Computer Security Symposium 2017
23 - 25 October 2017

データ圧縮アルゴリズムによるマルウェア分類および亜種判定について

武智 聡平[1]　甲斐 博[1]　森井 昌克[2]

概要：マルウェアを詳細な解析をすることなく，できるだけ短時間で，かつ自動的に分類することは重要な課題となっている．従来から，その動的解析より観測される API コールの特徴によって分類することが行われ，その精度が競われている．本研究では API コールの時系列発生順序に着目し，その特徴によって分類を試みる．独創的な点はその特徴の計算のためにユニバーサル圧縮手法として知られる LZ77 アルゴリズムを用いる点である．LZ77 はスライド辞書法の一種であり，スライド辞書のサイズを変更することにより，API コールの出現特性に応じた圧縮率となる．この特性を用いて，種々の辞書サイズによる圧縮率によって分類を試みる．分類するための方法としては SVM を用い，分類の精度を求め，本提案手法の有効性を示す．

キーワード：マルウェア解析，分類，亜種，データ圧縮アルゴリズム，SVM

Malware Classification Based on Data Compression Algorithm

Sohei Takechi[1]　Hiroshi Kai[1]　Masakatu Morii[2]

Abstract: The method to classify automatically the malwares as quickly as possible without their detailed analysis has become an important research subject. Traditionally, the malware classification has been performed by the characteristics of the APIs observed from the dynamic analysis, and the researchers compete the accuracy. In this research, we focus on the time order of the API call, and try to classify the malware variant by its characteristics. The originality of our research is to use the LZ77 algorithm, known as an universal data compression algorithm, to derive the characteristics. The LZ77 is a slide dictionary method, and its data compression ratio is depend on the size of the slide dictionary and the time order of the API call. We classify the malware variants by the characteristics derived from the data compression ratios, which are computed by the different kind of sizes of the slide dictionary. We perform the malware variant classification based on the SVM algorithm, and examine the accuracy of our method. Experimental results show that the proposed method is effective.

Keywords: Malware Analysis, Classification, Malware Variant, Data Compression Algorithm, SVM

1. はじめに

インターネットの利用者にとってマルウェアの増加は問題になっており，AV-TEST のレポートによると，新種のマルウェアは近年では年間に 1 億種類以上発見されている [1].

マルウェア対策のためのアンチウィルスソフトでは「ウイルス定義データベース」を利用してマルウェアを検出する．ウィルス定義データベースによる検出から逃れるため多くの亜種が作成されており，マルウェアの発見数が膨大になる要因の 1 つである．

マルウェアの亜種が増加する中で，マルウェアを詳細な解析をすることなく，できるだけ短時間で，自動的に分類することは重要な研究課題の一つである．従来から，動的

[1] 愛媛大学大学院理工学研究科
Graduate School of Science and Engineering, Ehime University
[2] 神戸大学大学院工学研究科
Graduate School of Engineering, Kobe University

解析より観測される API コールの特徴によって分類することが行われ，その精度が競われている．例えば，多次元ベクトルでマルウェアの挙動を表現しハミング距離で亜種判定を行う研究 [10] や，API コール列を Paragraph Vector を用いて表現し亜種判定を行う手法 [5] などが提案されている．

本稿では，API コールの時系列発生順序に着目し，その特徴によって分類を試みる．独創的な点はその特徴の計算のためにユニバーサル圧縮手法として知られる LZ77 アルゴリズムを用いる点である．LZ77 はスライド辞書法の一種であり，スライド辞書のサイズを変更することにより，API コールの出現特性に応じた圧縮率となる．この特性を用いて，種々の辞書サイズによる圧縮率によって分類を試みる．分類するための方法としては SVM を用い，分類の精度を求め，本提案手法の有効性を示す．FFRI dataset を用いた亜種判定の実験を行い，平均として約 82%で亜種分類が成功することを示す．

