations, with a maximum of 95.79 and a minimum of 82.48. ACFG's indicator also presents The indicator of ACFG also shows large fluctuations and is the lowest among the three structural features, with the highest value of 97.96 and the lowest value of 93.57. Thus, it is evident that the accuracy matching method of LCG5 is more stable in the function matching effect under the six association modes [27]. This method not only achieves high matching accuracy but also exhibits the least variation in matching accuracy across the six association modes. In addition, the Top100 metrics of the three structural features are compared and experimented, and in order to verify the efficiency of the algorithm's precision matching phase, the average time of structural feature distance calculation is used to evaluate it, and the results are shown in Figure 10.

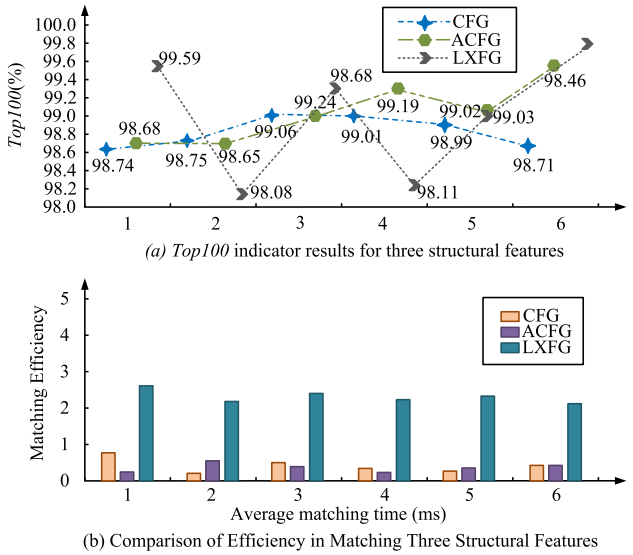


FIGURE 10. Comparison of Top100 index results and matching efficiency of three structural features.

In Figure 10, the Top100 indicators of LCG5 all achieved better results under the correlation modes in this study and were overall better than the two structural features of CFG and ACFG. LCG5's Top100 indicators show a better trend under the six correlation modes, with indicator values around 84%, ranging from a maximum of 99.63 to a minimum of 98.08. CFG's indicators are second only to LCG5's Top100 indicators, but they exhibit larger fluctuations, with a maximum of 99%. The indicators for ACFG also show fluctuations, but they are lower than the other two structural features. The maximum value is 98.46 and the minimum value is 98.65. In the comparison of the matching efficiency of the three structural features, LCG5 has the highest matching efficiency, with a matching time of 2.37ms, while CFG uses an average matching time of 0.43ms and ACFG uses an average matching time of 0.41ms. It can be seen that the accuracy matching method of LCG5 is more stable in the function matching effect under the six correlation modes, not only has a high matching accuracy, and in the six correlation modes under the minimum change in the matching accuracy. To assess the algorithm's effectiveness at all stages, we must consider its overall performance in the real FV. Therefore, in this chapter, we compare the algorithm to existing related work such as discovRE and Diff. Its experimental results are shown in Figure 11.

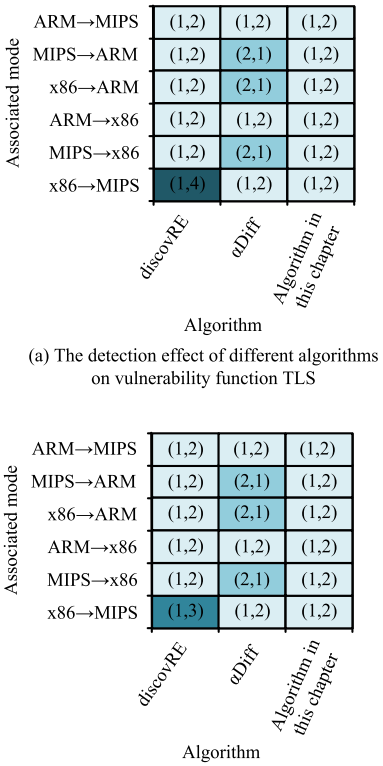


FIGURE 11. Comparative experimental results of vulnerability detection using different algorithms.

