Feature-Based Software Integrity Verification and Zero Watermarking Technology

Zhen Chen, De Li, and Hua Jin

***Abstract*—In view of the problem that Portable Executable(PE) files are transmitted in the network, resulting in information loss, duplication and malicious tampering, this paper proposes an integrity check scheme based on PE file structure feature extraction and binder technology. The basic idea is to extract PE file structure features, combined with the bundling technology to achieve PE file integrity verification. In view of the problem that the imitation of software is rapidly spreading, such as the core code segment of the software can be stolen, this paper proposes a software zero watermarking scheme based on PE file special instruction feature extraction. The basic idea is to extract the special features of PE file code segment and logical operation of copyright information to generate zero watermark. The algorithm does not affect the performance and efficiency of the PE file. When the structural integrity of the PE file is damaged, the software exits the operation and outputs the copyright information. By attacking and converting the PE file and testing it with different software, it is proved that the zero watermark algorithm is robustness and credibility.**

***Index Terms*—PE file, feature extraction, bundling technology, integrity verification, software zero watermark.**

I. Introduction

As a kind of digital products, software has high copying rate and great business benefits. The problems of software infringement, piracy, random tampering and malicious attacks are also becoming more and more serious, which seriously hinders the health and sustainability of the software industry. Software watermark is a kind of digital watermark, which is used to carry information of copyright issuers, users, developers and sellers. In software copyright management, it can be used for identity authentication, permission restriction, preventing copyright theft and illegal copying of software, etc. [1], [2]. Software birthmark feature technology, also known as zero watermark or zero knowledge, is an invariant feature or a key feature in the extraction software to achieve identification of the software or the software family [3].

Software feature acquisition techniques include: (1) static-based extraction algorithms, and (2) dynamic dependency graph methods combined with semantic analysis. Tamada *et al*. [4] improved the limitations of the source code, selected the execution code (binary or bytecode), applied the java bytecode set as a software feature; Heewan Park [5] proposed the use of opcodes and their static stacks as software functions for software identification. In order to improve the robustness of this feature, the literature [6]-[8] introduces the Chinese word segmentation n-gram method to extract the state of the dynamic operation runtime to improve the feature.

However, in terms of its development, the study of software features began with the analysis of source code features [9], [10]. At present, there are few zero watermarking algorithms based on Portable Executable (PE) files. The algorithm extracts PE file structure features and special instruction features, combines bundling technology and zero watermark technology to achieve PE file integrity verification and copyright protection.

Manuscript received September 17, 2019; revised December 12, 2019. This research project was supported by the Education Department of Jilin Province ("Thirteen Five" Scientific Planning Project, Grant No. JJKH20180898KJ), Jilin Education Science Planning Project ("Thirteen Five" Plan, Grant No. GH170043).

Zhen Chen, De Li, and Hua Jin are with Department of Computer Science, Yanbian University, Yanji, China (e-mail: 2576939243@qq.com, leader1223@ybu.edu.cn, hjin@ybu.edu.cn).

II. Research on Related Technology

A. PE File Overview

The Portable Executable (PE) format is the mainstream executable file format on the Microsoft Win32 platform, meaning a portable executable. Common PE format files include EXE and DLL files. Both use the same PE format. The basic structure of the PE file is shown in the Fig. 1:

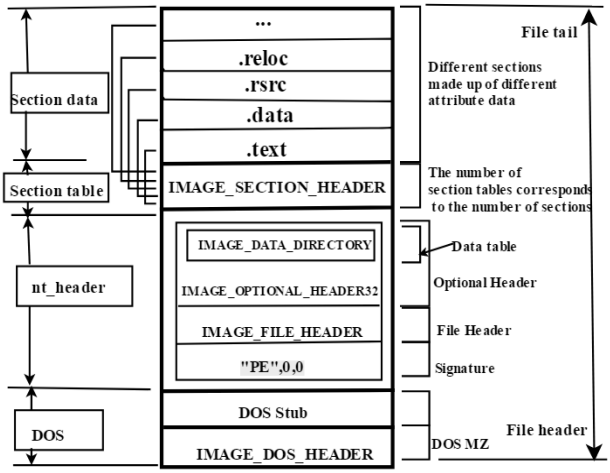

Fig. 1. The basic structure of the PE file.