2. マルウェアの動的解析と FFRI dataset

マルウェア解析手法としては大きく分類して表層解析，静的解析，動的解析がある．表層解析はファイルの情報などの収集を目的として行われ，静的解析では逆アセンブラなどによりバイナリファイルを用いてマルウェアの特徴を解析する．一方で，動的解析は検体を実際に実行し，プロセスの起動，ファイルの書き換えやダウンロード，通信パケットの記録などの不正な動作を記録し，マルウェアの振る舞いを解析するものである．

動的解析はその動作を検証するために監視ツールを用いて記録を行い，隔離された環境下で実施する必要がある．そのため動的解析のためのシステムが存在し，オープンソースマルウェア解析システムである Cuckoo Sandbox などが知られている．

また，研究用データセットとして配布されている MWS Dataset 2016 には，マルウェアの動的解析結果の FFRI Dataset 2013〜2016 が含まれている．このデータセットは株式会社 FFRI が独自に収集したマルウェアの動的解析ログである．

FFRI Dataset 2013 は 2641 検体，FFRI Dataset 2014 および 2015 はそれぞれ 3000 検体，FFRI Dataset 2016 は 8243 検体に対する動的解析のログファイルが存在している．FFRI Dataset の動的解析は Cuckoo Sandbox を用いて行われている [2]．また，FFRI Dataset 2014 では Cuckoo Sandbox での解析に加えて FFRI yarai analyzer Professional の解析結果ログも含まれている [3]．

動的解析結果には virustotal, behavior, network など 13 種類の項目に分類された結果が記録されている．

例えば virustotal には，VirusTotal の検査履歴との照合結果が示されており，Kaspersky, ESET などアンチウィルスベンダーが分類した亜種名が記載されている．

また behavior には，実行された API 名や時刻などが記載されており，実行された時系列順に並んでいる．例えば，“Trojan.Win32.Agent.rxht” について，behavior の中に現れる API コール列を取得した例を**図 1** に示す．

LdrLoadDll LdrGetDllHandle DeviceIoControl NtAllocateVirtualMemory LdrLoadDll IsDebuggerPresent LdrLoadDll LdrLoadDll LdrLoadDll LdrLoadDll LdrLoadDll NtCreateMutant NtOpenSection SetWindowsHookExA SetWindowsHookExA CreateThread NtAllocateVirtualMemory NtAllocateVirtualMemory NtAllocateVirtualMemory NtAllocateVirtualMemory NtOpenSection NtClose SetWindowsHookExA SetWindowsHookExA RegOpenKeyExW RegQueryValueExW RegCloseKey RegCloseKey LdrGetDllHandle ...

図 1 “Trojan.Win32.Agent.rxht” の API コール列
Fig. 1 The API call sequence of “Trojan.Win32.Agent.rxht”

ここで “Trojan.Win32.Agent.rxht” は Kaspersky によるマルウェア分類名である．Kaspersky の命名規則は次の通りである．

[Prefix:]Behaviour.Platform.Name[.Variant]

ここで，Prefix は対象を検知したサブシステム，Behaviour は検知された対象，Platform はプログラムコードが実行される環境，Name は検知された対象のファミリー名，Variant は亜種を指す [4]．

3. マルウェアの動的解析結果とそれに基づく分類

API コール列を利用したマルウェアの動的解析結果に基づくマルウェア分類では，呼び出された API 名の数による比較や，呼び出された API 名の出現パターンによる比較でマルウェアが分類される．

例として “Backdoor.Win32.Androm.bknn” と “Backdoor.Win32.Androm.affh” の API コール列を**図 2** と**図 3** に示す．これらの同じマルウェアファミリーに属する亜種では，“LdrGetProcedureAddress” と呼ばれる API がどちらも 17 回出現している．また，1 番目に呼び出された API の “NtOpenDirectoryObject” から，22 番目に呼び出された API の “LdrGetProcedureAddress” までの API の出現するパターンが全く同じであるという特徴を持つ．

