A Method for Efficient Malicious Code Detection Based on Feature Codes

Wenshuang Yin, Changcheng Xiang, Dingding Yang, and Shiqiang Chen*
School of Science, Hubei University for Nationalities, China.

* Corresponding author: Shiqiang Chen, Tel.: 18972401015; email: chensq8808@126.com
Manuscript submitted October 31, 2015; accepted January 20, 2016.
doi: 10.17706/jsw.11.9.952-959

Abstract: The malicious code detection can be implemented by the characteristics of the file. Based on the variant feature codes, a method of malicious code detection is proposed in this paper. A file signature library can be set up by using the feature codes of the original data segments, file Message-Digest Algorithm 5 (MD5) features and the types of files. Both match algorithm and extended Aho-Corasick (AC) are employed to identify the signature of wildcard and location information. It can be effectively demonstrated by experimental results that compared with traditional AC algorithm, the accuracy of algorithm is increased by 15.95% and the rate of false positive is reduced by 8.87%. Furthermore, the algorithm proposed in this paper also can identify the features of malicious code variants.

Key words: Pattern matching algorithm, characteristics of the file, malicious code detection, extended AC algorithm.

1. Introduction

With the rapid development of the Internet, more and more malicious codes spread like wildfire on the Internet, which threaten the safety of the Internet application. The attack of malicious codes has reported in [1], which becomes one of the main threats to the Internet security. Up to now, many tools are used to detect the malicious codes, for example, [2], [3], in which the design and implementation are described as follows.

1) The detection methods with the feature codes and behavioral characteristics. A new method has been proposed to identify malicious codes by the behavioral characteristics in [4]-[7]. The automatic extraction algorithm of the feature codes and the feature selection for malware detection has been investigated in [8], [9]. Compared with other detection methods, the detection methods with feature codes are faster and more accurate. However, they have some disadvantages. For example, using a simple mixing method [10] may bypass detection. It should be noted that, different malware needs different feature codes, which makes the virus library too big to handle. Furthermore, the feature codes are obtained from the malicious codes instead of unknown malicious codes. In order to prevent the other users from similar attack when some Internet users had been attacked, one can add some malicious codes to the feature library.

2) Heuristic method. In [11], the heuristic method has been used to identify malicious codes, which gives a way to test whether a program is malicious code in certain rules. It should be pointed out that: characteristics can identify the latest malicious codes, in which the rules are set by people. Therefore, the characteristic is not accurate, the rate of false positive, and non-response are higher than other methods.

3) Access control method. It is only used defensively, for example, under the sandbox and the secure operating system [12], it can protect itself from attacking by binding permission. In [13], the sandbox is

employed to detect malicious codes. However, it does not isolate the running environment of code from the real environment, which restricts some malicious behaviors by reducing the running permission of code. Yet it is worth pointing out that, all the current mainstream browsers support this feature, and a lot of anti-virus soft wares also have the function of the sandbox, which left the suspicious program runs in the sandbox.

4) Virtual machine detection technology is a new testing technology to confine malicious codes. The mechanical technology of anti-virus virtual has been investigated in [14-15], which isolates the running environment of malicious codes from the real environment without being known by the malicious codes. This method helps the host to test malicious codes, and without any damage caused by malicious codes. Because of isolation, the malicious codes would not do harmful effects on the host or other hosts on the Internet. This technology is mostly aimed at the malicious codes which use deformation technology. It is hard to do deformation technology work, and it requires a powerful host. Virtual machine detection is a trend in the future, which has great potential.

This paper proposes an improved scheme and designs a malicious code detection method with various feature codes. Using wildcard in the fragment features of the original codes, it helps identify some variant malicious codes. In order to reduce misjudgment or false positive, the location information is added to specify characteristically the malicious files with record of original fragment features. In addition to the location information, both matching algorithm and extended Aho-Corasick (AC) algorithm are introduced to identify the signatures of the wildcard. A file type recognition feature library is also designed to recognize the type of scanning file, and different scanning strategies are designed for different file types. The experiment is given to show the detection accuracy of the proposed method is improved, and the rate of false positive is reduced. It can identify simple variants of malicious codes and has great application value.

The remainder of this paper is organized as follows. In Section II, We design a feature library for a variety of malicious codes. In Section III, We present a testing scheme for malicious codes. Our experimental results and discussion are given in Section IV. Finally, concluding remarks are given in Section V.