B. Structural Feature Extraction and Integrity Checking

For a PE file, considering its operational characteristics and structural characteristics, and in order to prevent malicious attacks from modifying the original file, some structural information is particularly important. The following describes the structural characteristics of the PE file and the structural feature extraction method and integrity check.

DOI: 10.7763/IJCTE.2020.V12.1261

1) Extraction of PE file structure features

a) Extraction of DOS_header structure features

There are many fields defined by this structure, but most of them are designed to be compatible with the *DOS* system, which is basically useless in the *Windows* system. There are only two important fields to focus on throughout the structure:

e_magic, it's always equal to hex *0×5A4D*, namely ASCLL string *'MZ'*.

$$\text{If } e\_magic \neq 0 \times 5A4D$$

PE file is invalid

Another important field is the *e_lfanel* field with an offset of *0×3* bytes in hexadecimal. It points to the address where the PE header starts.

b) Extraction of nt_ Header structure features

Nt_header is defined as Fig. 2:

```
typedef struct _IMAGE_NT_HEADERS {
    DWORD Signature;
    IMAGE_FILE_HEADER FileHeader;
    IMAGE_OPTIONAL_HEADER32 OptionalHeader;
} IMAGE_NT_HEADERS32, *PIMAGE_NT_HEADERS32;
```

Fig. 2. The structure of *nt_Header*.

For PE files, Signature is always equal to hex 0×00004550, namely ASCLL string 'PE\0\0'.

$$\text{If } Signature \neq 0 \times 4550$$

PE file is invalid

Next is *IMAGE_FILE_HEADER*. The PE file header defines some basic fields of the PE file. The most important field in this structure is the *NumberOfSections*, distance from the PE header is 6 bytes. This is the number of sections in the file. These fields are used when the PE loader is loaded. If the loader finds that some fields defined in the PE file header do not satisfy the current running environment, the PE will be terminated. So select these fields as the features of the structure.

Next one is *IMAGE_OPTIONAL_HEADER32*. Although it is named optional header, this is necessary for PE files, which contains many important fields related to execution. The structure defines a lot of fields, select some fields as feature values, and the selected fields and meanings are as follows:

(1) *AddressOfEntryPoint*: The Relative Virtual Addresses (RVA) of program entrance.

(2) *BaseOfCode*: The RVA of the start of the code segment.

(3) *BaseOfData*: The RVA of the start of the data segment.

(4) *ImageBase*: The base address of the image.

(5) *SizeOfImage*: The size of the image, the PE file is loaded into the memory space is continuous, this value specifies the size of the virtual space.

(6) *SizeOfCode*: The length of the code segment, if there are more than one code segments, is the sum of the code segment lengths.

(7) *SizeOfInitiallizedeData*: The length of the initialized data.

c) Extraction of SECTION TABLE structure features

The properties of all sections in the PE file are defined in the *SECTION TABLE*. The *SECTION TABLE* is arranged by a series of structures, each of which is used to describe a section. The order of the structures is the same as that of the sections they describe in the file. The definition of the *SECTION TABLE* is as shown in Fig. 3:

```
IMAGE_SECTION_HEADER STRUCT
    DWORD PhysicalAddress;
    DWORD VirtualSize;
        ...
    WORD NumberOfLinenumbers;
    DWORD Characteristics;
IMAGE_SECTION_HEADER ENDS
```

Fig. 3. The structure of *SECTION TABLE*.

Only the following fields are really useful:

(1)*VitualSize*: The size of the block corresponding to the section table, which is the actual size of the block data before it is aligned.

(2)*VitualAdress*: The address where the block is loaded into memory.

(3)*PointToRawData*: Point out the location of the section in the disk file, which is the offset from the beginning of the file header.

(4)*SizeOfRawData*: The size of the block in the disk.

