マルウェアの部分コードによる類似度判定および機能推定法

大久保諒 † 森井昌克 † 伊沢亮一 ‡ 井上大介 ‡ 中尾康二 ‡

† 神戸大学大学院工学研究科
657-8501 兵庫県神戸市灘区六甲台町 1-1
okubo.1@stu.kobe-u.ac.jp
mmorii@kobe-u.ac.jp

‡ 独立行政法人情報通信研究機構
184-0015 東京都小金井市貫井北町 4 丁目 2-1
{isawa},{dai},{ko-nakao}@nict.go.jp

あらまし

マルウェアの類似度判定では，検体を入手する度にデータベースに保存されている大量の検体に対して類似度を求める必要があり，計算量の削減が課題として挙げられる．従来の類似度判定法では LCS(Longest Common Strings) や n-gram を用いるものが多い．LCS と n-gram の計算量はともに $O(m \times n)$ であり，大量の検体かつ頻繁に類似度を求める状況においては計算量が小さいとは言い難い．本研究では高速な類似度判定を目的とし，マルウェアのバイトコードの出現頻度を利用する方法を提案する．提案方式の要点は，各マルウェアのバイトコードの出現頻度をデータベース化するところにある．出現頻度のデータベースを作成した後は，正規化相互相関を用いることで類似度の計算量が $O(1)$ となる．また類似度から機能毎にポイントを与え，マルウェアの機能を推定する手法を提案する．

Function Estimation Method for Malwares based on part of Binary Code

Ryo Okubo† Masakatu Morii† Ryoichi Isawa ‡ Daisuke Inoue ‡
Koji Nakao ‡

†Graduate school of Engineering, Kobe University
1-1 Rokkodai-cho, Nada-ward, Kobe 657-8501, JAPAN
okubo.1@stu.kobe-u.ac.jp
mmorii@kobe-u.ac.jp
‡National Institute of Information and Communications Technology
4-2-1 Nukuikitamachi, Koganei-city, Tokyo 184-0015, JAPAN
{isawa},{dai},{ko-nakao}@nict.go.jp

Abstract

Because malware analysis using similarities between malwares needs to make a lot of comparison, calculation amount is an important factor. Existing methods use LCS(Longest Common Strings) or n-gram to calculate similarity. The calculation amount of them is $O(m \times n)$. This calculation amount is not appropriate for making a lot of comparison. We propose a fast method for calculating similarity between malwares. We focus on distribution of byte code. To calculate a similarity between the two of malwares, we apply Zero-mean Normalized Cross-Correlation. Once we get the distribution of byte code, the calculation amount is $O(1)$. Because this method does not influenced by file size, we can calculate similarity at short times in any case. Moreover we estimate function by adding point to each function according to similarity.

1 Introduction

These days a great number of people use the Internet for shopping or on-line banking. These services deal with their important personal information. Once a user's information leaks, the user will suffer a loss. If an Internet user's computer is infected by malware, these information will be easily stolen. Antivirus vendors provide antivirus softwares to prevent the Internet users from such damage. To protect the users from malwares, the antivirus vendors must analyze each malware. Symantec reported that 403 million malwares were created in 2011[2]. With various malwares increasing rapidly, a burden on malware analysts cannot be ignored. Many researchers propose new methods for efficient analysis to ease this burden. There are two methods for analyzing malware, one is dynamic analysis, and the other is static analysis. In dynamic analysis, researchers actually execute a malware and observe its behavior. Because this method just reports the behavior that the malware has done, the result will be precise. However, malwares infect the computer on which they execute, and we have to recover the computer system after analyzing malwares. On the other hand, static analysis does not require executing malware. In this paper, we propose a fast method for static analysis to calculate a lot of similarities at short times. We focus on the distribution of byte code to calculate similarities between malwares. The distribution of byte code is a simple characteristic and is easy to compare. We make use of ZNCC(Zero-mean Normalized Cross-Correlation) to calculate similarity. Because our method needs only the distribution of byte code, it can fast calculate the similarities regardless of the file sizes of malwares. Moreover we use the similarity to estimate function of malwares. We apply this method for MWS dataset.

