A Compiler Fingerprint Extraction-oriented Approach to Binary File Analysis

XinHong Hei[1,2] , YiLei Yao[1,2] , YiChuan Wang[1,2]* , JinPei Yan[1,2] , WenJiang Ji[1,2] , Lei Zhu[1,2] , YanNing Du[1,2]

Xi'an University of Technology School of Computer Science and Engineering
Shaanxi Key Laboratory for Network Computing and Security Technology

ABSTRACT

By extracting fingerprint characteristics of the GCC (GNU Compiler Collection) compiler, we can classify the GCC compiler which is invaded by malicious code.In this paper, we propose a binary file-based GCC compiler detection system, called T-GCC. The assembly phase of the GCC compiler yields a binary file from which we can extract the fingerprint features of the GCC compiler. The specific method is as follows: Visualize the binary file obtained by compiling the GCC compiler as a grayscale image, and then extraction of texture features GLCM-5 based on the gray level co-occurrence matrix and classify by back propagation neural network (BP) model. The accuracy of classification of two GCC versions of compilers by this model reached 99%, indicating that the method in this paper can effectively extract the fingerprint characteristics of the GCC compiler, and this method has a good effect on the future detection of whether the GCC compiler has been invaded by malicious code.

KEYWORDS

binary file, texture feature extraction,the GCC compiler security

1 Introduction

In recent years, more and more attackers have begun to focus on the software supply chain[1]. The software supply chain refers to the entire process of software from the software manufacturer to user download and use, including software source code writing, source code compilation, software distribution, software download and software update. Software supply chain attack refers to that the attacker attacks the security holes existing in each link of the software supply chain, and penetrates backward along the supply chain aiming at the trust relationship in the supply chain without conducting the traditional software security inspection[2], so as to achieve the penetration and attack on the target network. The traditional software attack is to implant malicious code by using the vulnerability of the developed software itself.Statistically speaking, the attacks on the software supply chain by attackers have far exceeded the attacks on the traditional software.At present, there is very little work that can realize automatic analysis and detection of whether the software supply chain is polluted. However, frequent attacks on the software supply chain have brought serious problems such as privacy leakage and property security to individuals, enterprises and even countries. In today's complex Internet background, the security detection and prevention of software supply chain has become a research direction that cannot be ignored in network security.

Because people do not know much about the software supply chain security, the defense is weak, and once this attack occurs, only one implant can affect the information security of tens of millions or even hundreds of millions of users. In 1984, K. T. Hompson[3] mentioned that attackers can contaminate all the software compiled and released by attacking a very important software development tool -- compiler, which is called KTH attack. The XcodeGhost incident in 2015 confirmed this[4]. However, people's work is mainly on the study of the senior holes problem, mostly by classifying the malicious code family to analyze the behavior of malicious software and attacks[5], but ignored the simple hole problem of influential -- the compiler security issues, software supply chain security problem has been ignored for too long by our security personnel.If the tools in the software supply chain are attacked, there is a risk that they will explode.

Compiler as an integral part of the software development process[6], once a developer's compiler is implanted with malicious code by attackers, all software compiled by this compiler will be affected afterwards, and these software may pose serious threats to users' privacy and property after being downloaded by users. At present,

people pay less attention to the security of compilers, but compilers are very important in the software development process, so people should also gradually pay attention to the security of compilers. In this paper, we focus on the security of GCC compiler, and extract the fingerprint features of GCC compiler from the binary files obtained after the preprocessing, compilation and assembly process of GCC compiler, and through this classification model, we can find whether the GCC compiler has been implanted with malicious code in time.

2 Related Work

Software feature extraction methods are crucial for software detection and classification, and the current feature extraction methods are static extraction, dynamic extraction and hybrid methods [7]. Static feature extraction approach extracts features by analyzing the source code without running the program, but it is hindered by code obfuscation techniques. The dynamic feature extraction method executes the executable in an isolated environment and extracts features from the memory image of the executable or its behavior, but this method is time consuming and requires a high code runtime environment. Hybrid feature extraction methods combine the features extracted by static feature extraction methods and dynamic feature extraction methods to improve the accuracy of detection results [8].