By relying on the values of the above four fields, the loader can find the data of a section from the PE file. And map it to memory.

2) Integrity check based on bundler technology

Bundling technology is a new type of computer technology. Bundle or attach additional data or an executable program to an executable file, and the program will still run. The role of the binder is to bundle two or more files and run the bundled files to achieve the purpose of running multiple files. This paper combines the bundler technology with the integrity check to achieve PE file integrity check.

C. Feature Extraction and Zero Watermarking Technology

1) Code section special instruction feature extraction

The principle of selecting the feature value of the code section is as follows: (1) If the code of the signature code has a special constant and the constant is not the memory address of the target program, it is preferentially selected as the signature. (2) If the signature has a code for the structure or class variable, it can be selected. (3) If the signature position, without the above code, special instructions can be used, and the code without absolute address is the signature. (4) The feature code should be made as short as possible, thereby reducing the complexity of time and space in the detection process and improving the detection efficiency. (5) It should not be too common, so if you choose an overly universal code as a signature, the probability of false positives is very high. (6) Guarantee the uniqueness of the feature code and cannot be repeated.

The call instruction is a jump instruction. When the jump instruction is executed, two steps are performed: (1) pushing the current execution position of the program onto the stack; (2) transferring to the called subroutine. As shown in Fig. 4,

these subroutines are special instructions without special addresses. They are short and precise, conform to the principle of feature value selection. Therefore, select the subroutine pointed to by the call instruction as the code segment feature value.

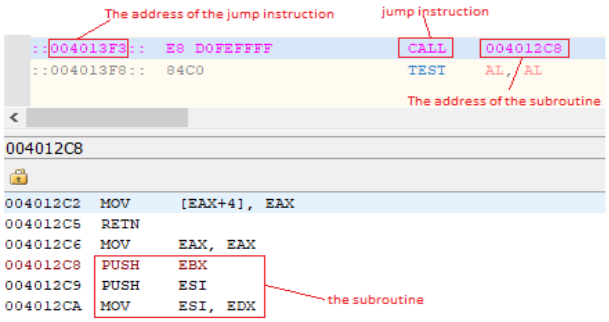

Fig. 4. Subroutine pointed to by the CALL instruction.

2) Zero watermarking technology

The idea of zero watermark is mostly used for image watermarking, which refers to a watermark embedding method that constructs image copyright information through image features without modifying image information. The software zero watermark, assuming that the software program is P, the watermark information W is obtained by the watermark generation algorithm and related information. After being attacked, you can still get the W through the watermark generation algorithm. The algorithm extracts special instructions from the code segment of program P, takes the disassembled binary number as a feature, and performs an exclusive OR operation on the copyright information to obtain W.

3) Robustness and credibility test of zero watermark generation algorithm

Robustness and credibility are two important indicators for evaluating the characteristics of software birthmarks. Robustness is the ability to measure the identity of two identical software; credibility is the ability to distinguish between two different software.

The chi-square test（ χ^2 test）is used to compare the correlation analysis of two or more samples, and compares the degree of agreement or the degree of fit between the theoretical frequency and the actual frequency.

When unknown software is tested, the fit of its characteristic contour to the known software birthmark characteristics is calculated. The whole process can be divided into two steps. 1). Calculate the threshold; 2). Chi-square test.

1) Calculate the threshold: that is, calculate the mean value of the chi-square value between the feature a obtained by the equivalent sample and the feature b of the original sample, as in formula (1), as the software detection threshold:

$$\xi = \sum_{i=1}^{n} \frac{(P_0 - P_i)^2}{P_i} \quad (1)$$

2) Chi-square test: Calculate the chi-square test value χ_{test} of the software feature to be tested and the existing software birthmark feature frequency, compare it with the detection threshold ξ, which is smaller than the detection threshold, indicating that the test program belongs to the same software version, otherwise not a similar version.

