L-029

マルウェアの部分コードによる類似度判定と機能推定

Function Estimation Method for Malwares based on part of binary code

大久保　諒 †　森井　昌克 †　伊沢　亮一 ‡　中尾　康二 ‡　井上　大介 ‡
Ryo Okubo　Masakatu Morii　Ryoichi Isawa　Koji Nakao　Daisuke Inoue

1　はじめに

近年，インターネットの普及によりマルウェアの増加が問題となっている．マルウェアとは悪意を持って作成されたソフトウェア (Malicious Software) であり，ウィルスやトロイの木馬がこれに含まれる．マルウェア被害からユーザを保護するために多くのアンチウィルスソフトが存在する．アンチウィルスソフトのほとんどがパターンマッチングによってマルウェアの検出を行っている．パターンマッチングでは既知のマルウェアのコードから一部を抜き出し定義ファイルを作成する．この定義ファイルと実行ファイルを比較することによってマルウェア検出を可能にしている．しかし，パターンマッチングでは定義ファイルとして抽出された部分が改変されればマルウェアとして検出されないため，新たに発生したマルウェアの亜種を検出できない可能性がある．現在新たに発生しているマルウェアのほとんどが既知のマルウェアの亜種である．亜種とされるマルウェアでは既存のマルウェアの一部を改変して作成されるため発生頻度が高い．本研究では，マルウェアの亜種判定と未知のマルウェアの機能推定を行う．パターンマッチングによる亜種判定では前述したとおり問題があるため，マルウェアのバイトコードの出現頻度に着目した．バイトコードの出現頻度から類似度を導出するため，一部のみを改変された亜種の検出により高い信頼性を持たせることができる．また，未知のマルウェアに対しては類似度をもとに機能ごとにポイントを与えることによって確率的に解析対象としたマルウェアが持つ機能を推定する．この際，一対一の類似度から機能を推定することは困難であるため未知のマルウェアに対して複数の既知のマルウェアとの類似度を導出し，統合することによって機能推定を可能にする．

2　予備知識

本研究ではバイトコードの出現頻度から類似度を導出する手法として正規化相互相関を用いた．また，ファイル全体のバイトコードの出現頻度ではなく実行コードを含む，もしくは実行可能セクションのみのバイトコードの出現頻度を取得することにより，マルウェアの動作に着目した類似度を導出した．本章では類似度導出に用いた正規化相互相関と PE ファイル構造について言及する．

2.1　正規化相互相関

正規化相互相関による一次元グラフの類似性の導出式を式 1 に示す．正規化相互相関では，サンプリング点を決定しその点における値からグラフ間の類似度を求める．ある 2 つのグラフを考えた場合，サンプリング点での値が高い割合で一致している場合，類似度が高くなることは明らかである．一方で各サンプリング点での値がすべて大きく異なっていてもその差が均等であれば高い類似度を導く．正規化相互相関から導かれる値は –1 から 1 であり，本研究においては導かれた値が 1 に近ければ検体に類似性があると判断する．

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

2.2　PE ファイル構造

PE ファイル構造を図 1 に示す．PE ファイルは Windows において実行ファイルがとる形式であり，exe や dll などがこれに含まれる．本研究ではセクションテーブルから各セクションの情報を抽出し，実行コードを含む，もしくはコードとして実行可能となっているセクションを対象として類似度の導出を行う．ここでは各データ領域について説明を加えるが特にセクションテーブルを詳細に説明する．

2.2.1　DOS MZ ヘッダ

DOS MZ ヘッダの先頭の 2byte は 0x4D,0x5A がシグネチャとして設定されている．これによってファイルが PE フォーマットをとっていることを示す．また最後の 4byte は e_lfanew と呼ばれ，PE ヘッダの先頭アドレスが記述されている．

2.2.2　DOS スタブ

ここでは PE ファイルが誤って MS-DOS 上で実行された場合に出力するプログラムが記述されている．ほとんどの場合において，この領域は「This program cannot be run in DOS mode」を出力し，終了するプログラムが記述されている．