Since the binary file obtained after the GCC compiler preprocessing, compilation, and assembly process does not go through the linking stage and avoids the possible hazards of running the source code directly, the fingerprint features of the GCC compiler can be well extracted from the binary file. In the static feature extraction technique, features need to be extracted from the code flow. In the dynamic feature extraction approach, the features are extracted during the execution of the binary code and the dynamic detection requires running the code in a specific environment, but the binaries we get are non-executable code. After reviewing a lot of literature, we found that the current binary file-based malicious code code detection methods almost always visualize the binary code as an image for processing [9-13], which describes the similarity between the codes and the file structure through the texture features of the image, which overcomes the problems related to feature selection and extraction and deep learning, [10] by drawing the entropy image corresponding to the binary file to achieve malware classification, [12] converts the attributes of the original malware binary executable into grayscale images,and uses the ResNeXt network model for classification.The method achieves 98.32% and 98.86% classification accuracy on malware dataset and improved malware dataset, respectively, [14] MCSC converts the disassembled malware code into a grayscale image based on SimHash grayscale images and then classifies the malicious code family by convolutional neural network with 98.862% classification accuracy on malware dataset of 10, 805 samples, which shows that the binary code visualization method can extract features well.

At present, there are three main methods for image texture feature extraction: GIST texture feature extraction[15], Gabor filter texture feature extraction[16] and GLCM texture feature extraction[17]. Among them, GIST recognition classification does not need image segmentation and local feature extraction. Compared with local features, this feature is a more "macro" feature description, ignoring the local features of the picture, while different GCC compilers are modified only a certain part of the features, which are generally more similar. Gabor filter texture feature extraction method, implementation is more complex.

3 Fingerprint Feature Extraction of GCC Compiler Based on Binary Files

3.1 System architecture

This paper studies the feature extraction method of GCC compiler based on machine learning.The method mainly includes 5 steps, C language source code data collection, C language source code collected by 8.1.0 version and 8.3.1 version of the GCC compiler through the command "GCC -- C *. C -- O *. O" compiled into binary files, binary file preprocessing, BP model construction and training, BP model test.The system framework of this experiment is shown in Figure 1.

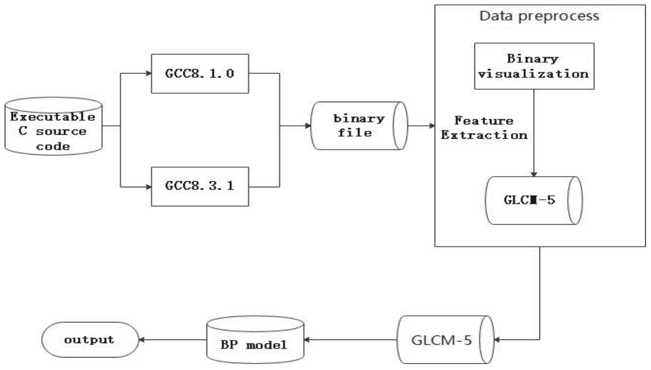


Fig 1 System architecture diagram

3.2 Data Preprocessing

Before using BP neural network to extract features, the binary file data set needs to be preprocessed first. In this paper, the binary file data set preprocessing stage mainly includes two steps: B2M mapping binary files into grayscale images and GLCM texture feature extraction. B2M mapping is to convert binary files compiled by GCC compiler into grayscale images, and GLCM texture feature extraction is to extract GLCM-5 features from grayscale images obtained by mapping.

3.2.1 Binary Visualization

Binary files are files stored in binary format, which are machine-readable languages. Human beings cannot read information directly from binary files, but the information contained in binary files does exist. This paper analyzes the binary files compiled by different versions of the GCC compiler to detect the GCC compiler, so the study of binary files is of great significance to extract the characteristics of the GCC compiler in this experiment.