III. Feature-Based Software Integrity Check and Zero Watermarking Algorithm

With the continuous popularization of computer and network technologies in various industries, different versions of software are also emerging in the network environment, especially for commercial applications such as configuration software, etc. Software products are transmitted in the network. On the one hand, information may be lost or duplicated due to objective factors such as network lines. On the other hand, an attacker may maliciously tamper with software information by means of interception or forwarding. The algorithm extracts the structural characteristics of the protected software, performs the MD5 operation as the basis for input to the integrity check part, and bundles the integrity check program with the protected software. When the software is attacked by tampering, the software exits the operation and displays the software copyright.

The software shows its rich commercial profit, the software imitation spreads rapidly, the core code segment of the software is stolen, and other parts such as data segments and resource segments are replaced, which brings difficulty to the software identification. The need for pseudo-authentication is also becoming more and more urgent. The algorithm extracts the special instruction feature of the PE file code segment, and is used to identify the core code section of the PE file, and performs an exclusive OR operation with the preprocessed watermark information, and stores it as zero watermark authentication information in the zero watermark information database.

A. Integrity Check and Zero Watermark Generation Algorithm

The flow chart of integrity check and zero watermark generation algorithm is shown in Fig. 5. The specific algorithm includes the following steps:

Step1 Read the carrier PE file;

Step2 Extracting the structural features of the PE file;

1) Define the DOS structure *DOS header*, read the *e_magic* and *e-lfanew* field under *DOS header*;

2) Define the PE header *nt_header*, read the *Signature* field under *nt_header*;

Define the *FileHeader*, read the *NumberOfSections*;

Define the *OptionalHeader*, read the following fields:

1	AddressOfEntryPoint
2	BaseOfCode
3	BaseOfData
4	ImageBase
5	SizeOfImage
6	SizeOfCode
7	SizeOfInitializedeData

3) Define the PE file section table *SECTION_header*,

traversing each section, read the following fields:

1	VitualSize
2	VitualAddress
3	PointToRawData
4	SizeOfRawData

Step3 The field read by Step 2 is used as the feature T_1, do it With MD5. Record the result of the operation as $Info_1$, it is a string of hexadecimal data, saved it in the integrity check section;

Step4 Binding the carrier PE file to the integrity check portion;

1) Define the carrier file, denotede as *myself , get the path $\&ST_1$ of *myself, read the length $size_1$ of *myself.

$$\text{if } size_1 = 0 \text{, report errors;}$$

2) Create a composite file. Use the malloc function to allocate a $size_1$ length space for *myself, open *myself by $\&ST_1$, write the contents of *myself ;

3)Define the integrity verification file, denotede as *heself, get the path $\&ST_2$ of *heself, read the length $size_2$ of *heself.;

4) Use the malloc function to allocate a $size_2$ length space for *heself in the composite file, write the contents of *heself. Create the final composite file, record *out;

Step5 The copyright information C is encrypted by using the DES algorithm to form a binary watermark information W_1, and the watermark length $L(W_1)$ is recorded; the input 64-bit plaintext is replaced according to the replacement rule. Information database *IPR*.

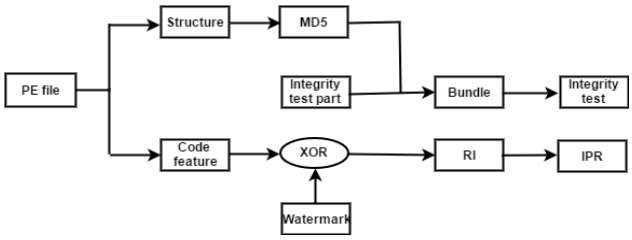

Fig. 5. The flow chart of integrity check and zero watermark generation algorithm.

L_0 is the first 32 bits of the replaced data;

R_0 is the replaced data.

After the 32-bit, each iteration of the encryption process can be expressed as shown in (2):

$$\begin{aligned} L_n &= R(n-1) \\ R_n &= L(n-1) \oplus f(R_{n-1}, K_{n-1}) \end{aligned} \tag{2}$$

The function consists of a four-step operation:

1) K_n generation;

2) extended permutation;

3) *S*- box substitution;

4) *P*- permutation.