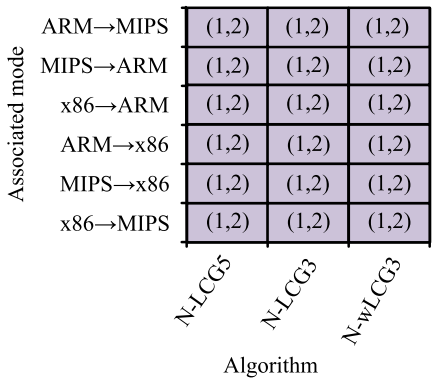
In Figure 11, two Rank values are recorded for each element, and the two Rank values are separated by a comma. The first Rank value indicates the Rank value of the corresponding

TABLE 4. Comparison of matching accuracy of three local call graph features.

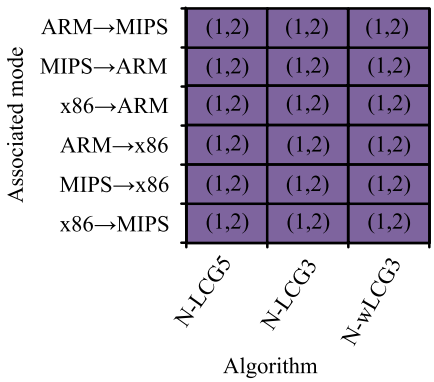
Associated mode	Top1(%)			Top10(%)			Top100(%)		
	LCG5	LCG3	w LCG3	LCG5	LCG3	w LCG3	LCG5	LCG3	w LCG3
ARM→MIPS	88.73	87.33	87.85	97.45	95.99	96.62	99.67	99.65	99.47
MIPS→ARM	87.15	85.77	87.13	94.64	94.57	95.16	97.64	97.21	96.73
ARM→x86	87.69	82.46	86.34	96.99	94.71	96.21	99.66	98.83	98.91
x86→ARM	88.58	85.73	88.16	96.47	95.42	96.76	98.17	97.71	98.25
MIPS→x86	88.63	87.31	88.27	96.59	96.38	94.77	99.26	98.73	98.61
x86→MIPS	90.34	89.24	88.78	98.11	97.36	97.53	99.73	99.61	99.53

target function when VD is performed with the current VF as the target. And the second Rank value records the Rank value of a non-target function that is similar to the current VF. The smaller the two Rank values in the table, the better the VD effect of the algorithm, so (1,2) is the best detection result, which indicates that the target function corresponding to the VF can be accurately detected. In Figure 10(a), when the TLS function is used as the target for VD, the worst VD result is (1,4) under x86→MIPS, while the αDiffx function has better VD results under 86→ARM, x86→MIPS and ARM→MIPS, and the rest of the VD results are the best. In Figure 10(b), when VD is performed with the TLS function as the target, the worst VD result is (1,3) under x86→MIPS, while the αDiffx function has a better VD under 86→ARM, x86→MIPS and ARM→MIPS, and the rest of the VD results are the best. It can be observed that this research algorithm can effectively detect the VF, and the detection effect is also the best, the matching effect under all six association modes is more stable, and has a high matching accuracy rate. It shows that the algorithm in this paper has better scalability and can be better adapted to different association modes. In order to evaluate the effectiveness of the weighted three-layer local call graph, the five-layer call graph is recorded as LCG5, the three-layer local call graph is recorded as LCG3, and the weighted three-layer local call graph is recorded as wLCG3, where LCG denotes the local call graph, the number after LCG denotes the number of layers of the local call graph, and w denotes the weighting. The results of their comparison experiments are shown in Table 4.