In this experiment, B2M algorithm is used to map binary files to grayscale images[18].The B2M algorithm reads the binary file every 8bit, and each 8bit binary character stream can get an unsigned integer number in the range of 0-255. Each unsigned integer number exactly corresponds to a pixel of the gray image. The integer number read out from the target file is formed into a one-dimensional vector according to a fixed length.Then these one-dimensional vectors are formed into two-dimensional vectors in order, and finally the two-dimensional array can be visualized as a grayscale image in JPG format without compression.The process of B2M algorithm is shown in Figure 2.

The height of the image is determined by the size of the target file, and the product of the width and height is the size of the target file.Width Settings of grayscale images determine whether texture features of subsequent images can be extracted well. Table 1 lists the optimal widths corresponding to different file sizes[19]. Figure 3 is the grayscale image of the binary file compiled by the 8.1.0 GCC compiler and the binary file compiled by the 8.3.1 GCC compiler mapped by the B2M algorithm.

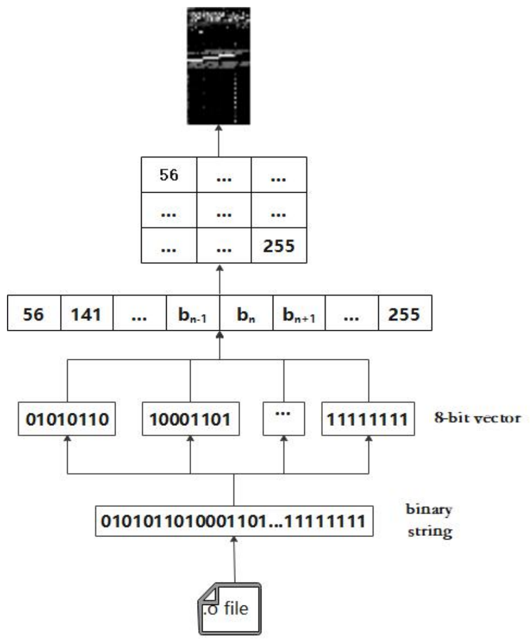


Fig 2 Binary visualization

Table 1 Width Set According to the Binary File Size

file size	image width
<10KB	32
10KB-30KB	64
...	...
1000KB-2000KB	1024
2000KB-4000KB	1280

（a）Version 8.1.0 **(b) Version 8.3.1**

Fig 3 Grayscale of Version 8.1.0 and 8.3.1

3.2.2 Grayscale Image Feature Extraction

In this paper, we adopt the texture feature extraction method based on GLCM (Gray-Level Co-occurrence Matrix) to extract the texture features[20] of the grayscale image obtained in subsection 3.2.1, and this process mainly contains three steps: downscaling the grayscale image, calculating the GLCM of the downscaled grayscale image and feature extraction of GLCM.

GLCM is a method to describe the texture of an image by studying the spatial correlation properties of grayscale, describing the joint distribution between two pixels with some spatial relationship, which can reflect the comprehensive information of image grayscale about the direction, adjacent interval and the magnitude of change. GLCM is obtained by counting the probability of two pixels with grayscale levels i and j in a grayscale image in a fixed direction θ (00, 450, 900, 1350), and the distance between i and j is d. And finally, we can get a GLCM of size M*M (M is the highest gray level of the grayscale image) in each direction. Figure 4 shows the GLCM of a grayscale image with the highest gray level of 3 (M=3) in four directions. Due to the sequential execution of the code, only the horizontal direction (θ=00) is selected to calculate the GLCM. In order to reduce the computational effort of the next feature extraction process, this paper first reduces the highest gray level of the obtained gray map to 64, and then calculates the GLCM of the downgraded gray map. Haralick [21-22] et al. proposed the description of image textures using grayscale covariance matrix in the early 1970s and extracted 14 features from them. But some of these 14 features have strong correlation inside. After selection, five of these features, ASM (Angular second moment), Con (ContrastCon), Cor (Correlation), Ent (Entropy), and Idm (Dissimilarity), which have a strong influence on this experiment, are selected in this paper.