Step6 Read the subroutine pointed to by the appropriate number of jump instructions (CALL instruction) from the beginning of the PE file code segment, and disassemble it by c32asm. Record the result of the disassembly as T_2, it is a string of hexadecimal data, and record the length $L(T_2)$ of T_2;

Step7 Calculation $L(T_2) - L(W_1)$, add the same number of 0 after W_1 to get W_2;

Step8 Calculation $T_2 \oplus W_2$, save the result as zero watermark authentication information *RI* in the zero watermark Watermark Extraction and Tamper Detection Algorithm

The flow chart of watermark extraction and tamper detection algorithm is shown in Fig. 6. The specific algorithm includes the following steps:

Step1 Read the PE file to be detected;

Step2 Extract the structural feature values of the PE file to be detected, and encrypt it with the MD5 algorithm to obtain $Info_2$;

Step3 Run the PE files to be tested, release the final composite files and run them at the same time.

1) Define the final composite file *out, create a new file *temp _ exe $_2$, and position the file pointer to the end of *out;

2) Read the length $size_2$ of the integrity check part *heself , read the contents of *heself and write it into *temp _ exe $_2$;

3) Close the handle of file *temp _ exe $_2$;

Step4 Enter $Info_2$ into the integrity check section and compare $Info_1$ and $Info_2$:

$$\begin{gathered} \text{if } Info1 = Info2 \\ \text{the PE file runs normally;} \\ \text{if } Info1 \neq Info2 \end{gathered}$$

the PE file exits, prompting for copyright information;

Step5 Reading the subroutine pointed to by the jump instruction (CALL instruction) of the PE text segment, and disassembling to form a binary feature sequence T_3;

Step6 Perform χ^2 test for T_2 and T_3.

$$\begin{gathered} \text{if } \chi^2 > \xi \\ \text{the software to be tested is not infringed;} \\ \text{if } \chi^2 < \xi \\ \text{read } RI \text{ from IPR, } T_3 \oplus RI \text{ , get } W_3 ; \end{gathered}$$

Step7 W_3 is decrypted by *DES* and compared with copyright information *C* to authenticate copyright.

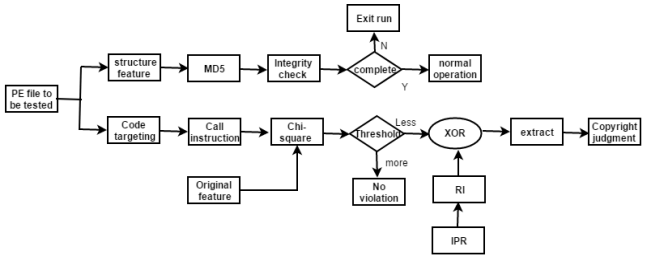

Fig. 6. The flow chart of watermark extraction and tamper detection algorithm.

IV. Experimental Results and Analysis

A. Experimental Environment and Pretreatment

In order to verify the feasibility of the algorithm, the selected PE file is a PDF reading software gsview, and the simulation experiment is performed under C++, MatlabR2010b and analysis software c32asm.

1) Integrity inspection section

The structural features of DOS_header and nt_header of the PE file are extracted as shown in Fig. 7. The structural feature extraction of the section table is shown in Table I, all the above features are hexadecimal data:

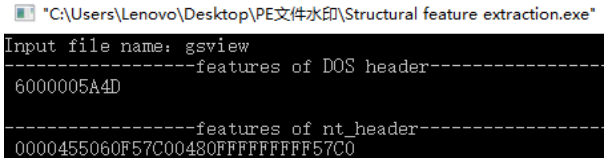

Fig. 7. The structural features of *DOS_header* and *nt_header*.