2.2.3　PE ヘッダ

PE ヘッダは 3 つのメンバからなっており，それぞれが Signature，IMAGE_FILE_HEADER，IMAGE_OPTIONAL_HEADER と呼ばれる PE ヘッダの先頭 4byte は Signature であり，0x00004550 がシグネチャとして設定されている．IMAGE_FILE_HEADER には適応マシンタイプやセクションの数，タイムスタンプ等のファイル情報が記述される．IMAGE_OPTIONAL_HEADER ではコードサイズやエントリポイントのアドレスなど，IMAGE_FILE_HEADER より詳細なファイルの情報が記述される．

2.2.4　セクションテーブル

セクションテーブルには各セクションの情報が記述される．セクション情報は Name，VirtualSize，VirtualAddress，SizeOfRawData，PointerToRawData，PointerToRelocations，PointerToLinenumbers，NumberOfRelocations，NumberOfLinenumbers，Characteristics のメンバで構成さ

† 神戸大学大学院工学研究科，Graduate School of Engineering, Kobe University
‡ 独立行政法人情報通信研究機構, National Institute of Information and Communications Technology

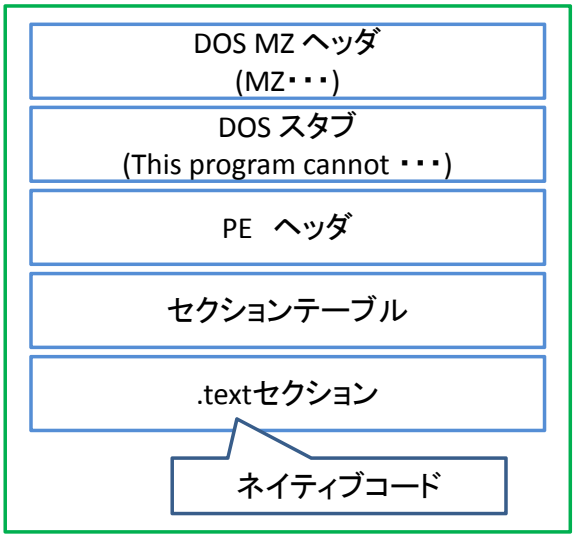

図 1　PE ファイル構造

れている．一般的にはネイティブコードを含むセクションは.text であり，.data セクションは初期値を持たない変数を格納するためのセクションとなっている．しかしこれらの Name は単なる名前であるため.text セクションであっても実行コード部分ではない場合がある．実際にセクションの属性を示すのは Characteristics である．Characteristics は 4byte であり，セクションの属性を表すフラグの論理和が記述されている．本研究では実行コードを含む，もしくは実行可能セクションに着目した．これらのフラグは実行コードを含むセクション場合 0x00000020 であり，実行可能セクションの場合 0x20000000 である．

3　マルウェアの機能

マルウェアの機能推定を行うにあたりマルウェアの機能に定義を与える．本研究では既知のマルウェアの解析結果として NICT の NONSTOP から得られた動的解析結果を用いる．NONSTOP の解析レポートではマルウェアの活動概要が出力され，機能ごとに sid として管理されている．本研究で用いた 235 検体がもつ sid とその詳細を表 1 に示す．

4　既存研究

本研究では静的解析によりマルウェア間の類以度を導出する手法をとった．同様に静的解析により類似度を判定する手法を提案した既存研究とその問題点について考察する．既存研究の類似度判定において用いられるものとして N-gram，LCS，Jaccard 係数などがあげられる．そこで本章ではまず N-gram，LCS，Jaccard 係数について説明する．次にこれらをマルウェアの類似度判定に用いた既存研究として東らの手法と岩村らの手法，我々が以前提案した手法について概説し最後にこれらの既存研究を考察する．

sid	機能
00000001	addReg
00000002	createReg deleteReg
00000005	execute
00000006	alterFile
00000007	createFile(system dir)
00000009	search(Microsoft Phonebook file)
00000012	movefile
00000015	alterTxtFile
00000017	newHash
00000018	readFile
00000019	openFile attrFile
00000020	openProcess
00000021	createDir
00010001	addReg(auto start)
00200001	copyFile
00300001	createService
00400001	alterFile
00600001	createFile(connect to Microsoft RPC service)
00700001	deleteFile(delete itself)
00800001	search
01000001	createMutex
02000001	openwindow
04000002	alterProcess
06000013	connect(web)
20000002	connect(DNS)
30000001	sendMail
40000001	backdoor