ASM reflects the uniformity of image gray distribution and texture thickness. If the element values of GLCM are similar, the energy is small, indicating that the texture is meticulous.If some of these values are large and others are small, then the energy value is large.Large energy values indicate a more uniform and regular texture pattern.The calculation formula is as shown in Formula (1). a is the size of GLCM, and P_{ij} is the value of row i and column j in GLCM.μ_x is the mean of the rows, μ_y is the mean of the columns, σ_x is the standard deviation of the rows, and σ_y is the standard deviation of the columns.

$$ASM = \sum_{i=0}^{a-1}\sum_{i=0}^{a-1} P(i,j)^2 \tag{1}$$

Con reflects how the values of the matrix are distributed and how much local variation there is in the image, reflecting the sharpness of the image and the depth of the grooves in the texture. The deeper the groove of the texture, the greater the contrast, the clearer the effect. On the contrary, if the pair ratio is small, the grooves are shallow and the effect is vague. The calculation formula is shown in Formula (2).

$$\text{Con} = \sum_{i=0}^{a-1}\sum_{j=0}^{a-1} (i-j)^2 P_{ij} \tag{2}$$

Idm reflects the homogeneity of image texture and measures the local change of image texture.If its value is large, it indicates that there is no change between different areas of the image texture, and the local area is very uniform. The calculation formula is shown in Formula (3).

$$\mathrm{Idm} = \sum_{i=0}^{a-1} \frac{P_{ij}}{1+(i-j)^2} \tag{3}$$

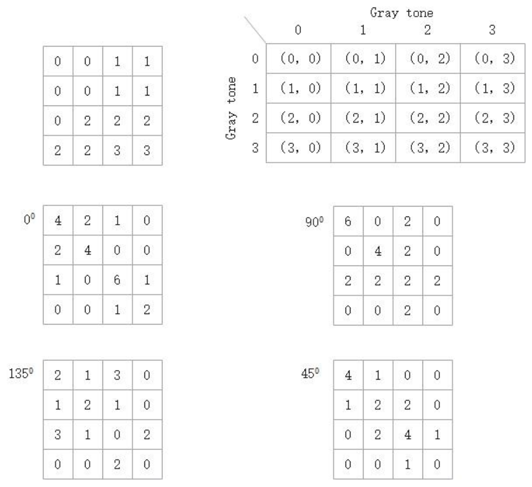
Ent is a random measure of the amount of information an image contains.When all values in GLCM are equal or pixel values show maximum randomness, Ent is the largest.Therefore, Ent value indicates the complexity of image gray distribution. The larger the entropy value is, the more complex the image is. The calculation formula is shown in Formula (4).

$$\mathrm{Ent} =- \sum_{i=0}^{a-1}\sum_{j=0}^{a-1} P_{ij} \lg P(i,j) \tag{4}$$

Cor is used to measure the similarity degree of image gray level in the direction of row or column, so the size of value reflects the local gray level correlation. The larger the value, the greater the correlation.The calculation formula is shown in Formula (5).

$$\mathrm{Cor} = \sum_{i=0}^{a-1}\sum_{j=0}^{a-1} \frac{(ij)P_{ij} - \mu_x\mu_y}{\sigma_x\sigma_y} \tag{5}$$

In this paper, these five features are collectively referred to as GLCM-5, and the feature extraction process is shown in Figure 5.Figure 6 shows the binary grayscale image of a source file compiled by the 8.1.0 version GCC compiler and the GLCM-5 extracted by the GLCM texture feature extraction method.