2. Design of Feature Library

2.1. The Feature Library of Original Data Fragments

Every record in feature library of the original data fragments represents a characteristic of a known malicious code, in which virus' name is the name of malicious code and semicolon represents a separator. File type indicates which record of file type goes for. The reason why we add the field is that records of some malicious codes' characteristics are only applicable to specific file. The position where begins to match is the initial position but its previous parts can not be matched. The structure of the original data fragments is shown in Fig. 1.

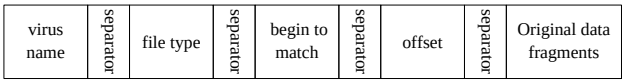

Fig. 1. The record format of characteristics of original data fragments.

Original date fragments are extracted from malicious codes, shown in hexadecimal. It is possible to add wildcard to make the matching process more effective. Table 1 defines wildcard and its description.

Table 1. The Wildcard and Its Description

Wildcard	Description
??	Match any one byte
*	Match any number of bytes

{n}	Match any *n* bytes
{-n}	Match any *n* bytes or less
{n-}	Match any *n* bytes or more
(aa|bb|cc)	Match *aa* , *bb* or *cc*

Sometimes we need useful offset to define concrete location where the characteristics show up and we can not determine the file as malicious codes even if the characteristics show up in other locations. Table 2 is our customized offset descriptors which use EP, SX, and SL to describe the location of executable procedure. The $\{S2+10\}$ means that original data fragments are matched successfully until they must appear in the location of second field of executable procedure and more 10 fields.

Table 2. The Offset Descriptor

Offset	Description
*	Offset any byte
n	Offset any *n* bytes
EOF-n	Offset to the end of file before *n* bytes
EP+n	Offset to the entrance of file ,then add *n* bytes
EP-n	Offset to the entrance of file , then reduce *n* bytes
Sx+n	Offset to *x* section, then add *n* bytes
Sx-n	Offset to *x* section, then reduce *n* bytes
SL+n	Offset to *x* section, then add *n* bytes
SL-n	Offset to *x* section, then reduce *n* bytes

2.2. The Feature Library of Message-Digest Algorithm 5 (MD5) Signature Types

A file name can be randomly modified but its MD5 which cannot be altered on the premise that we do not modify file content. It is hard to find the same MD5 in two files with different content. When it happens, it is called MD5 collision. We can use MD5 and size of files as characteristics of malicious codes' files to reduce MD5 collision. Fig. 2 is the record format of MD5 characteristics. Many records make up the MD5 signature feature library.

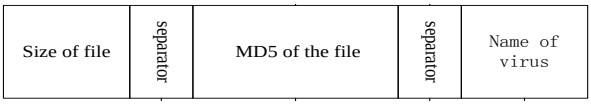

Fig. 2. The record format of MD5 characteristics.

2.3. The Feature Library of File Type

It is necessary to discern file types before detecting malicious codes and defining strategies for file scanning. Wrong file types will seriously influence the scanning results and increase the scanning time. It is not effective to discern files based on file suffixes and some malicious codes can mask themselves. Many characteristic data fragments of malwares are in common file types. Fig. 3 is the record format of original data fragments and the file type. File type feature library collects characteristics of file types in the existing operation system.

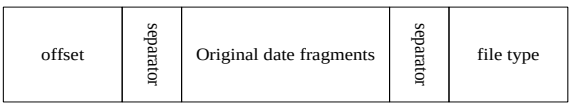

Fig. 3. The record format of original data fragments of file types.

3. Design of Testing Scheme for Malicious Code

3.1. The Design of Testing Tools for Malicious Codes

In this paper, our design of malicious code detection tool is shown in Fig. 4.

Graphical user interface and user command-line interface both establish conversations through joint words, controller, and receive returned results from controller after sending files or catalogs which are waiting to be scanned to controller.

Controller is responsible for connecting with user interface, receiving scanning tasks from user interface and scheduling scanner to achieve scanning tasks. Scanner, the scanning engine, is responsible for loading feature library, executing specific scanning tasks and returning scanning results to the controller.

3.2. The Scanning Flow

Fig. 5 is the flow chart of scanner scanning files. Firstly, the scanning engine will judge file types according to characteristics of file types' feature library.

Different scanning strategies will be adopted in different file types. Let's take the compressed files as an example. You need to unzip its files and scan them one by one. If MD5 of the file, after calculation, is in the white list, the file will be released. File program of malicious codes normally copy themselves in a fast way. When scanning the files with malicious codes, its MD5 will be put into Cache. If MD5 of the file is in the Cache, the file has malicious codes.