表 1　sid と機能の対応表

4.1　類以度導出に用いられる概念

本節ではマルウェア間の類以度を導出する過程において一般的に用いられる概念について説明する．

4.1.1　N-gram

N-gram はある文書の先頭から N 文字を抜き出し次に先頭を 1 文字後退させまた N 文字を抜き出す作業を繰り返すことによってすべて N 文字の要素の集合を得る．たとえば 2-gram を適用する文書が ABCD である時得られる集合は {AB，BC，CD} となる．N-gram では文書同士でどの程度重複する要素があるかを考えることで類似度を導出する．N-gram では一方の文書から要素を取得しその各々の要素がもう一方の文書にあるかを走査するため計算量は $O(n^2)$ となる．

4.1.2　LCS(最長共通部分列)

LCS(最長共通部分列) は 2 つの文字列における最長の共通部分を求めるものである．文字列 a_i と b_j の LCS 長 $(L(a_i, b_j))$ は式 2 のようにして求めることができる．

$$L(a_i, b_j) = \begin{cases} 0\ (a = 0 \text{ or } b = 0) \\ \max(L(a_{i-1}, b_j), L(a_i, b_{j-1}))\ (a_i \neq b_i) \\ L(a_{i-1}, b_{j-1}) + 1\ (a_i = b_i) \end{cases} \quad (2)$$

LCS 長は表を用いることによって容易に導出することができる．例えば $a = \{abc\}$，$b = \{abd\}$ とすると表 2 に

よってその LCS 長は 2 であることが分かる．この LCS 長がもとの文書の長さと似ている場合類似度が高いと判断され，短い場合は類似度が低いと判断かれる．LCS を求める計算量は $O(n^2)$ である．

	0	a	b	c
0	0	0	0	0
a	0	1	1	1
b	0	1	2	2
d	0	1	2	2

表 2　LCS 長の導出

4.1.3　Jaccard 係数

Jaccard 係数はある二つの集合がある時にその集合同士の類似度を導出する．導出方法を式 3 に示す．ここで A, B を比較する集合とする．

$$R_{Jaccard} = \frac{|A \cap B|}{|A \cup B|} \tag{3}$$

式から分かるように Jaccard 係数は 0 から 1.0 の値をとる．Jaccard 係数の値が 1.0 の場合は 2 つの集合がまったく同じであることを示し，0 の場合は完全に異なっていることを示す．

4.2　類以度に着目した研究

前節で説明した概念を用いてマルウェア間の類以度を導出する既存研究を紹介する．

4.2.1　機能に着目した手法 [3]

東らは機能面での類似度を求めることに重点を置いている．その手法としては比較したいマルウェアの全関数を取得しその総当たりによって類似度を決定する．類似度を求める際には 3-gram を用いた手法と LCS と Jaccard 係数を併用した手法を提案している．3-gram を用いた手法は関数ごとにそのバイナリデータと逆アセンブルしたテキストデータを用いている．全関数の総当たりを実験するため大量のデータが生まれることになる．そこで東らは関数ごとの類似度の最大値に着目しそれを最終的な類似度としている．LCS と Jaccard 係数を用いた手法ではバイナリデータのみを用いて検証している．これは LCS を用いると計算量が多くなり，テキストデータを用いると莫大なメモリを消費するためである．

4.2.2　LCS を用いた手法 [4]

岩村らはマルウェアのオリジナルコードを逆アセンブルし LCS を求める．この LCS を用いて 2 つのマルウェアの和集合と積集合を考えることによって Jaccard 係数を求めてそれをマルウェアの類似度として導出している．また岩村らは LCS の問題点である計算量をビットベクトル化と新たに提案した縮約命令を用いることで LCS による類似度の導出を実現している．

4.2.3　バイナリコードに着目した手法 [5]