0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

Gray tone	0	1	2	3
0	(0, 0)	(0, 1)	(0, 2)	(0, 3)
1	(1, 0)	(1, 1)	(1, 2)	(1, 3)
2	(2, 0)	(2, 1)	(2, 2)	(2, 3)
3	(3, 0)	(3, 1)	(3, 2)	(3, 3)

0^0				
	4	2	1	0
	2	4	0	0
	1	0	6	1
	0	0	1	2

90^0				
	6	0	2	0
	0	4	2	0
	2	2	2	2
	0	0	2	0

135^0				
	2	1	3	0
	1	2	1	0
	3	1	0	2
	0	0	2	0

45^0				
	4	1	0	0
	1	2	2	0
	0	2	4	1
	0	0	1	0

Fig 4 GLCM in four directions

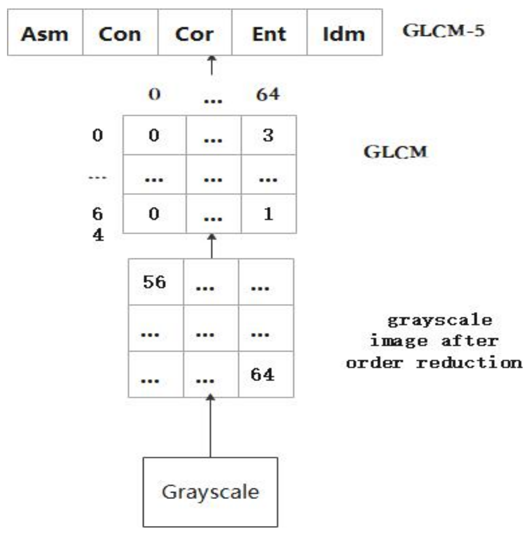
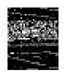


Fig 5 Grayscale image feature extraction



```
ASM: 0.0485089162049865
Con:148.065789473684
Ent:4.24211629578683
Idm:0.301672110300147
Cor:7.67105263157894
```

Fig 6 Grayscale and GLCM-5

3.3 BP Neural Network

After feature extraction in Section 3.2, BP neural network is used to extract software features of GCC compiler.BP neural network is a kind of forward feedback neural network, which is composed of input layer, hidden layer and output layer. By calculating the difference between the output layer and the expected output, the network parameters are constantly adjusted to make the error smaller, so as to make the output of BP network and the expected output closer and closer.The BP network topology used in this paper is shown in Figure 7.

The input layer has 5 neurons, the hidden layer has 11 neurons, and the output layer has 2 neurons. The tanh function is used as the activation function of the hidden layer, and the sigmoid function is used as the activation function of the output layer. where x1, x2, x3, x4, x5 is the input of the neural network, i.e. the eigenvalue GLCM-5 extracted in Section 3.2.2. Y1 and Y2 are the output of the network, which Determine which version of the GCC compiler assembles the detected binaries. W_{ij} and W_{jk} are the weights of the network, using Crossentropy $\mathrm{Loss\ C} =- \frac{1}{n}\sum_x [yln a + (1-y)\ln(1-a)]$ As a loss function, where n is the number of samples in the training set, y is the true label, a is the output of the neural network, and x is the input sample. Figure 8 is the accuracy diagram during the training process of the BP neural network model in this experiment, and Figure 9 is the variation diagram of the loss function during the training process.