2 Related Works

We calculate similarities between malwares by static analysis, and use it for function estimation. In this section we introduce two works that calculate similarity by static analysis.

Higashi et al.[3] calculate similarities based on functions such as Windows APIs or subroutines. They extract functions from disassembled code. They calculate similarities between functions of an unknow malware and functions of an analyzed malware to estimate the function of the unknown malware. Suppose that we have a target malware that we try to analyze and an analyzed malware. The target malware has functions a, b, and c. The analyzed malware has function e, f, and g which are known. If the similarity between function a and function e is high, Higashi et al. assume that the target malware have function e. They use 3-gram and LCS (Longest Common Strings) to calculate similarities. Both methods are used for extracting common part of two strings. 3-gram makes factors of 3 length from strings and compare the factors. LCS extracts the longest common strings from two strings, according to eq. 1, where both a_i and b_i denote strings.

$$L(a_i, b_j) = \begin{cases} 0 \ (a = 0 \text{ or } b = 0) \\ \max(L(a_{i-1}, b_j), L(a_i, b_{j-1})) \\ \qquad\qquad (a_i \neq b_i) \\ L(a_{i-1}, b_{j-1}) + 1 \ (a_i = b_i) \end{cases} \quad (1)$$

Their method outputs a lot of data for comparing one pair of malwares. Therefore, the problem is that how to deal with these data.

Iwamura et al.[4] calculate similarity from disassembled code using LCS. Because LCS requires a large amount of memory, they propose a contraction algorithm for disassembled code. One of the major problems is that calculating similarity with LCS requires computational costs.

The number of malwares is increasing rapidly. Since we considered the fast algorithm for malware analysis is needed in these days. In this paper we propose the fast method to calculate similarities between malwares and to estimate function of malwares.

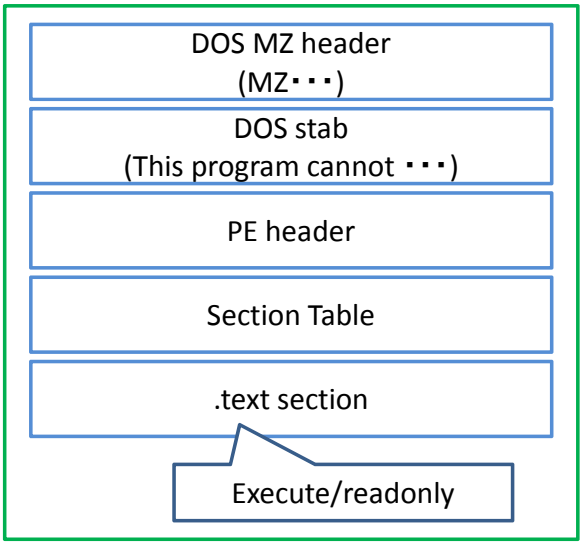

Figure 1: PE File structure

3 Similarity between Malwares

We use similarities between malwares to estimate function of malwares. We focus on the distribution of byte code to calculate similarities. The distribution of byte code is the simplest character of a malware, and we can easily compare two malwares using their distribution of byte code. Moreover once we get the distribution of byte code, we can calculate similarity without influence of file size. To calculate similarity according to malware function, we extract the executable section. PE executable file consists of several sections. Figure 1 shows a structure of PE executable file. Section table is available to decide which section is executable. Section table includes section information such as Name, Virtual Size, and we can also get section Characteristics from section table. Section Characteristics consists of 4 byte code. Characteristics is logical sum of flags. If a section includes executable code, the characteristics has 0x00000020 flag or 0x20000000 flag. We extract executable sections or sections that have executable code to calculate similarity. Moreover, we consider 2 byte opcode as a trait. Because 2 byte opcode begins with 0x0f, we extract the byte just after 0x0f and get the distribution. Figure 2 shows