我々は以前バイナリコードからランダムに一定長のバイト列を抜き出しそのバイト列が比較対象のバイナリコードに含まれるかどうかによって類似度を導出した．具体的には比較したい検体のバイナリコードからチェックコードとして 8byte のバイト列を 50 個抜き出しそれがもう一方のマルウェアのバイナリコードと照らし合わせ，合致した数をもとに類似度を導出する．この 8byte のバイト列のうちその半分以上の長さが 0x00 もしくは 0xff で占められる場合は再度別のバイト列の取得をする．類似度導出にはターゲットとしたマルウェアのチェックコードがサンプルのマルウェアに含まれる数によって決定する．

4.3　既存研究の問題点

東らの手法はマルウェアの関数をすべて総当たりで類似度を導出している．このことから計算量が多く無為なデータが大量に発生しデータ処理にコストかかってしまうといった問題点を抱えている．また類似度がファイルサイズに少なからず依存してしまい，同様の機能を持った関数同士でも低い類似度が出てしまうなど結果に対しても不安要素を残している．また岩村らの手法はテキストを LCS と Jaccard 係数をもちいて計算コストの削減をしているがそれでも計算コストが高い．N-gram と LCS はともに $O(n^2)$ の計算量を持つ．我々が以前行った手法はチェックコードがランダムに取得されているためどこをとるかによって結果が大きく変わる可能性がある．現在のネットワーク社会において日々大量に発生しているマルウェアを解析するためには計算コストは重要な要素である．そこで本研究では高速にマルウェア同士の類似度を導出するためにバイトコードの出現頻度に着目した．

5　バイトコードの出現頻度に着目したマルウェアの類似度導出

本研究ではパックされていないマルウェアのバイナリデータもしくはアンパックされたマルウェアのバイナリデータを用いる．バイナリデータからバイトコードの出現頻度を取得し，それを用いることによって任意の 2 検体の類似度を導出する．出現頻度から類似度を導出するため完全なデータが必ずしも必要というわけではなくアンパックの過程において一部が欠損している場合であっても，比較し類似度を導出できる．またすでに取得済みのマルウェアに関してはそのバイトコードの出現頻度を保持しておくことで，高速に類似度を導出することができる．以下に本研究で提案する類似度導出までの過程を示す．

Step1PE ファイルのセクションテーブルから各セクション情報を抽出する．
Step2 セクション情報の内，Characteristics メンバに 0x00000020，または 0x20000000 のフラグが立っているセクションのバイナリデータをすべて取得する．この時 0x0f の後にある 1byte のみを取得する．
Step3 取得したバイトコードごとにその出現頻度を求める．
Step4 正規化相互相関によりバイトコードの出現頻度から類似度を導出する．

実行コードを含むかもしくは実行可能なセクションを抽出することによってファイルの動作に応じた類似度が求められる．類似度の偏差を大きくするために 2byte 以上のオペコードには必須となる 0x0f の後の 1byte のみに着目した．実際に検体を用いてすべてのバイトコードの出現頻度をグラフ化したものを図 2，0x0f の後のバイトコードのみを取得し，出現頻度を求めたものを図 3 に示