一方で，全く別のマルウェアファミリーの亜種 “Trojan-Spy.Win32.Zbot.hpyq” の API コール列の一部を**図 4** に示す．このマルウェアには “LdrGetProcedureAddress” と呼ばれる API の出現回数は 407 回出現しており，先ほど述べたマルウェアファミリー “Backdoor.Win32.Androm” における API の出現回数とは全く異なる．マルウェアファミリー “Backdoor.Win32.Androm” において観測できた API コール中の特徴的なパターンはマルウェアファミリー “Trojan-Spy.Win32.Zbot.hpyq” では見られない．

以上の例に見られるように，マルウェアファミリーが異

NtOpenDirectoryObject NtFreeVirtualMemory LdrLoadDll LdrGetProcedureAddress LdrGetProcedureAddress LdrLoadDll LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress NtProtectVirtualMemory NtOpenFile NtOpenFile NtOpenFile NtOpenFile NtOpenFile NtOpenFile NtOpenFile NtOpenFile NtOpenFile NtSetInformationFile NtOpenFile NtOpenFile CreateProcessInternalW NtOpenFile WriteProcessMemory NtOpenFile WriteProcessMemory WriteProcessMemory NtGetContextThread NtSetContextThread NtResumeThread ExitProcess

図 2 “Backdoor.Win32.Androm.bknn” の API コール列
Fig. 2 The API call sequence of “Backdoor.Win32.Androm.bknn”

NtOpenDirectoryObject NtFreeVirtualMemory LdrLoadDll LdrGetProcedureAddress LdrGetProcedureAddress LdrLoadDll LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress NtOpenFile NtOpenFile NtOpenFile NtSetInformationFile CreateProcessInternalW WriteProcessMemory WriteProcessMemory WriteProcessMemory NtGetContextThread NtSetContextThread NtResumeThread ExitProcess

図 3 “Backdoor.Win32.Androm.affh” の API コール列
Fig. 3 The API call sequence of “Backdoor.Win32.Androm.affh”

なれば，異なる API コール列の特徴を持ち，マルウェアファミリーが同じであれば何らかの関係があると考えられる．

佐藤らが提案した手法は，API コール列が持つパターンを Paragraph Vector を用いて表現しマルウェアの亜種判定を行っている．彼らは Paragraph Vector を作成するために以下の 3 通りのコーパスを提案しており，それぞれ手法 1，手法 2，手法 3 と呼ぶ．

手法 1 API 名のみを用いた Paragraph Vector
手法 2 API 名，API の引数名，および API の引数値を用いた Paragraph Vector
手法 3 API 名および引数値の長さを用いた Paragraph Vector

これらのいずれかの手法を用いて作成した Paragraph Vector に対して，各マルウェアファミリーについて 1 つの分類器を作成し実験を行う．分類器は SVM あるいはランダムフォレストを用いて作成される．ここでは SVM を用いた場合について佐藤らの手法 [5] を説明する．

RegOpenKeyExA RegOpenKeyExA NtAllocateVirtualMemory NtAllocateVirtualMemory NtAllocateVirtualMemory LdrGetProcedureAddress LdrLoadDll LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetProcedureAddress LdrGetDllHandle NtAllocateVirtualMemory NtAllocateVirtualMemory LdrLoadDll LdrGetProcedureAddress . . .

図 4 “Trojan-Spy.Win32.Zbot.hpyq” の API コール列
Fig. 4 The API call sequence of “Trojan-Spy.Win32.Zbot.hpyq”

佐藤らの実験では FFRI dataset 2013〜2016 を対象として実験されており，FFRI dataset から得られる API コール列を Paragraph Vector で表現して，マルウェアの特徴を SVM で学習させる．SVM への教師データとして，1 つの検体から得られる Paragraph Vector に対してラベル付けが行われ，SVM に入力される．ラベル付けはそのマルウェアファミリーに属する検体の場合は 1，そうでない検体の場合は-1 をつけたものとされている．

実験では亜種の数が 100 検体以上あるマルウェアファミリー 22 種類に対して分類器が作成され手法の有効性が検討される．4 分割交差検証により実験の検証が行われ，その推定結果では，手法 1 は平均で 81.94%，手法 2 は平均で 81.08%，手法 3 は平均で 82.06%の割合で推定を行うことができると報告されている [5].