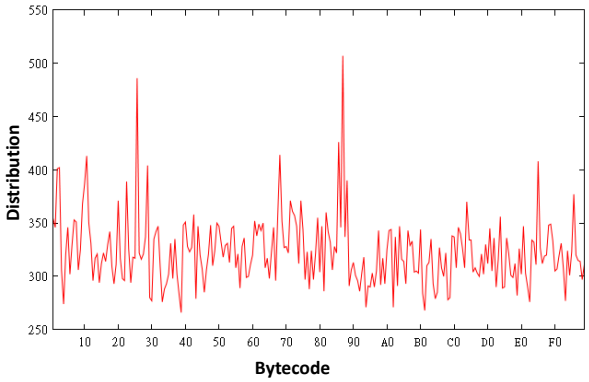

Figure 2: PE File structure

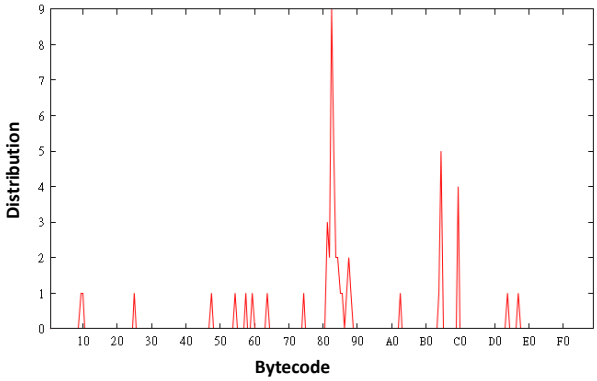

Figure 3: PE File structure

a distribution of all byte code in executable section, on the other hand figure 3 shows a distribution of the byte code just after 0x0f. We can expect that deviation of similarity using figure 3 is greater than that of figure 2. We make use of ZNCC to calculate similarity by distribution of byte code. Eq. 2 shows how to derive similarity R between I and T by ZNCC.

$$R = \frac{\sum_{i=1}^{N-1}(I(i)-\bar{I})(T(i)-\bar{T})}{\sqrt{\sum_{i=1}^{N-1}(I(i)-\bar{I})^2 \times \sum_{i=1}^{N-1}(T(i)-\bar{T})^2}} \quad (2)$$

Similarity calculated by ZNCC runs from -1 through 1. In this work, if similarity is near to 1, we assume that there is a certain relationship between malwares. Finally in this section we give a procedure for calculating similarity.

Step1 Extract section information from section table.

Step2 Obtain section which characteristics includes 0x00000020 or 0x20000000 flag.

Step3 Determine the distribution of byte code that are just after 0x0f.

Step4 Calculate similarity by ZNCC.

4 Function Estimation

We make use of similarity to estimate functions of a malware. To estimate function of unknown malware, we calculate hundreds of similarities between unknown malwares and analyzed malwares. Functions of analyzed malware is supposed to be known. We add point to functions that analyzed malware have according to similarities between the target malware and the analyzed malware. The added point is calculated by eq. 3. If a point of a function was higher than that of other functions, that function is estimated to be possessed by the analyzing malware.

$$P(k) = \frac{\sum_m |r(m)f(m,k)|\, r(m)f(m,k)}{\sqrt{\sum_m f(m,k)}} \quad (3)$$

We put the case that we estimate function of unknown malware A, and we have analyzed malwares B and C. We suppose that Malware B have function a and function b, Malware C have function a and function c. We consider the distribution of A as I in eq. 2 and that of B as T to calculate similarity R_{AB}. The points of functions P_a, P_b, and P_c are calculated as below.

$$P_a = \frac{R_{AB}|R_{AB}| + R_{AC}|R_{AC}|}{\sqrt{2}} \quad (4)$$

$$P_b = R_{AB}|R_{AB}| \quad (5)$$

$$P_c = R_{AC}|R_{AC}| \quad (6)$$

In this work, we use NICT's dynamic analysis result for function estimation. Table 1 shows all function that we use for function estimation.