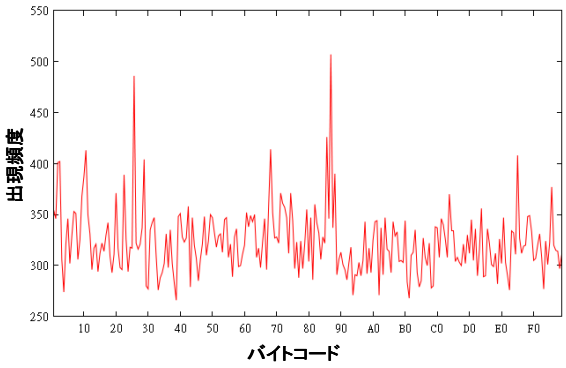

図 2　すべてのバイトコードの出現頻度

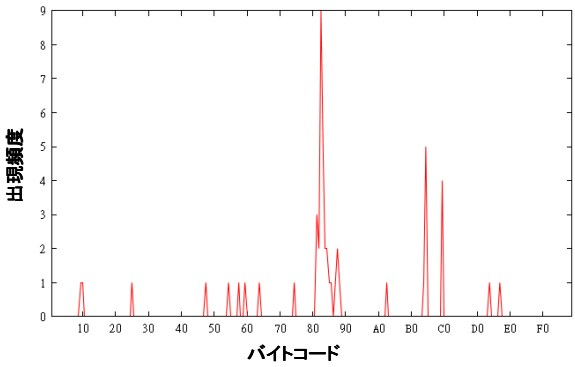

図 3　0x0f の後のバイトコードの出現頻度

す．0x0f の後のバイトのみを取得することで全体として取得するバイトコードが減っていることが分かる．

本研究では機能を推定することを目的としているため，動作に類似性が無いものに関してはできる限り低い類似度を与える必要がある．0x0f の後のバイトコードにのみ着目することによって，挙動が違ったマルウェアの類似度は全体のバイトコードの出現頻度を用いた場合に比べて，より低く導出できると考えられる．

6　類似度を用いたマルウェアの機能推定法

前述した類似度は PE ファイルの動作の観点から導出したことからこの類似度を用いてマルウェアが持つ機能を推定できると考えられる．本章では類似度からマルウェアの機能を推定する方法について説明する．本研究での類似度導出では既知のマルウェアのバイトコードごとの分布を導出し保持していれば，あるマルウェアを与えられたときにそのマルウェアのバイトコードごとの分布を求めるだけで高速に類似度を導出できる．これにより短時間で未知のマルウェアに対して多数の既知のマルウェアとの類似度を導出することができる．一対一の類似度では一定以上の値が与えられない場合を除いては未知のマルウェアの機能を推定することは困難である．しかし一つの未知のマルウェアに対して多数の既知のマルウェアの類似度を求め，統合的に解析することによって未知のマルウェアの機能が推定できる．機能推定の際には機能ごとに保持するポイントを用いる．本研究ではマルウェアの機能として NICT の NONSTOP の動的解析結果から sid を用いた．

具体的に未知のマルウェアの機能推定を行う方法について述べる．解析が完了したマルウェアをサンプルマルウェア，解析対象とするマルウェアをターゲットマルウェアとする．サンプルマルウェアには機能が既知のものを用いる．ターゲットマルウェアとサンプルマルウェアの類以度に更に類以度の絶対値を乗じた値をサンプルマルウェアが持つ機能にあたえる．ここで類以度に類以度の絶対値を乗じた値を利用するのは類以度の高いものと類以度の低いものの偏差を大きくする目的がある．ターゲットマルウェア 1 検体に対して複数のサンプルマルウェアとの類以度を導出し機能ごとに与えられた値の総和をとり，機能ごとのポイントを導出することで機能推定を行う．大きなポイントを持つ機能ほどターゲットマルウェアが保持する確率が高いと仮定し，機能推定を行う．具体的なポイントの導出方は以下の通りである．機能を k，検体を m とする．この時，検体 m が機能 k を保有する場合 $f(m,k) = 1$ その他の場合は $f(m,k) = 0$ となるような f を定義する．また解析対象とするマルウェアと検体 m の類似度を $r(m)$ とする．この時解析対象が持つ機能を推定するために機能ごとにポイントを導出する．機能 k を考えたとき，k にあたえられるポイント $P(k)$ は以下のようにして求める．

$$P(k) = \frac{\sum_m |r(m) f(m,k)| \, r(m) f(m,k)}{\sqrt{\sum_m f(m,k)}} \qquad (4)$$

本研究ではこのようにして機能ごとにポイントを導出し，与えられた値の大きいものから順に解析対象としたマルウェアがもつ機能である可能性が高いと考える．

7　実験結果

実験対象として本研究では 2012 年 6 月 15 日に NICT で採取された検体の内パックされていない 235 検体を用いた．本章では 2 検体に対してそれ自体を除くすべての検体との類似度を導出した結果と 2 検体に対して機能推定を行った結果を示す．