4. LZ77 を用いた特徴ベクトルの提案

マルウェアに感染した端末の通信ログの特徴抽出 [6] や通信パケット情報の異常検知のための特徴抽出 [7] においてデータ圧縮アルゴリズムを利用する研究が行われている．本研究ではデータ圧縮アルゴリズムの新しい応用として，マルウェアの亜種分類について検討した．

データ圧縮ではファイル中のデータのパターンなどを利用してファイルサイズを小さくすることができる．本稿では，データ圧縮の結果を利用して，マルウェアの動的解析結果の API コール列中の同じパターンの繰り返しをマルウェアの特徴とするため，以下で述べる特徴ベクトルを定義する．

特徴ベクトルの計算のために，辞書に基づいた符号化の一種である LZ77 アルゴリズム [8] を用いる．LZ77 では，現在符号化を行っている記号の前後の何記号かを保存する

スライド窓と呼ばれるバッファを用意し，このバッファを辞書として利用することで符号化を行う．符号化における辞書の参照は，現在符号化の対象となっている位置から始まる記号列と一致する（スライド窓中の記号列で最も長い）ものを探すことを基本としている．スライド窓の長さは固定長であり，アルゴリズム実行時に決定する．

圧縮の対象とする入力データが 1 文字単位の語で構成される場合を考えてアルゴリズムの説明を行う．まずはアルゴリズムの中で用いる語句を説明する．

入力ストリーム　データ圧縮の対象となる入力データ列

符号化位置　符号化を行う入力ストリーム上の位置（先読みバッファの先頭）

先読みバッファ　入力ストリームから記号を読み込む固定長のバッファ

辞書　先読みバッファで処理されたあとの入力ストリームが書き込まれる固定長のバッファ（初期状態は空である．辞書と先読みバッファをあわせてスライド窓と呼ばれる．）

ポインタ　先読みバッファの先頭から始まる文字列と最長一致するスライド窓中の文字列の先頭の位置とその文字列の長さ

このとき LZ77 のアルゴリズムの手順は以下のようにまとめられる．

(1) 入力ストリームの開始位置に符号化位置を設定する．
(2) スライド窓の中で先読みバッファと最長一致する文字列を見つける．
(3) そのような文字列が見つかれば，ポインタを出力し，符号化位置を最長一致した文字列の長さ分だけシフトさせて先読みバッファの最初の 1 文字を出力する．もし見つからなければ，空ポインタを出力し 1 文字分だけ符号化位置をシフトして先読みバッファの最初の 1 文字を出力する．
(4) 先読みバッファが空でなければステップ（2）に戻る．

つまり処理の手順としては，スライド窓に入力ストリームから文字列を読み込んでは，辞書を参照しながら先読みバッファの中で繰り返されるパターンを見つけて符号化を行う処理をする．

ここで，動的解析結果より抽出した API コール列を保存したファイルを A とし，スライド窓サイズ s の LZ77 により圧縮したファイルを $f_s(A)$ とする．またファイル A のバイト数を $B(A)$ とすると，スライド窓サイズ s に関する圧縮率 $\rho_s(A)$ を次のように定義する．

$$\rho_s(A) = \frac{B(f_s(A))}{B(A)} \times 100.$$

この時，API コール列 A に対して n 種類のスライド窓のサイズを $s_i, 1 \leq i \leq n$ とした時，本稿で用いる n 次元特徴ベクトル $V(A)$ を

$$V(A) = (\rho_{s_1}(A), \rho_{s_2}(A), \cdots, \rho_{s_n}(A))$$

と定義する．

本研究では，特徴ベクトル $V(A)$ の計算を次のような手順で求める．

(1) FFRI dataset からの API コール列の取得
(2) API コール列を 2 バイトコード列に変換
(3) 特徴ベクトルの作成

API コール列は図 1 に示したように API が呼び出された順に API の名前を並べることにより得られる．

API 名のままではパターンを利用した圧縮を行うことができない．そのため API の名前に対して一意に対応がつく ID（整数値）を割り当てる．FFRI dataset 2013〜2016 では現れる API の種類数が 256 以上あるため 2 バイトで整数値を表現しそれを 1 文字とする．