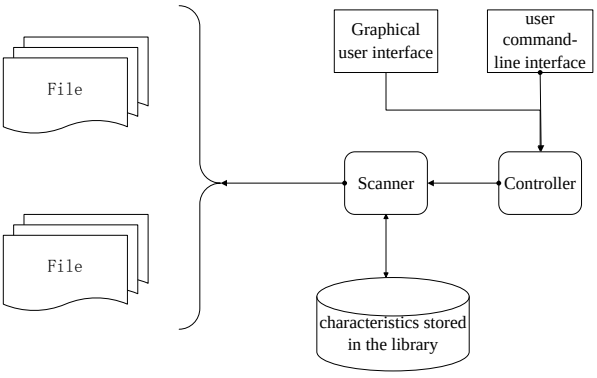

Fig. 4. The design of malicious code detection tool.

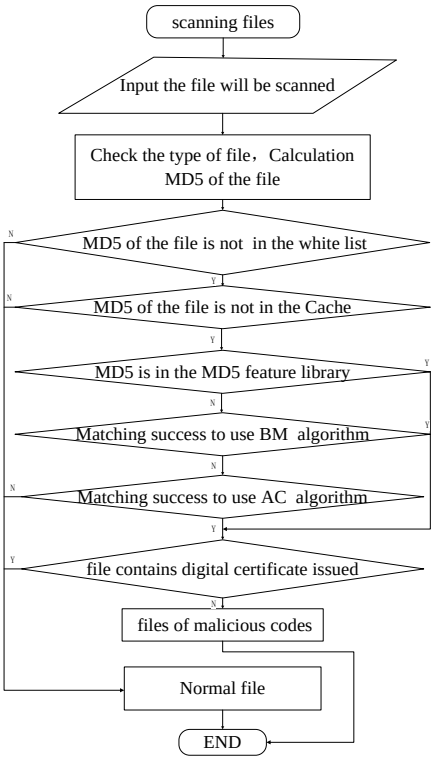

Fig. 5. The scheme overview.

It is unnecessary to precede subsequent scan which consists of checking whether its MD5 is in the MD5 feature library and whether it contains characteristics of original data fragments feature library by using Boyer-Moore (BM) algorithm, The AC algorithm. If we find they are in the feature library, we must further judge whether the file contains digital certificate issued by the trusted authority.

If the file contains the certificate, it will be released. Otherwise, it will be judged as files with malicious codes and its MD5 should be put into the Cache.

3.3. The Expended AC Algorithm

The AC algorithm adopts finite automation principle which is one of the fundamentals of compiling. The principle compares character string through the state transition. Using key words, character string will be searched to construct a finite state machine. Text searched is regarded as input. When inputting a character, statement may be transited based on the principle of the state machine. When the statement is transited to the corresponding state of outputting function, patterns will be compared. It is unnecessary to return back while scanning text with the AC algorithm, for saving time. In addition, time complexity of AC algorithm is $o(n)$ and time complexity is irrelevant to number and length of character strings to search.

While matching, the AC algorithm will construct turning function, failure function and output function. The process of constructing the failure function includes a Breadth-first search and a failure path which processes nodes hierarchically.

The process of failure path in constructing nodes: when you construct failure path, e.g., for Node1, the failure path of its parent node has been finished because of adapting Breadth-first search. At this time, you input value $'A'$ in Node1 and look for Node2 along the failure path from the parent node. If there is $'A'$ in the child node of Node 2 (Node3), the failure node of Node1 will point at Node 2. Otherwise, the failure node of Node1 will be pointed at root node.

For pattern sets, i.e., $\{hb, hbmy, she, her\}$, Fig. 6 is turning function, Fig. 7 is failure function, and Fig. 8 is output function.

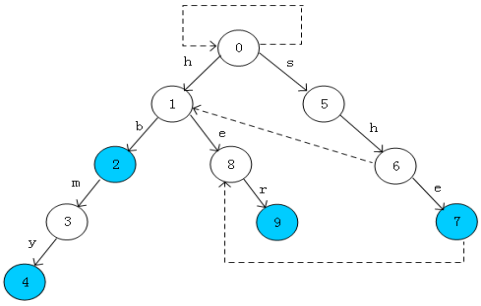

Fig. 6. The turning function *g.*

i	0	1	2	3	4	5	6	7	8	9
f	0	0	0	0	0	0	1	8	0	0

Fig. 7. The failure function *f.*

i	output(i)
2	hb
4	hbmy
7	she
9	her

Fig. 8. The output function.