In Table 4, among the Top1 metrics, the matching accuracy of LCG5 is better than that of the other two local call graphs in all the six association modes. WLCG3's Top1 metrics value is between LCG3 and LCG5, and its overall has a great improvement to a certain extent with respect to LCG3. The Top1 metric value of wLCG3 has improved by 3.73% at the most and by 1.47% on average. As for the Top10 metrics, the experimental results of LCG5 are closer, and the matching accuracy of wLCG3 is better than that of LCG3. However, on the Top100 metrics, the differences in the matching accuracy of all the three kinds of local call graphs are small, and their reference significance is not significant. Combining MSNC with different local call graphs, the algorithm based on MSNC with five-layer local call graphs (N-LCG5), the algorithm based on MSNC with three-layer local call graphs (N-LCG3), and the algorithm based on MSNC with weighted three-layer local call graphs (N-wLCG3) are presented respectively, with N denoting the numerical features. The results are shown in Figure 12.



(a) The detection effect of different algorithms on vulnerability function TLS



(b) The detection effect of different algorithms on vulnerability function DTLS

FIGURE 12. Comparative experimental results of vulnerability detection using different algorithms.

In Figure 12, Figure10(a) and Figure 10(b) both when VD is performed with structural features, the three algorithms have consistently good VD results under 86→ARM,

x86→MIPS and ARM→MIPS, and their VD values are (1,2). As can be observed, this study algorithm has a high matching accuracy rate, a more stable matching effect under the six association modes, and the best detecting effect when it comes to VF detection. It shows that the target function corresponding to the VF can be accurately detected, and the similar non-target function of the vulnerability can also obtain the similarity ranking second only to the target function. The comparison shows that the accuracy of the weighted three-layer local call graph matching method is similar to that of the five-layer method, with only a small difference in efficiency. This achieves a better balance between accuracy and efficiency.

V. CONCLUSION

With the rapid development of IoT technology, IoT devices are promoted and popularized. However, when more and more devices are connected to the Internet, the security problems of their devices become increasingly significant. In order to improve device security, it is very important to detect and repair FVs in time. This study proposes an innovative method for detecting FVs. The method combines pattern-specific numerical features and structural feature matching to reduce the risks associated with FVs and improve overall system security. The effectiveness of the FV detection method is verified through a comparison experiment. The experimental results showed that compared to the kNN-G method, the recall rate of the kNN-S method increased by an average of 5.17%, with an improvement range of 2.11% to 7.72%. In the average rank evaluation index, the kNN-S method showed an average decrease of 13.98, with a decrease range of 3.57-22.53. The effectiveness of the proposed screening method in accurately screening the objective function to the candidate function set and quickly completing the screening work of the function set to be detected was demonstrated. Additionally, the exact matching method using a weighted three-layer local call graph was nearly as accurate as the exact matching method of a five-layer local call graph. This method achieved a good balance between efficiency and accuracy. However, due to the rapid development of technology and the widespread application, FV detection still faces more challenges and problems.

Further improvement of detection algorithms is still needed in future research. Firstly, optimizing the accuracy of the algorithm requires a more in-depth study of numerical and structural features, as well as their impact on FV identification. Secondly, improve the efficiency of the algorithm to quickly identify potential vulnerabilities in a large number of IoT devices. In addition, increasing the universality of FV detection algorithms can adapt to a wider range of device types and firmware structures, providing stronger security for IoT devices.

REFERENCES

[1] Y. Guo, Z. Mustafaoglu, and D. Koundal, "Spam detection using bidirectional transformers and machine learning classifier algorithms," *J. Comput. Cognit. Eng.*, vol. 2, no. 1, pp. 5–9, Apr. 2022.

[2] I. Hidayat, M. Z. Ali, and A. Arshad, "Machine learning-based intrusion detection system: An experimental comparison," *J. Comput. Cognit. Eng.*, vol. 2, pp. 88–97, Jul. 2022.

[3] Y. Lei, "Research on microvideo character perception and recognition based on target detection technology," *J. Comput. Cognit. Eng.*, vol. 1, no. 2, pp. 83–87, Jan. 2022.

[4] R. Zhou, R. Zhou, and Z. Zhu, "K-means clustering algorithm-based detection of carotid atherosclerotic plaque using contrast-enhanced ultrasound images," *Sci. Program.*, vol. 2021, pp. 1–7, Sep. 2021, doi: 10.1155/2021/2223344.