以上のように作成したファイルに対して LZ77 を用いた圧縮を行う．例えば，$n = 4$ の場合の亜種名 “Worm.Win32.WBNA.hae” の特徴ベクトルを示す．

(77.95, 77.51, 76.86, 74.45)

ここで左からスライド窓のサイズが 2B, 3B, 4B, 5B とした時のデータである（B はバイトを表す）．計算には Python を用いてデータの表示に str 関数を用いたため圧縮率は 12 桁の浮動小数点数で得られるが，ここでは 4 桁で丸めて表示している．同じ亜種の “Worm.Win32.WBNA.bul” では，

(77.92, 76.80, 76.13, 74.32)

となり “Worm.Win32.WBNA.hae” と非常に近い特徴ベクトルが得られていることがわかる．また，マルウェアのファミリーが異なる例として，“Trojan.Win32.Llac.ddxv” に対する特徴ベクトルを示す．

(80.15, 63.74, 63.74, 63.74)

このように “Worm.Win32.WBNA” のファミリーの特徴ベクトルとは大きく異なる結果が得られ，本研究で提案する圧縮率を用いた特徴ベクトルによりマルウェア亜種の特徴が表現できていることを実験的に確認することができる．

5. SVM と特徴ベクトルを用いたマルウェア亜種分類の方法

本研究で用いる亜種分類手法は佐藤らの手法 [5] と同じであるが，Paragraph Vector の代わりに前節で定義した特徴ベクトルを用いる点が異なる．すなわち，入力とする特徴ベクトルが，あるマルウェアファミリーに属するかそうでないかを分類することを目的とし，学習用特徴ベクトルを SVM に入力しマルウェアの特徴を学習させる．

SVM の分類ラベルとして，特徴ベクトルがマルウェア

91.2510936133	89.4138232721	88.3639545057	86.0017497813
83.3770778653	80.927384077	77.4278215223	70.5161854768
69.3788276465	69.1163604549	68.7664041995	66.0542432196
65.0043744532	65.0043744532	64.1294838145	63.9545056868
63.3420822397	62.5546806649	62.2922134733	61.854768154
61.7672790901	61.154855643	60.3674540682	59.4925634296
59.3175853018	59.1426071741	58.0927384077	57.8302712161
57.4803149606	57.1303587052	91.2510936133	89.4138232721
88.188976378	85.5643044619	83.1146106737	81.8022747157
76.8153980752	70.2537182852	69.4663167104	69.4663167104
68.5914260717	65.8792650919	64.47944007	64.1294838145
63.2545931759	63.0796150481	62.9921259843	62.5546806649
62.3797025372	61.7672790901	61.5923009624	60.8923884514
60.542432196	59.6675415573	59.4925634296	59.230096238
57.7427821522	57.7427821522	57.217847769	56.605424322

図 5 “Trojan.Win32.Agent.rxht” の特徴ベクトル
Fig. 5 The characteristics vector of “Trojan.Win32.Agent.rxht”

ファミリーに属する場合は 1、そうでない場合は-1 を，特徴ベクトルの先頭に付加し，各マルウェアファミリー毎に学習用特徴ベクトルを作成し，分類器を作成する．

次節で述べる実験では，スライド窓のサイズは 2B，3B，4B，5B，6B，7B，8B，9B，10B，12B，14B，16B，18B，20B，22B，24B，26B，28B，30B，33B，36B，39B，42B，45B，50B，55B，60B，65B，70B，80B と取り，これら 30 種類のスライド窓のサイズにおける圧縮率を求める．同様に API コール列を逆順に並べたものに対しても同じ処理を行い，各スライド窓のサイズにおける圧縮率を求める．求めた圧縮率を API コール列の正順，逆順という順番で，スライド窓サイズの小さい順に並べ，60 次元の特徴ベクトルを用いる．例えば “Trojan.Win32.Agent.rxht” の特徴ベクトルを**図 5** に示す．マルウェアファミリー “Trojan.Win32.Agent” の分類器を作成する場合は，学習用特徴ベクトルとして図 5 のベクトルの先頭に 1 を付加したベクトルを作成し SVM で利用する．