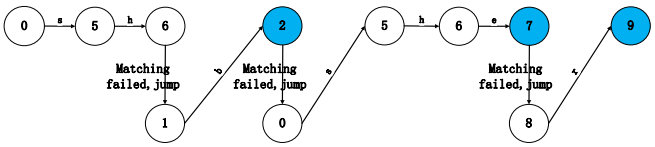

Fig. 9. Input transformation process of $\{shbsher\}$.

For the scanning text sets, i.e., the $\{shbsher\}$, Fig. 9 is the transformation process. When text transfers to the blue state points, i.e., the $\{2, 7, 9\}$, it starts pattern matching. The three key words i.e., *hb*, *she* and *her*, are searched from the set of $\{shbsher\}$.

Application context of AC algorithm in the malicious codes testing is that malicious codes engine regards thousands of original data fragments as patterns waiting to be matched, so do files, as input. In the meantime, malicious codes engine expands AC algorithm and makes its process existing wildcard characteristic of the pattern string. Our method mainly expands the regular features for the string which include the wildcard characters (*'*', '.', '^', '$'*). The basic idea of this method is the regular feature of regular expression (such as *"ab.*cd"*) into multiple fixed strings (*"ab" and "cd"*). When a match occurs in the order

of the fixed string or location, it is considered that a match also occurs with the entire regular expression. The algorithm flow of extended AC algorithm is described as follows: Firstly, split pattern and serial number, and load into a Trie-tree. Secondly, add failure path for the Trie-tree. Lastly, use the AC algorithm for pattern matching text

4. The Experiment and Evaluation

4.1. Factors that Affect Performance

Experiment 1: The experiment uses 4k size file which was generated randomly for testing, and the results are shown in Table III. Because it is necessary to load the feature library every time, when the number of files increases, the time of loading characteristic library is gradually offset, and the average scanning time also tends to be stable.

Experiment 2: The size of file affects the performance. The experiment scanned several files with different size separately, and recorded the scanning time of every file, the test results are shown in Table IV.

Table 3. The Number of Files and The Scanning Time

number of files	100	500	1000	2500	5000	10000
scanning time (s)	0.546	2.075	3.635	9.297	18.674	37.908
average time (s)/file	0.0055	0.0042	0.0036	0.0037	0.0037	0.0038

Table 4. The Size of Files and the Scanning Time

size of file(M)	32	64	128	256	512	1024	2048
Time(s)	0.414	0.440	0.478	0.468	0.749	0.780	0.764

Experiment 3: Our extended AC algorithm and the traditional AC algorithm are compared for malicious code detection. We select a set of 744 sample files. It contains 500 known malicious code files and 94 unknown malicious code files in the feature library, and 150 clean code files. The experimental result shows that it is very difficult for traditional AC algorithm to detect some simple malicious code files, but our extended AC algorithm can detect some malicious code files. It depends on the extended AC algorithm to join the wildcard in the characteristic segment. This scheme is compared with the traditional AC algorithm and the rate of false positive is reduced by 8.81%. It proves that adding features in the original data fragment can effectively reduce false positives.

TABLE 5. Comparition of Extended and Tradition AC Algorithm

Detection method	False negative (number)	False positive (number)	False negative rate (%)	False positive rate (%)	Accuracy rate (%)
Traditional AC algorithm	94	66	12.63%	8.87%	78.5%
Extended AC algorithm	44	0	5.91%	0%	94.09%

5. Conclusion

This paper introduces the development of the malicious code detection technology, and analyzes several kinds of malicious code detection methods. Working principle of the malicious code detection technology is described based on feature codes, the advantages and disadvantages are stated. The detection speed and the accuracy rate are compared with various malicious code detection methods. A feature library and a scanning algorithm are designed to be the key factors affecting the detection speed and the accuracy rate. Our approach has used the feature library with original data fragments, MD5 signature types, and file types, for testing. It also adds features of the wildcard in segments of the original data characteristics, and location

information of files. The scanning algorithm adapts a method combining an extended AC, BM algorithm and MD5 matching algorithm.

References

[1] Bonfante, G., Kaczmarek, M., & Marion, J. Y. (2009). Architecture of a morphological malware detector. *Journal in Computer Virology*, *5(3),* 263-270.

[2] Willems, C., Holz, T., & Freiling, F. (2007). Toward automated dynamic malware analysis using CW sandbox. *IEEE Security and Privacy*, *5(2),* 32-39.

[3] Undercoffer, J., Perich, F., & Nicholas, C. (2002, October). An Open Architecture for Distributed Intrusion Detection Services.