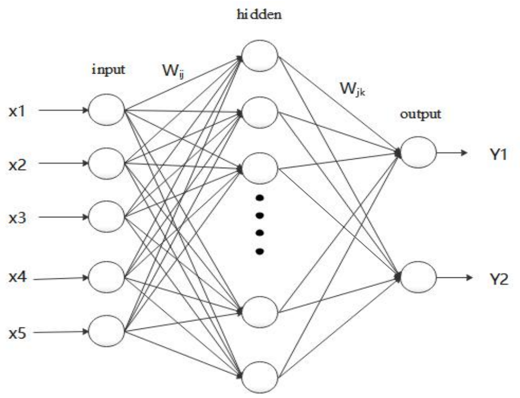


Fig 7 BP model

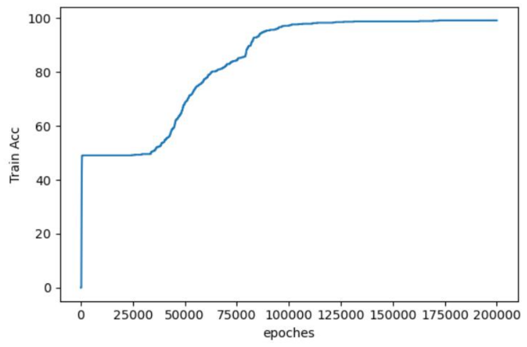


Fig 8 The variation of accuracy during the training process

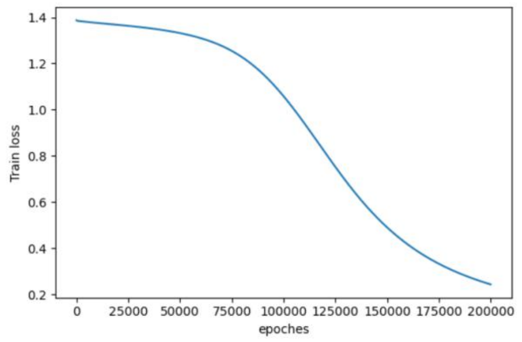


Fig 9 The variation of the cross entropy loss function during training process

3.4 Model Test

The data set obtained in this paper is divided into training set and test set according to the ratio of 8:2, and the test result of the final classification model can reach 0.99. It can be seen that the method proposed in this paper has a good effect on the detection of GCC compiler. Figure 9 shows the classification result of test set.

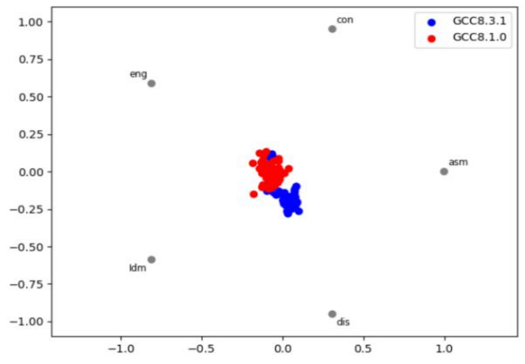


Fig 9 The classification of the test set

4 Model Analysis

4.1 Reasons for Feature Extraction

Since compiler optimization is the main reason for the syntax differences of binary files [23], the differences of compilers can be found by analyzing and detecting binary files in this paper. Because the binary file is C language source code after the GCC compiler pretreatment, compilation and assembly process, without a link stage, to avoid the direct operation of the source code may bring harm, so from the binary file can be very good extraction of the fingerprint characteristics of the GCC compiler without causing harm. Although the binary file is unreadable to human beings, the information contained in it is real.The object file is mapped into a grayscale image through the B2M algorithm.

4.2 Complexity Analysis

The scale of the grayscale image converted from binary file is h*w, h is the height of the grayscale image, and w is the width of the grayscale image. Since the grayscale of the grayscale image is M, the obtained GLCM size is M*M. In order to reduce the algorithm complexity, the gray level of the grayscale image needs to be reduced to a (a<M). The process of texture feature extraction algorithm based on GLCM is mainly divided into three steps, from the grayscale image with the gray level of M to a, from the degraded grayscale image to calculate GLCM and from GLCM to calculate the feature vector. The time complexity of degrading grayscale image is O(hw), the time complexity of calculating GLCM is O(a2), and the time of calculating eigenvector from grayscale co-occurrence matrix is O(a2). Therefore, the time complexity of the algorithm using gray level co-occurrence matrix to extract grayscale image texture features is O(HW + 2a2).

5 Results and Discussion

5.1 Experimental Data and Experimental Setup

The Dataset of this paper is 502 normally compiled C language source code downloaded from the SARD (Software Assurance Reference Dataset). After compilation, 1004 binary files can be obtained, including 502 of the GCC8.1.0 version.GCC8.3.1 version 502, the data set is divided into two subsets, 80% for training, 20% for prediction.
The experimental environment is the GCC compiler version 8.3.1 under the CentOS 8.1.1911 operating system, and the GCC compiler version 8.1.0 under the Windows 10 system.Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz 2.80GHz, 8.00GB RAM, 1T hard disk.