7.1　マルウェアの類似度導出

図 4 および図 5 にそれぞれ異なった検体に対してそれを除いた他のすべての検体との類似度を導出した結果を示す．ここではマルウェアのバイナリデータ全体からバイトコードの出現頻度を導出した場合と，実行コードを含むかもしくは実行可能なセクションから 0x0f の後のバイトのみを抽出し，取得した出現頻度を用いて類似度を導出した場合を比較している．凡例にある条件なしはバイナリ全体からバイトコードの出現頻度を求めて類似度を導出した場合であり，条件付きは実行コードを含むか若しくは実行可能なセクションから 0x0f の後のバイトコードの出現頻度を求め，類似度を導出した場合である．図からも分かるようにバイナリ全体の出現頻度から類似度を取得する場合に比べて条件を指定してバイトコードの出現頻度を求め，類似度を導出した場合の方がより大きな偏差を得られていることが分かる．また，類似度が条件を指定した場合でも高い値になっている検体はアンチ

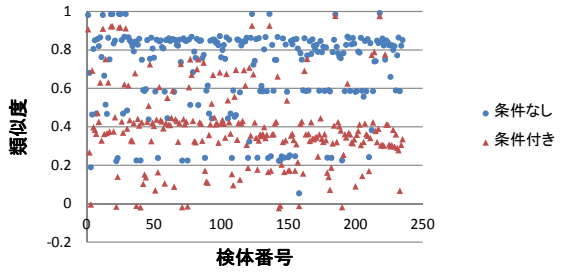

図 4　バイトコードの出現頻度による類似度

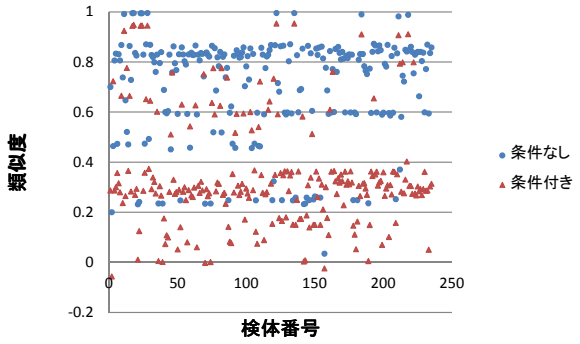

図 5　バイトコードの出現頻度による類似度

ウィルスベンダによる科名が一致するか，そうでなくても NONSTOP の解析結果から同様の動作をする検体であることが分かった．

7.2　マルウェアの機能推定

本研究で機能推定の対象とした 2 検体をそれぞれ検体 A，検体 B として各々について機能推定を行った．検体 A は Trend Micro のスキャン結果により WORM_RBOT.GD と判定された検体である．表 3 に検体 A が持つ機能と持たない機能についてそれぞれの機能ポイントを導出した結果を示す．検体 A が持たない機能に関してはポイントの高かったものの上位 10 までの機能とそのポイントを示す．検体 A が持つ機能の内で最もポイントの低かったものは sid が 00000018 の 0.747 であった．一方で，検体 A が持たない機能の内最も高かった機能は sid が 00700001 の 0.6339 であり，検体 A がもつ機能がすべてポイントの上位を占めていることが分かる．

次に検体 B の機能推定結果を行った結果を示す．検体 B は Trend Micro のスキャン結果により，WORM_ALLAPLE.IK とされた検体である．検体 A と同様に検体 B についても機能推定結果から得られたポイントを所有機能と非所有機能に分けて表 4 に示す．検体 B では検体 A に比べて全体的に高いポイントとなった．検体 B の所有機能の内，最もポイントが低かったものは sid が 00000017 の 1.462 であり，非所有機能の内で最もポイントが高かった機能は sid が 00000018 の 1.254 である．所有している機能がポイントの上位を占め，所有機能と非所有機能でポイントに差があることが分かる．

所有機能		非所有機能	
sid	機能ポイント	sid	機能ポイント
00000019	0.749665071	00700001	0.633911020
00600001	0.911473661	00000005	0.570638807
04000002	0.946379398	00000017	0.514151110
00000020	0.754539398	00000006	0.489977201
00000002	0.969784830	00000015	0.479262740
00000001	0.902697552	00200001	0.464724986
01000001	0.915780141	00000009	0.323582585
00000018	0.746906183	02000001	0.229645204
00800001	0.851088106	30000001	0.185966788
20000002	0.948768813	00010001	0.158242616
40000001	0.948768813		