TABLE I: The Structural Feature Extraction of the Section Table

name	Vitual Size	Vitual Address	PoinTo RawData	SizeOfRawDta
.text	0c64	1000	0400	0e00
.data	0010	2000	1200	0200
.rdada	0220	3000	1400	0400
.bss	0060	4000	0000	0000
.idata	0378	5000	1800	0400
/4	0020	6000	1c00	0200
/35	0a4c	7000	2000	0c00
/47	0282	8000	2c00	0400
/61	01bb	9000	3000	0200
/73	0074	a000	3200	0200
/86	018d	b000	3400	0200
/97	0018	c000	3600	0200

Perform MD5 operation on the feature value, the operation result is hexadecimal: 5172317495DB4D73, input the operation result into the integrity check program, and then bind gsview to the integrity check part, using the icon of gsview, as shown in Fig. 8. After bundling, the software running interface is shown in Fig. 9.

gsview.ico

Fig. 8. The icon of gsview.

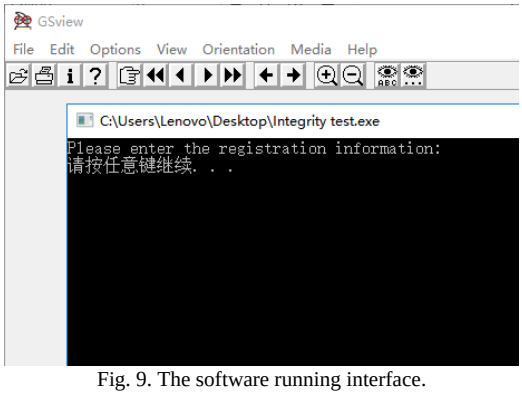

Fig. 9. The software running interface.

2) Zero watermark generation section

Locate the PE file code segment and find the first four jump instruction CALL instructions, as shown in Fig. 10. Taking the first CALL instruction as an example, after the jump, the address is hexadecimal: 004013E0, and the signature is extracted at the subroutine, as shown in Fig. 11.

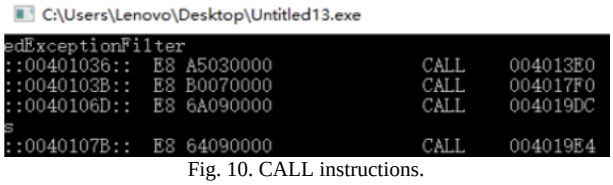

Fig. 10. CALL instructions.

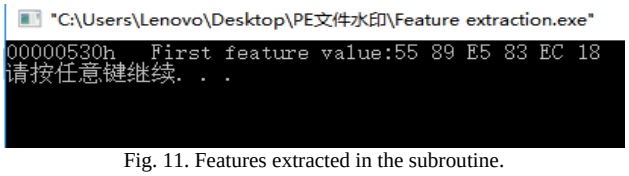

Fig. 11. Features extracted in the subroutine.

If the copyright information is: copyright yanbian, convert it to binary. Extract the subroutine pointed to by the first 4 jump instructions. The eigenvalues are as shown in the following Table II, all of the eigenvalues are hexadecimal.

TABLE II: The Eigenvalues

Serial number	Eigenvalues
1	55 89 E5 83 EC 18
2	55 89 E5 53
3	55 89 E5 DB E3
4	55 89 E5 57

Perform OR operate on eigenvalues and copyright information. The calculation result: 36 E6 95 FA 9E 71 32 E1 91 73 2C E8 B9 8A 34 E7 E5 57. It is stored in the software registration information database *IPR* as the registration information *RI*.

B. Performance Analysis

The evaluation of information hiding techniques for PE files relies on three metrics: performance and efficiency, integrity, robustness, and credibility.

1) Performance and efficiency

Because the copyright certificate does not belong to the normal function of the software, the performance and efficiency of the software will decrease after the watermark information is added to the software. The algorithm extracts the birthmark features of the software and uses it to construct a software zero watermark without embedding information into the carrier, so the performance and efficiency of the PE file does not change.

2) Integrity test

The integrity of the PE file is destroyed by performing the following operations on the PE file:

1) Change the program entry of the PE file;
2) Change the PE file size by adding useless instructions;
3) Use tool PEditor to add or delete a section table;
4) Modify the size of the optional header of the PE file;

5) Modify the base address of the code and the base address of the initialization data;

The result of the operation is shown in Fig. 12. The integrity of the PE file is corrupted, the software exits the run, and the copyright information is displayed.