[5] F. K. de Araújo Lima, J. M. Guerrero, F. L. Tofoli, C. G. C. Branco, and J. L. Dantas, "Fast and accurate voltage sag detection algorithm," *Int. J. Electr. Power Energy Syst.*, vol. 135, Feb. 2022, Art. no. 107516, doi: 10.1016/j.ijepes.2021.107516.

[6] A. Vitale and M. Dacier, "Inmap-t: Leveraging TTCN-3 to test the security impact of intra network elements," *J. Comput. Commun.*, vol. 9, no. 6, pp. 174–190, Feb. 2021, doi: 10.4236/jcc.2021.96010.

[7] V. Steinhage, J. Greis, A. Yushchenko, M. Meier, and D. Vogel, "Automated identification of vulnerable devices in networks using traffic data and deep learning," *Int. J. Inf. Privacy, Secur. Integrity*, vol. 5, no. 1, p. 1, 2021.

[8] S. Falas, C. Konstantinou, and M. K. Michael, "A modular end-to-end framework for secure firmware updates on embedded systems," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 18, no. 1, pp. 1–19, Jan. 2022.

[9] X. Li, R. Bala, and V. Monga, "Robust deep 3D blood vessel segmentation using structural priors," *IEEE Trans. Image Process.*, vol. 31, pp. 1271–1284, 2022, doi: 10.1109/TIP.2021.3139241.

[10] A. Qasem, P. Shirani, M. Debbabi, L. Wang, B. Lebel, and B. L. Agba, "Automatic vulnerability detection in embedded devices and firmware: Survey and layered taxonomies," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–42, Mar. 2022, doi: 10.1145/3432893.

[11] S. Cao, X. Sun, L. Bo, Y. Wei, and B. Li, "BGNN4 VD: Constructing bidirectional graph neural-network for vulnerability detection," *Inf. Softw. Technol.*, vol. 136, Aug. 2021, Art. no. 106576, doi: 10.1016/j.infsof.2021.106576.

[12] L. Wartschinski, Y. Noller, T. Vogel, T. Kehrer, and L. Grunske, "VUDENC: Vulnerability detection with deep learning on a natural codebase for Python," *Inf. Softw. Technol.*, vol. 144, Apr. 2022, Art. no. 106809.

[13] A. McDaid, E. Furey, and K. Curran, "Wireless interference analysis for home IoT security vulnerability detection," *Int. J. Wireless Netw. Broadband Technol.*, vol. 10, no. 2, pp. 55–77, Jul. 2021, doi: 10.4018/ijwnbt.2021070104.

[14] S. Alrabaee, M. Debbabi, and L. Wang, "A survey of binary code fingerprinting approaches: Taxonomy, methodologies, and features," *ACM Comput. Surv.*, vol. 55, no. 1, pp. 1–41, Jan. 2023, doi: 10.1145/3486860.

[15] C. Ntantogian, P. Bountakas, D. Antonaropoulos, C. Patsakis, and C. Xenakis, "NodeXP: NOde.js server-side Javascript injection vulnerability DEtection and eXPloitation," *J. Inf. Secur. Appl.*, vol. 58, May 2021, Art. no. 102752, doi: 10.1016/j.jisa.2021.102752.

[16] Y. Nong, H. Cai, P. Ye, L. Li, and F. Chen, "Evaluating and comparing memory error vulnerability detectors," *Inf. Softw. Technol.*, vol. 137, Sep. 2021, Art. no. 106614, doi: 10.1016/j.infsof.2021.106614.

[17] A. P. Kuruvila, I. Zografopoulos, K. Basu, and C. Konstantinou, "Hardware-assisted detection of firmware attacks in inverter-based cyber-physical microgrids," *Int. J. Electr. Power Energy Syst.*, vol. 132, Nov. 2021, Art. no. 107150, doi: 10.1016/j.ijepes.2021.107150.

[18] P. Sun, Q. Yan, H. Zhou, and J. Li, "Osprey: A fast and accurate patch presence test framework for binaries," *Comput. Commun.*, vol. 173, pp. 95–106, May 2021, doi: 10.1016/j.comcom.2021.03.011.