5.2 Experimental Result and Comparisons

In this paper, the following evaluation indicators are used to evaluate the experimental results. Table 2 shows the meanings of each parameter. F-Measure is used to measure the accuracy of the model and is the harmonic mean of precision and recall rate, which can well reflect the classification performance of the model.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F} - \text{Measure} = \frac{2 * Precision * Recall}{Precision + Recall}$$

Table 2 Variable Table of Evaluation Indicators

symbol	meaning
TN	The number of normal applications that are accurately classified
FP	The number of normal applications that are misclassified
TP	The number of malicious applications that are correctly classified
FN	The number of misclassified malicious applications

In this part we will this experiment with some existing similar to the experimental methods of malware feature extraction and detection method [12, 14], by can be seen in table 3, is mainly used for the detection of the malicious code to feature extraction, the malicious code itself there was no attention to the GCC compiler safe environment for software development,This further demonstrates the advancement and effectiveness of our experiment.

Table 3

Malware Detection Models	ACC	F-Measure	GCC test
ResNeXt	0.98	-	-
MCSC	0.98	0.98	-
Our method	**0.99**	**0.99**	**Yes**

6 Conclusion

Through this experiment, we find that the compilation process of the compiler has a great relationship with the system environment. The compilation part and the static link part of the compiler are system programs running on the operating system. The features automatically extracted by machine learning in this experiment mainly come from the difference caused by the system environment to the compilation process of the compiler.In the future, I hope to find better test cases. Feature extraction for different versions of compilers under the same system can find the features optimized by the compiler.

ACKNOWLEDGMENTS

This research work is supposed by the National Joint Funds of China (U20B2050), National Key R&D Program of China(2018YFB1201500), National Natural Science Founds of China (62072368, 61773313, 61702411), Key Research and Development Program of Shaanxi Province (2020GY-039, 2021ZDLGY05-09, 2017ZDXMGY-098, 2019TD-014).

REFERENCES

[1]Xixun H E , Zhang Y , Liu Q. 2020. Survey of Software Supply Chain Security Vol. 5. Xidian University.
DOI: https://doi.org/10.19363/J.cnki.cn10-1380/tn.2020.01.06
[2] Mohanty, S. , Urciuoli, L. , Boekesteijn, E. G. , & Hintsa, J. . 2014. The resilience of energy supply chains: a multiple case study approach on oil and gas supply chains to europe. Supply Chain Management.
DOI: https://doi.org/10.1108/SCM-09-2012-0307
[3]Thompson, K.1984. Reflections on trusting trust. Communications of the ACM, 27(8), 761-763.
DOI: https://doi.org/ 10.1145/358198.358210
[4] Gui, X. , Liu, J. , Chi, M. , Li, C. , & Lei, Z. . 2016. Analysis of malware application based on massive network traffic. China Communications, 13(8), 209-221.
DOI: https://doi.org/10.1109/CC.2016.7563724
[5] Xiao, G. , Li, J. , Y Chen, & Li, K. . (2020). Malfcs: an effective malware classification framework with automated feature extraction based on deep convolutional neural networks. Journal of Parallel and Distributed Computing, 141.
DOI: https://doi.org/ 10.1016/j.jpdc.2020.03.012
[6] Ullman, & D Jeffrey. 1977. Principles of compiler design. Addison-Wesley Pub. Co.
DOI: https://doi.org/ 10.1109/MC.1982.1654000
[7] Xiao, G. , Li, J. , Y Chen, & Li, K. . 2020. Malfcs: an effective malware classification framework with automated feature extraction based on deep

convolutional neural networks. Journal of Parallel and Distributed Computing, 141.
DOI: https://doi.org/ 10.1016/j.jpdc.2020.03.012

[8] Abusitta, A. , Li, M. Q. , & Fung, B. C. M. . 2021. Malware classification and composition analysis: a survey of recent developments. Journal of Information Security and Applications, 59.
DOI: https://doi.org/10.1016/j.jisa.2021.102828.