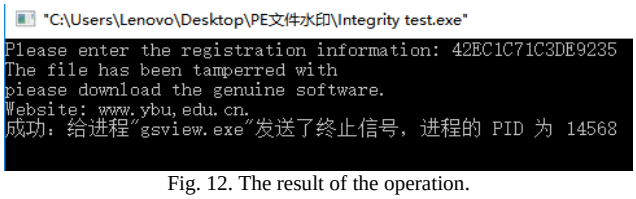

Fig. 12. The result of the operation.

3) Robustness and credibility test of zero watermark generation algorithm

a) Robustness

Let the software P become P_1 after the equivalent semantic transformation (SPT), and if there is $B_P = Extract(P)$, $B_Q = Extract(Q)$, when the similarity value satisfies(3):

$$\chi^2(B_P, B_Q) \le \xi \tag{3}$$

The extraction method Extract is said to be robust to the transformation SPT.

Attack and transform program gsview (tool plus manual simulation), as shown below, get 6 equivalent programs, and perform χ^2 test with the original program, as shown in Table III:

1) Change the software portal;

2) Add instructions to the software: by adding unnecessary code to the PE file without changing the program semantics. For example, add esp, 8 . It may affect the efficiency of program execution, but the semantics of the program will not be changed;

3) Change the size of the software, add or delete PE file section tables, etc. through tool PEditor;

4) Modify the running path;

5) Use tool PEditor to perform an optimization attack;

6) Instruction equivalent replacement means that some instructions or functions of the program are completed by other instructions or compound functions. Many instructions can perform similar functions, and can also perform the functions of another simple instruction by splitting and merging the instructions. For example:

push eip; jmp subAddress and CALL subAddress ,

The two-stage instruction function is equivalent, but the form is different and can be replaced with each other.

After several attacks and transformations on the PE file, the detected is small, indicating that the code segment special instruction feature value extracted by the algorithm can identify the same software after the deformation, so the robustness is better.

TABLE III: CHI-SQUARE TEST

Attack mode	χ^2	Attack mode	χ^2
Change program entry	0	Change size	0
Add instruction	0	Instruction replacement	3.75
Optimization attack	1.5	Modify the running path	0

b) Credibility

There are two different programs P and Q, which are independently developed and use the same feature extraction algorithm.

If there is $Extract(P) \neq Extract(Q)$,

The birthmark extracted by method d is said to have credibility.

Take 20 different programs (maximum, minimum, including desktop applications, music programs, download programs, cryptographic algorithms, user-defined programs, etc.), extract features, perform similarity calculations, and perform test with software. The calculated statistical results are shown in Table IV:

By performing test on different software and the test software, it can be seen from the test results that the value is large, indicatding that the code segment special code feature value extracted by the algorithm can distinguish different softwares, so the credibility is good.

TABLE IV: THE CALCULATED STATISTICAL RESULTS

Software(.exe)	Size(M)	χ^2	Software(.exe)	Size(M)	χ^2
QQMusic	0.215	19	wpscloudsver	0.824	15
DES	0.261	7	Thunder	1.31	10.5
Notebook	0.372	14	BaiduMusic	2.08	17
QyClient	0.457	15.5	CoputerZ_CN	3.77	9.25
WeChat	0.481	16.25	baidunetdisk	8.48	12.25
Xunjie	0.521	8.75	iku_startpic	8.56	12.5
AliIM	0.537	10	WebServe	9.49	13
wow_helper	0.624	11.25	ytbrowser	9.51	11.5
AliTask	0.621	13.5	360ExtLoader	10.79	9
QyKernel	0.753	7.75	CDRAFT_M	10.62	9.75

c) Threshold setting

In this experiment, by attacking and transforming the gsview program, six equivalent programs are obtained.

χ^2 test is performed with the original program, and the maximum is 3.75. Take 20 different programs, extract features, perform similarity calculation, and perform χ^2 test with the original software, the minimum is 7, so the threshold can be set to 5.5. When the suspected object and the genuine software are tested for similarity, χ^2 is less than this threshold. It is necessary to consider that the suspected object infringes the copyright.