[4] McGraw, G., Morrisett, G. (2000). Attacking malicious code: A report to the infosec research council. *IEEE Software*, *17(5),* 33-41.

[5] Christodorescu, M., Jha, S., Seshia S. A., *et al.* (2005). Semantics-Aware malware detection. *IEEE Symposium Security and Privacy*, *11(8),* 32-46.

[6] Zou, C. C., Gong, W. B., & Towsley, D. (2005). The monitoring and early detection of Internet worms. *IEEE/ACM TON*, *13(5),* 961-974.

[7] Moser, A., Kruegel, C., & Kirda, E. (2007). Exploring multiple execution paths for malware analysis. *IEEE Symposium on Security and Privacy*, 231-245.

[8] Kruegel, B. C., & Kirda, E. (2006, January). A tool for analyzing malware. Retrieved from http://www.doc88.com/p-3495915580351.html.

[9] Tang, L. (2007). *Research on Function Isolation Mechanism of Secure Operating System.* University of Science and Technology of China, Hefei, China.

[10] Wang, D. Q. (2005). Research and Implementation of Malware Design and Analysis*. Computer and Science and Technology Department*, Tsinghua University, Beijing, China.

[11] Gu, R. J., Yan, P. L., Zou, T., *et al.* (2006). An automatic worm signature extraction algorithm based on attribution-oriented induction method. *Computer Science*, *33(7),* 127.

[12] Liu, W. W., Shi, Y., Guo Y., *et al*. (2009). A malicious code detection method based on integrated behavior characterization. *Acta Electronica Sinica*, *37(4),* 696-700.

[13] Li, Y., & Zuo, Z. H. (2007). An overview of object-code obfuscation technologies. *Computer Technology and Development*, *17(4),* 125-127.

[14] Cao, Y., Liang, X., Li Y. C., *et al*. (2008). Variance Analysis Based Stealthy Malicious Code Detection. *Computer Science*, *35(2),* 96-98.

[15] Wan, L., Liao, F. X., Zhang W., *et al*. A malicious code detection method. *Chinese Fault Tolerant Computing Conference*, CFTC'11.

[16] Chen, L. (2012). *Research on the Main Technologies in Malware Code Detection.* Yangzhou University of China, JiangSu, China.

[17] Du, N., Han, L. S., Fu C., *et al*. (2015). Detection of malware code based on acquaintance degree. *Computer Science*, *42(1),* 187-192.

[18] Yang, H., Zhang, Y. Q., Hu Y. P., *et al*. (2014). A malware behavior detection system of android applications based on multi-class features. *Chinese Journal of Computers, 37(1),* 15-27.

[19] Wang, R., Feng, D. G., Yang, Y., *et al.* (2012). Semantics-based malware behavior signature extraction and detection method. *Journal of Software*, *23(2),* 378-393.

[20] Chen, H. Q. (2009). Feature selection in malware detection. *Journal of University of Electronic Science and Technology of China*, *38*.

[21] An, J., Yang, Y. X., & Li, Z. X. (2013). Obfuscated malicious code detection with path condition analysis.

Journal of Hunan University, *40(9),* 86-90.

Wenshuang Yin is a lecturer in School of Science, Hubei University for Nationalities. She received the B.S. degree in mathematics from Hubei University for Nationalities, China, in 1999, and the M.S. degree in fundamental mathematics from Guizhou University, China, in 2006. Her research interests mainly focus on nonlinear analysis and pattern recognition.

Changcheng Xiang is an associate professor in School of Science, Hubei University for Nationalities. He received the M.S. degree in applied mathematics from School of mathematics and statistics, Huazhong Normal University, China, in 2004，and the Ph.D in control theory and control engineering from School of automation, Chongqing University, China, in 2008. His research interests mainly focus on intelligent computing, image processing, and cryptography.

Dingding Yang is a postgraduate in School of Science, Hubei University for Nationalities. He received the B.S. degree in mathematics and applied mathematics from Lvliang University, Shanxi, China, in 2015. He is currently pursuing the M.S. degree in operational research and cybernetics from School of Science, Hubei University for Nationalities, China. His research interests mainly focus on digital image processing and VC ++ programming.

Shiqiang Chen is a professor in School of Science, Hubei University for Nationalities. He received the M.S. degree in computer software and theory from Institute of Computer Software, Guizhou University, China, in 2005. His research interests mainly focus on network measurement and monitoring, network communication, internet of things and network security technology.