6. 提案手法によるマルウェア亜種推定の実験

本実験では，動的解析結果として FFRI dataset 2013〜2016 のデータを対象として実験を行う．但し，前述したように，FFRI Dataset 2014 では Cuckoo Sandbox での解析に加えて FFRI yarai analyzer Professional の解析結果ログも含まれているが，解析方法を統一するため，Cuckoo Sandbox による解析結果のみを用いる．また，FFRI Dataset 2016 には Windows10 と Windows8.1 の解析結果が含まれているが，本稿では Windows10 の動的解析結果のみを用いる．また，亜種判定における正解として，Kaspersky が命名した名称を基準として分類を行う．

API のコールパターンに着目し SVM を用いた判定を行うため，全 16887 検体のうち，API コールが取得できるマルウェアであり，亜種の数が 100 検体以上ある 22 種類のマルウェアファミリーを判定対象とした．22 種類のマルウェアファミリーの名前とマルウェアファミリーに含まれる検体の数の一覧を**表 1** に示す．

表 1 マルウェアファミリー名
Table 1 The malware family names

マルウェアファミリー名	亜種検体数
Trojan.Win32.Waldek	687
Trojan-Spy.Win32.Zbot	581
Trojan.Win32.Yakes	577
Backdoor.Win32.Androm	445
Trojan.Win32.Agent	434
Trojan.Win32.Inject	351
Trojan.Win32.WBNA	308
Hoax.Win32.ArchSMS	304
Trojan-PSW.Win32.Fareit	272
Trojan-PSW.Win32.Tepfer	218
Backdoor.Win32.DarkKomet	206
Trojan-Ransom.Win32.Foreign	204
Trojan-Dropper.Win32.Injector	185
Trojan.Win32.Jorik	171
Packed.Win32.Tpyn	163
Trojan.Win32.Kovter	145
Trojan.Win32.Scar	142
Downloader.Win32.LMN	123
Worm.Win32.Vobfus	120
Backdoor.Win32.Matsnu	110
Trojan-Downloader.Win32.Upatre	108
Trojan.Win32.Llac	100

佐藤らの手法 [5] の結果と条件を揃えるため，判別するマルウェアファミリーとそうではないマルウェアファミリーの検体数が 21:1 となるように検体数を揃えて，マルウェアファミリー毎に 22 種類の分類器を作成する．

また亜種判定には 4 分割交差検証を行う．つまり 4 分割したデータセットのうちの 1 つをテストセットとし，残りをトレーニングセットとする．4 個に分割されたデータセットそれぞれをテストセットとして 4 回検証を行う．

ここで，SVM には LIBSVM[9] を用い，グリッドサーチを行い最適な数値を求めた．

提案手法の推定精度の結果を**表 2** に示す．また同様の実験を Paragraph Vector を用いて行っている佐藤らの手法 [5] による実験結果（API 名のみを扱った手法 1 の結果）も示す．

表 2 より 22 種類のマルウェアファミリーのうち 9 種類のマルウェアファミリーについて，佐藤らの手法より推定精度が向上したことがわかる．また，推定率が 90%を超えたマルウェアは 22 種類中 6 種類となった．この中で最も推定率が高いものは “Worm.Win32.WBNA” の 94.05%であり，最も推定率が悪いものは “Trojan.Win32.Agent” の 63.69%となった．