5 Experiment

We used MWS dataset for this experiment. Table 2 shows the malware names by McAfee and the number of each malware in MWS dataset.

Table 1: Functions

Function
addReg
create,delReg
execute
alterFile
createFile(system dir)
search(Microsoft Phonebook file)
movefile
alterTxtFile
newHash
readFile
open,attrFile
openProcess
createDir
addReg(auto start)
copyFile
createService
alterFile
createFile(connect to Microsoft RPC service)
deleteFile(delete itself)
search
createMutex
openwindow
alterProcess
connect(web)
connect(DNS)
sendMail
backdoor

We estimated the functions of two malwares. Their hash values are 0ea635* and 6acb0c*, one is named W32/RAHack and the other is named BackDoor-DOQ.gen.e. A lot of W32/RAHack are in the dataset, whose hash values are distinct from each other, and one BackDoor-DOQ.gen.e is in the dataset. Figure 4 and figure 5 present distributions of similarities between target malwares and analyzed malwares.

In Figure 4, we can clearly divide similarities into two. While in figure 5, we cannot. This means that estimating function of 6acb0c* is harder than that of 0ea635*. Table.3 is the result of function estimation of 0ea635*. We can see that the points of the functions in 'Pos-

Table 2: MWS dataset Classified by McAfee

Name	number
W32/Conficker.worm.gen.b	134
W32/Conficker.worm.gen.a	440
W32/Conficker.worm	106
BackDoor-DOQ.gen.e	1
W32/RAHack	9828
Generic Dropper.acj	6
W32/Virut.n.gen W32/IRCbot.gen.b	1
W32/Sdbot.worm!mj	2
W32/Conficker.worm.gen.d	3
W32/Autorun.worm.h	1
W32/Virut.gen.a , W32/RAHack	8
UNKNOWN	1
W32/Sdbot.worm.gen.ax	1
W32/Virut.n.gen W32/IRCbot.gen.b	1
Generic.dx!bcqz	1
W32/IRCbot.gen.a	1
W32/Sdbot.worm!mh	1
Generic BackDoor!1cj	1
W32/Sdbot.worm!mc	1

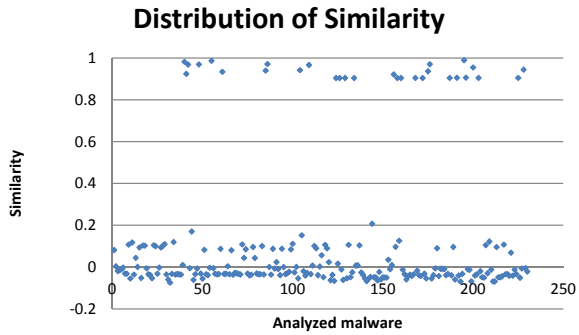

Figure 4: 0ea635* simlirality distribution

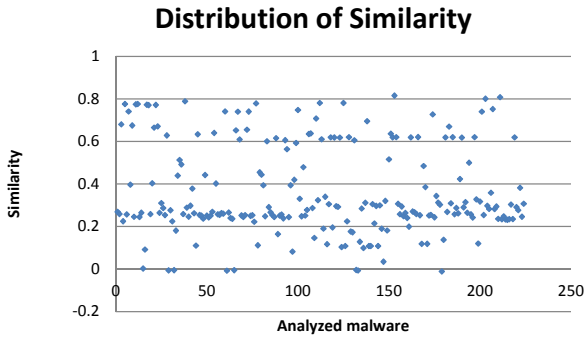

Figure 5: 6acb0c* simlirality distribution

session' are clearly higher than those of 'Non-possession'. This means that we can estimate function of 0ea635*. While 6acb0c* has only two functions, add register and delete itself. This malware seems to detect analyzing environment and delete itself. 6acb635* is named BackDoor-DOQ.gen.e, which original malware is BackDoor-DOQ. We make use of report of BackDoor-DOQ for function estimation. Table.4 shows its result. From this table, the point of copyFile is higher than that of two of possessed functions. While points of other non-possessed functions are lower than that of possessed functions. From this result, we can estimate the functions of the malware.