[9]Q. Wei, S. Xiao and L. Dongbao, 2019. "Malware Classification System Based on Machine Learning," 2019 Chinese Control And Decision Conference (CCDC) pp. 647-652.
DOI: https://doi.org/10.1109/CCDC.2019.8832802

[10] Ap, A. , Aml, A. , Vp, B. , Cav, C. , An, A. , & As, A. , et al. 2021. Malware detection employed by visualization and deep neural network. Computers & Security.
DOI: https://doi.org/ 10.1016/j.cose.2021.102247

[11] HD Menéndez, Bhattacharya, S. , Clark, D. , & Barr, E. T. . 2019. The arms race: adversarial search defeats entropy used to detect malware. Expert Systems with Applications, 118(MAR.), 246-260.
DOI: https://doi.org/10.1016/j.eswa.2018.10.011

[12] Jin, H. G. , Jan, T. , Mohanty, M. , Patel, O. P. , & Prasad, M. . 2020. Visualization Approach for Malware Classification with ResNeXt. 2020 IEEE Congress on Evolutionary Computation (CEC). IEEE.
DOI: https://doi.org/10.1109/CEC48606.2020.9185490.

[13] Fu, J. , Xue, J. , Wang, Y. , Liu, Z. , & Shan, C. . (2018). Malware visualization for fine-grained classification. IEEE Access, 14510-14523.
DOI: https://doi.org/ 10.1109/ACCESS.2018.2805301

[14] Ni, S. , Qian, Q. , & Zhang, R. . 2018. Malware identification using visualization images and deep learning. Computers & Security, 77(AUG.), 871-885.
DOI: https://doi.org/ 10.1016/j.cose.2018.04.005

[15] [1]Neha Gour & Pritee Khanna.2020.Automated glaucoma detection using GIST and pyramid histogram of oriented gradients (PHOG) descriptors. Pattern Recognition Letters(),. doi:10.1016/j.patrec.2019.04.004.
DOI: https://doi.org/10.1016/j.patrec.2019.04.004

[16] Lu, W. , Fan, Z. , Wei, L. , Xie, X. M. , & Wei, H. . 2015. A method of sar target recognition based on gabor filter and local texture feature extraction. Journal of Radars.
DOI:info:doi/10.12000/JR15076

[17] R A Saputra,Saputra R A,Suharyanto,Wasiyanti S,Saefudin D F,Supriyatna A & Wibowo A.2020.Rice Leaf Disease Image Classifications Using KNN Based On GLCM Feature Extraction. Journal of Physics: Conference Series(1),.
DOI: https://doi.org/10.1088/1742-6596/1641/1/012080

[18] Vv, A. , Skm, A. , & Vbs, B. . 2020. Multiclass malware classification via first- and second-order texture statistics. Computers & Security, 97.
DOI:https://doi.org/10.1016/j.cose.2020.101895

[19] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath. 2011. Malware images: visualization and automatic classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security (VizSec '11). Association for Computing Machinery, New York, NY, USA, Article 4, 1–7.
DOI: https://doi.org/10.1145/2016904.2016908

[20] Zhang, Y. , Ren, W. , Zhu, T. , & Ren, Y. . (2019). Saas: a situational awareness and analysis system for massive android malware detection. Future Generation Computer Systems, 95(JUN.), 548-559.
DOI: https://doi.org/10.1016/j.future.2018.12.028

[21] Haralick, & R., M. . 2005. Statistical and structural approaches to texture. Proceedings of the IEEE, 67(5), 786-804.
DOI: https://doi.org/ 10.1109/PROC.1979.11328

[22] Haralick, R. M. , Shanmugam, K. , & Dinstein, I. . 1973. Textural features for image classification. Studies in Media and Communication, SMC-3(6), 610-621.
DOI: https://doi.org/ 10.1109/TSMC.1973.4309314

[23] Ren, X. , Ho, M. , Ming, J. , Lei, Y. , & Li, L. . 2021. Unleashing the hidden power of compiler optimization on binary code difference: an empirical study.