表 2 提案手法の精度

Table 2 The accuracy of the proposed method

マルウェアファミリー名	手法 1[5] 推定率（%）	提案手法 推定率（%）
Trojan.Win32.Jorik	90.48	85.71
Worm.Win32.WBNA	91.87	94.05
Trojan.Win32.Agent	72.02	63.69
Worm.Win32.Vobfus	94.05	93.45
Trojan.Win32.Yakes	81.25	71.43
Trojan-PSW.Win32.Tepfer	75.60	86.90
Trojan-Spy.Win32.Zbot	74.80	70.83
Trojan-Dropper.Win32.Injector	70.24	70.24
Hoax.Win32.ArchSMS	94.84	91.07
Trojan.Win32.Inject	69.64	75.60
Trojan-Ransom.Win32.Foreign	86.31	77.38
Trojan.Win32.Scar	75.00	73.21
Trojan.Win32.Llac	76.19	81.55
Backdoor.Win32.DarkKomet	81.25	83.93
Downloader.Win32.LMN	89.29	91.67
Backdoor.Win32.Androm	79.88	73.81
Trojan-PSW.Win32.Fareit	79.37	79.17
Trojan-Downloader.Win32.Upatre	76.79	91.67
Packed.Win32.Tpyn	79.76	85.71
Trojan.Win32.Kovter	90.48	93.45
Trojan.Win32.Waldek	91.59	88.10
平均	81.94	82.03

ここで佐藤らの手法と比べて大きく向上している “Trojan-Downloader.Win32.Upatre” について検討する．このマルウェアファミリーでは，スライド窓が大きくなるに従って圧縮率が高くなる．例えば “Trojan-Downloader.Win32.Upatre.foqk” と “Trojan-Downloader.Win32.Upatre.fpbh” の特徴ベクトルをそれぞれ**図 6** と**図 7** に示す．特徴ベクトルの間の L^∞ ノルムを取ると約 0.3647 でありマルウェア亜種の特徴ベクトルの値が似ている．同様な特徴の特徴ベクトルをもつマルウェア亜種が含まれるため推定率が高くなったと考えられる．

一方で，“Trojan.Win32.Agen” は推定率が大きく低下してしまっている．このマルウェアファミリーには，API コール列のパターンが全く異なる亜種が多く含まれている．例えば，“Trojan.Win32.Agent.rxht” と “Trojan.Win32.Agent.wlgm” の API コール列を比較する．“Trojan.Win32.Agent.rxht” の API コール列は図 1 に示しており，“Trojan.Win32.Agent.wlgm” の API コール列を**図 8** に示す．

“Trojan.Win32.Agent.wlgm” の API コール列では，“NtAllocateVirtualMemory” と呼ばれる API が 2 回だけ呼び出されたあと，“WriteConsoleA” が 80 回呼び出されるものになっている．これらの API コール列を比較してわかるようにパターンが全く異なる．そのためマルウェア

84.4483298058	83.3716757155	74.4087604675	72.2922609736
70.359804914	65.685101684	59.3448053741	57.1270819913
48.7347013895	45.3759087145	44.621330634	43.6274960891
43.2041961903	42.8085028067	42.7164810895	41.8422747768
37.2135824054	35.7044262446	35.4099567498	34.986656851
34.3885156897	34.1308548818	33.5603202356	33.3486702862
32.0971749333	31.977546701	31.6830772062	31.4622250851
31.2321707923	30.7720622067	84.4483298058	83.3808778872
74.4731756695	72.3014631453	70.4794331462	65.8967516334
59.8325204748	57.651605779	49.3972577528	45.8912303304
44.9618109874	43.8115395233	43.2962179074	42.9189288672
42.541639827	41.9250943223	37.1491672035	35.5387871538
35.308732861	34.8210177602	34.3241004877	34.0940461949
33.6891506396	33.4406920033	32.170792307	32.0603662464
31.8579184688	31.5450446305	31.2873838226	30.7536578633

図 6 “Trojan-Downloader.Win32.Upatre.foih” の特徴ベクトル

Fig. 6 The characteristics vector of “Trojan-Downloader.Win32.Upatre.foih”