表 3　検体 A の機能ごとのポイント

所有機能		非所有機能	
sid	機能ポイント	sid	機能ポイント
00000002	1.485718273	00000018	1.253669641
00000001	1.526406611	00000019	1.224473291
00200001	2.085897624	00000020	0.702363894
00000005	1.871111443	00800001	0.633555126
00300001	3.439351729	40000001	0.619847670
00600001	1.520092176	20000001	0.619847670
00000009	2.721216889	04000002	0.613280564
01000001	1.512314477	00000012	0.023184534
00700001	1.705192191	00000007	0.021042905
00000006	2.050798102	30000001	0.012372655
00000015	1.480228080		
00000017	1.462352985		

表 4　検体 B の機能ごとのポイント

8　まとめ

本研究ではマルウェア間の類似度を導出する手法および，導出した類似度を用いてマルウェアの機能推定をおこなう手法を提案した．マルウェア間の類似度導出にはバイトコードの出現頻度から正規化相互相関を用いた．取得するバイトコードはマルウェアのバイナリデータの実行コードを含むかもしくは実行可能セクションの 0x0f の次のバイトのみに限定した．バイトコードを抽出する対象として実行コードを含むかもしくは実行可能セクションとすることによってマルウェアの挙動に応じた類似度が導出できると考えられる．また，さらにマルウェアごとの特徴を抽出するめ 2byte のオペコードに用いられる 0x0f のバイトに着目した．0x0f の後の 1byte のみを取得していき，バイトコードごとの出現頻度を用いることでより大きな偏差を持つ類似度の導出を可能にした．類似度を導出する実験では，すべてのバイトコードの出現頻度を用いて類似度を導出した結果との比較を行った．この結果亜種関係にあるかもしくは機能に類似性のみられる

マルウェアの類似度はどちらとも高かったもののすべてのバイトコードの出現頻度を用いた類似度が低かった検体に関してはより低い類似度を導出し，より大きな偏差を得る結果となった．機能推定にはある検体に対して多数の解析済み検体との類似度を導出し，利用する．機能ごとに類似度から得られるポイントを加算していき，最終的に機能ごとの出現頻度を考慮することによってポイントを与える．本研究ではポイントの高い機能から対象とした検体が持つ確率の高い機能と考える．機能推定の実験では実際に検体が保有する機能がポイントの上位を占め，これにより機能推定が可能であることが示された．

9 今後の課題

マルウェアの類似度導出では 0x0f の後にある 1byte を取得していき，バイトコードごとの出現頻度を求めることにより，偏差の大きい類似度導出が可能となった．一方で類似度を用いた機能推定は上位を検体が保有する機能が占めたものの 2 つの課題が残っている．まず課題として挙げられることは機能推定の対象とした検体によって全体的にポイントの数値が変動することである．これによって一意にあるポイントが高いか低いかを決定することができず，現状ではヒューリスティックに機能推定を行う必要がある．またもう一つの課題として所有機能と非所有機能のポイントの差が大きくないことがあげられる．所有機能と非所有機能のポイントの差は機能推定の精度に大きく関わるためより大きな差を与える必要がある．

謝辞

(独) 情報通信研究機構ネットワークセキュリティ研究所サイバーセキュリティ研究室各位のご助言，ご協力に感謝する．

参考文献

[1] Intel 64 and IA-32 Architectures Software Developer’s Manual,“ http://www.intel.com/ ”
[2] Symantec Security Response,“ http://www.symantec.com/ ”
[3] 東結香，中津留勇，真鍋敬士，猪俣敦夫，藤川和利，砂川秀樹，“コードに基づいたマルウェアの機能推定に関する研究，”SCIS2011
[4] 岩村誠，伊藤光恭，村岡洋一，“機械語命令列の類似性に基づく自動マルウェア分類システム，” 情報処理学会論文誌，vol.51，No.9，pp1-11，2010
[5] 小篠裕子, 勝手壮馬, 森井昌克, 中尾康二, “マルウェアの類似度による機能推定，”CSS2009
[6] 新井悠，岩村誠，川古谷裕平，青木一史，星澤裕二，“アナライジングマルウェア，” オイラリージャパン