V. CONCLUSION

Software products are transmitted in the network. On the one hand, information may be lost or duplicated due to objective factors such as network lines. On the other hand, an attacker may maliciously tamper with software information by means of interception or forwarding. The algorithm extracts the structural characteristics of the protected software, performs the MD5 operation as the basis for input to the integrity check part, and bundles the integrity check program with the protected software. When the software is attacked by tampering, the software exits the operation and displays the software copyright.

In order to prevent the core code segment of the PE file from being stolen, the algorithm extracts the special

instruction feature of the PE file code segment, and is used to identify the core code section of the PE file, and performs an exclusive OR operation with the preprocessed watermark information, and stores it as zero watermark authentication information in the zero watermark information database.

Through the six different attack transformations of the experimental software, the feature extraction algorithm proposed in this paper is proved to be robust. Perform χ^2 test on the features extracted by other software and the features extracted by the experimental software. It is proved that the feature extraction algorithm proposed in this paper has the belief. And thus set the threshold 5.5. When testing the similarity between suspected software and genuine software, it is necessary to consider suspicious object copyright infringement below this threshold.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Zhen Chen designed and implemented experiments, collected and analyzed data, wrote the paper; De Li critically reviewed and guided the intellectual content of the article; Hua Jin received research funding and provided technical or material support; all authors had approved the final version.

References

[1] H. Lim and H. Park, "Detecting theft of java applications via a static birthmark based on weighted stack patterns," *Transactions on Information and Systems*, vol. 91, no. 9, pp. 2323-2332, 2008.

[2] M. D. Preda and M. Pasqua, "Software watermarking: A Seman-based approach," *Electronic Notes in Theoretical Computer Science*, vol. 331, pp. 71-85, 2017.

[3] J. Fan, D. D. Li, and Y. M. Yan, "Research on software features and software watermarking in software protection," *Computer Applications and Software*, vol. 35, no. 12, pp. 298-302, 2018.

[4] Z. Khorsand and A. Hamzeh., "A novel compression-based approach for malware detection using PE header," *Information & Knowledge Technology*, 2013.

[5] E. Raff, R. Zak, and R. Cox, "An investigation of byte n-gram features for malware classification," *Journal of Computer Virology and Hacking Techniques*, 2016.

[6] D. Fleck, A. Tokhtabayev, and A. Alarif, "PyTrigger: A system to trigger & extract user-activated malware behavior," in *Proc. Eighth International Conference on Availability*, 2013.

[7] W. Rui, F. D. Guo, and Y. Yi, "Semantics-based malware behavior signature extraction and detection method," *Journal of Software*, 2012.

[8] Z. Salehi, A. Sami, and M. Ghiasi,"Using feature generation from API calls for malware detection," *Computer Fraud & Security*, vol. 2014, no. 9, pp. 9–18, 2014.

[9] Z. Y. Jie, T. Zhan, and W. Ni, "Evaluation of code obfuscating transformation," *Journal of Software*, vol. 23, no. 3, pp. 700-711, 2012.

[10] R. J. Oentaryo and D. Lo, "Information retrieval and spectrum based bug localization: Better together," in *Proc. Joint Meeting on Foundations of Software Engineering*, 2015.

Zhen Chen received a bachelor's degree in measurement and control technology and instrumentation from China University of Mining and Technology (Beijing). Now he is studying for a master's degree in computer application at Yanbian University. His research interests are software watermarking and information security.

De Li received the Ph.D. in computer science from Xiangming University. He is a professor at Yanbian University, an academic master's tutor and a master's tutor. He is the project leader of the National Natural Science Foundation project and has published many research papers and invention patents.

Hua Jin (Corresponding author) was born in 1970. She is hold with a master degree of engineering. She is engaged in undergraduate and postgraduate teaching work in Yanbian University, College of technology, with the main research direction of database, information security and Internet of things application technology.