84.0836453292	83.0152497489	74.1302164186	72.0299516026
70.0849237513	65.4734727422	59.2548625696	57.1089398228
48.7535384896	45.4113779564	44.635193133	43.6215870697
43.1924025203	42.7723495571	42.6627705232	41.7222171491
37.1472924847	35.6588439412	35.3666331842	34.9100538764
34.3256323623	34.06994795	33.5129211944	33.2846315405
32.0062094786	31.8874988586	31.5678933431	31.3487352753
31.0747876906	30.5360241074	84.0836453292	83.0335129212
74.212400694	72.0025568441	70.185371199	65.6652360515
59.684047119	57.5655191307	49.356223176	45.8405625057
44.9000091316	43.758560862	43.2563236234	42.8727970048
42.4892703863	41.8226645968	37.1107661401	35.5127385627
35.3118436672	34.8461327733	34.3530271208	34.1247374669
33.7320792622	33.4763948498	32.1614464432	32.0336042371
31.8692356862	31.5404985846	31.3030773445	30.6638663136

図 7 “Trojan-Downloader.Win32.Upatre.fpbh” の特徴ベクトル

Fig. 7 The characteristics vector of “Trojan-Downloader.Win32.Upatre.fpbh”

NtAllocateVirtualMemory NtAllocateVirtualMemory WriteConsoleA ...

図 8 “Trojan.Win32.Agent.wlgm” の API コール列

Fig. 8 The API call sequence of “Trojan.Win32.Agent.wlgm”

ファミリー内に明確な特徴が現れず，推定率が低下したのではないかということが考えられる．

7. おわりに

本研究ではマルウェアの動的解析結果に含まれる API コール列をデータ圧縮アルゴリズムである LZ77 を用いて

圧縮し，30 種類のスライド窓で圧縮した場合の圧縮率を特徴ベクトルとする方法を提案した．機械学習の手法の一種である SVM を用いることで，平均で約 82%の精度でマルウェアの亜種を推定できることを示した．

今後の課題として，推定率が 70%以下になるようなマルウェアファミリーに対して有効な手法の検討を行うことがあげられる．

参考文献

[1] AV-TEST, Statistics, Malware, 入手先 ⟨https://www.av-test.org/en/statistics/malware/⟩ (2017-08-19)

[2] 神薗雅紀，畑田充弘，寺田真敏，秋山満昭，笠間貴弘，村上純一: マルウェア対策のための研究用データセット MWS Datasets 2013 , 情報処理学会シンポジウムシリーズ（コンピュータセキュリティシンポジウム 2013 論文集）, Vol.2013, No.4, pp.1-8(2013).

[3] 株式会社 FFRI: FFRI Dataset 2014 のご紹介，入手先 ⟨http://www.iwsec.org/mws/2014/files/FFRI_Dataset_2014.pdf⟩ (2017-08-05).

[4] Securelist: Rules for naming, 入手先 ⟨https://securelist.com/threats/rules-for-naming/⟩ (2017-08-08).

[5] 佐藤拓未，後藤滋樹，武部嵩礼: Paragraph Vector を用いたマルウェアの亜種推定法，情報処理学会シンポジウムシリーズ（コンピュータセキュリティシンポジウム 2016 論文集）, Vol.2016, No.2, pp.298-304(2016).

[6] 岡野靖，神谷和憲，谷川真樹: データ圧縮アルゴリズムを用いたマルウェア感染通信ログの判定，情報処理学会シンポジウムシリーズ（コンピュータセキュリティシンポジウム 2016 論文集）， Vol.2016, No.2， pp.640-646, 2016.

[7] 衛藤公希，小野廣隆，山下雅史，竹内純一，文字列圧縮を用いたネットワークセキュリティにおけるインシデント検出，情報科学技術フォーラム講演論文集， Vol.11, No.1, pp.9-16, 2012.

[8] Ziv J. and Lempel A.: A Universal Algorithm for Sequential Data Compression, IEEE Trans. Inf. Theor., Vol.23, No.3, pp.337-343(1977).

[9] LIBSVM – A Library for Support Vector Machines, 入手先 ⟨https://www.csie.ntu.edu.tw/ cjlin/libsvm/⟩ (2017-08-05).

[10] 堀合啓一，今泉隆文，田中英彦: マルウェア亜種の動的挙動を利用した自動分類手法の提案と実装，情報処理学会論文誌，Vol.50, No.4, pp.1321-1333(2009).