On the other hand, we have several malwares that function cannot be estimated. The reason is that we do not have a sufficient database to estimate function of malwares. Because the database we have now consists of malwares that occurred in a specified period, samples are biased. In this method, if all similarities between the target malware and analyzed malwares are low, the estimation cannot be succeeded. The database is the most important part for this method.

6 Conclusion

In this work, we proposed a method for estimating function of malwares. We also proposed a fast algorithm for calculating similarity between two malwares to estimate functions. We made use of distribution of byte code that is just after 0x0f, which is used for two byte opcode. We calculated similarity using Zero-mean Normalized Cross-Correlation between the two of malwares. Using this method, we can fast calculate similarity. This method allows us to calculate a lot of similarities in a short time. Making use of this characteristic, we proposed a method for estimating function of malware. Our method estimates unknow malware's function by calculating similarities between target malware and a lot of analyzed malwares which function is known. We estimated function of two

Table 3: Function Estimation of 0ea635*

Possession		Non-possession	
Function	Point	Function	Point
createService	3.662	readFile	1.420
search	2.893	open,attrFile	1.388
copyFile	2.341	openProcess	0.874
alterFile	2.300	search	0.790
execute	2.097	backdoor	0.771
deleteFile	1.911	connect	0.771
alterTxtFile	1.749	alterProcess	0.763
addReg	1.734	createFile	0.021
newHash	1.727	movefile	0.018
createFile	1.726	sendMail	0.009
createMutex	1.718	openwindow	0.003
create,delReg	1.686	createDir	0.003

Table 4: Function Estimation of 6acb635*

Possession		Non-possession	
create,delReg	2.695	copyFile	2.120
createMutex	2.486	open,attrFile	1.975
createFile	2.477	alterTxtFile	1.974
addReg	2.450	readFile	1.973
execute	2.414	openProcess	1.846
connect	2.352	search	1.788
backdoor	2.352	openwindow	0.950
alterProcess	2.345	createService	0.809
deleteFile	2.210	movefile	0.725
alterFile	2.170	createFile	0.663
newHash	2.073	sendMail	0.643
search	1.981	createDir	0.502

malwares from MWS dataset. One was correctly estimated function, the other was not correctly estimated but was sufficient for estimation. In our method, the database of analyzed malwares is the most important factor for estimation. Because we do not have sufficient database for this method, creating it is one of our future works.

Acknowledgement

It is a pleasure to acknowledge the hospitality and encouragement of the members, especially Dr. Masashi Eto, of the Cybersecurity Laboratory in Network Security Research Institute, NICT

References

[1] Intel 64 and IA-32 Architectures Software Developer's Manual, ‘‘http://www.intel.com’’

[2] Symantec Security Response, ‘‘http://www.symantec.com/’’

[3] Yuka Higashi, You Nakatsuru, Takashi Manabe, Atsuo Inomata, Kazutoshi Fujikawa, Hideki Sunahara, “Proposal of function estimation by malware code's function similarity, ”SCIS2011

[4] MAKOTO IWAMURA, MITSUTAKA ITOH, YOICHI MURAOKA, “Automatic Malware Classification System Based on Similarity of Machine Code Instruction, ”Journal of Information Processing, vol.51, No.9, pp1-11, 2010

[5] Yuko Ozasa, Souma Katsute, Masakatu Morii, Koji Nakao, “Estimated Function of Malicious Code by Memory Dump Analysis, ”CSS2009

[6] Yuu Arai, Makoto Iwamura, Yuhei Kawakoya, Kazufumi Aoki, Yuji Hoshizawa, “Analyzing Malware Fighting against infection incidents with free tools, ”O'REILLY Japan, Inc.