[19] C. Wright, W. A. Moeglein, S. Bagchi, M. Kulkarni, and A. A. Clements, "Challenges in firmware re-hosting, emulation, and analysis," *ACM Comput. Surv.*, vol. 54, no. 1, pp. 1–36, Jan. 2022, doi: 10.1145/3423167.

[20] A. P. Kuruvila, I. Zografopoulos, K. Basu, and C. Konstantinou, "Hardware-assisted detection of firmware attacks in inverter-based cyber-physical microgrids," *Int. J. Electr. Power Energy Syst.*, vol. 132, no. 10, pp. 1–14, Feb. 2021, doi: 10.1016/j.ijepes.2021.107150.

[21] G. Lin, S. Wen, Q.-L. Han, J. Zhang, and Y. Xiang, "Software vulnerability detection using deep neural networks: A survey," *Proc. IEEE*, vol. 108, no. 10, pp. 1825–1848, Oct. 2020, doi: 10.1109/JPROC.2020.2993293.

[22] Q. Zhao, C. Huang, and L. Dai, "VULDEFF: Vulnerability detection method based on function fingerprints and code differences," *Knowl.-Based Syst.*, vol. 260, Jan. 2023, Art. no. 110139, doi: 10.1016/j.knosys.2022.110139.

[23] X. Zhao, D. Wang, H. Xu, Z. Ding, Y. Shi, Z. Lu, and Z. Cheng, "Groundwater pollution risk assessment based on groundwater vulnerability and pollution load on an isolated island," *Chemosphere*, vol. 289, Feb. 2022, Art. no. 133134, doi: 10.1016/j.chemosphere.2021.133134.
[24] C. Xin, C. Wang, Z. Xu, M. Qin, and M. He, "Marker-free vision-based method for vibration measurements of RC structure under seismic vibration," *Earthq. Eng. Struct. Dyn.*, vol. 51, no. 8, pp. 1773–1793, Jul. 2022.
[25] A. B. Nelson, L. S. Chow, C. C. Hughey, P. A. Crawford, and P. Puchalska, "Artifactual FA dimers mimic FAHFA signals in untargeted metabolomics pipelines," *J. Lipid Res.*, vol. 63, no. 5, May 2022, Art. no. 100201, doi: 10.1016/j.jlr.2022.100201.
[26] T. Liao, X. Zhao, P. Coates, B. Whiteside, Z. Jiang, and Y. Men, "Structural heterogeneity dependence of the fracture feature distribution in the tensile elongation of microinjection molded polyethylene," *Macromolecules*, vol. 56, no. 5, pp. 1983–1994, Mar. 2023, doi: 10.1021/acs.macromol.3c00010.
[27] J. Li, P. Li, X. Hu, and K. Yu, "Learning common and label-specific features for multi-label classification with correlation information," *Pattern Recognit.*, vol. 121, Jan. 2022, Art. no. 108259, doi: 10.1016/j.patcog.2021.108259.

PENG LIU was born in Quwo, Shanxi, in August 1984. He received the master's degree in high voltage and insulation technology from Xi'an Jiaotong University, in 2010.

Since 2022, he has been a Secretary and the Deputy Manager of the Party Branch (Data Center), State Grid Ningbo Power Supply Company's Xintong Branch. In 2022, he was the deputy Editor-in-Chief, and has published a monograph entitled "Design and Application of Electric Power Big Data Algorithms" (Xi'an Jiaotong University Press). His research focuses on electrical engineering and power informatization.

YASU CAO was born in Ningbo, Zhejiang, in September 1994. She received the bachelor's degree in automation from Zhengzhou University, in 2016. Since 2016, she has been with State Grid Ningbo Supply Company.

YUJUN YAN was born in Ningbo, Zhejiang, in September 1980. He received the master's degree in engineering in the field of computational technology from Zhejiang University of Technology, in 2016. Since 2002, he has been with State Grid Ningbo Power Supply Company. He has published six papers and three works.

YONG WANG was born in Xiaoshan, Zhejiang, in September 1984. He received the bachelor's degree in software engineering from Zhejiang University of Technology, in 2007. Since 2007, he has been with State Grid Ningbo Power Supply Company. He has published two papers and two works.