自動化されたマルウェア動的解析システムで収集した大量 API コールログの分析

藤野 朗稚 †　　森 達哉 †

† 早稲田大学
169-8555　東京都新宿区大久保 3-4-1
{fujino,mori}@nsl.cs.waseda.ac.jp

あらまし 本研究は自動化されたマルウェア動的解析システムが出力する大量の API コールを機械学習のフレームワークを用いて分析し，マルウェアの分類と特徴抽出を試みる．約 2,600 個のマルウェア検体の API コールデータに本研究で提示する一連の前処理を施した後，クラスタ分析を行った．この結果，Adware, Backdoor，および Trojan の一部は他と明確に異なる特徴を有すること，および残りの Trojan と Worm についてはファミリー間の区別がつきにくいことが判明した．後者に関しては共通の特徴を有するファミリー群の存在が確認できた．また非負値行列因子分解によるクラスタ分析の結果を用いて，特定のマルウェアファミリーに共通する API コールの組み合わせをトピックとして自動抽出できること，およびトピックを利用して類似検体を検索可能であることを示す．

Analysis of massive amount of API call logs collected from automated dynamic malware analysis systems

Akinori Fujino†　　Tatsuya Mori†

†School of Fundamental Science and Engineering, Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555, JAPAN
{fujino,mori}@nsl.cs.waseda.ac.jp

Abstract This work aims to classify malware samples and extract intrinsic characteristics from them, using machine-learning approach. For this, we build a novel data preprocessing scheme for API call logs. We then employ clustering analysis to Win32 API call logs that were collected from automated dynamic malware analysis system. We found that malware samples labeled as Backdoor, Adware, and some Trojan ones were successfully classified while the rest of samples such as Worm and other Trojan oens were not. We also found that the latter samples exhibit common characteristics in terms of API calls; i.e., they are likely using similar high-level functions. We also found the "topics" extracted through the non-negative matrix factorization technique can be used to find new malware samples that exhibit similar characteristics.

1　はじめに

マルウェアの脅威を防ぐ対策として行われるマルウェア解析は静的解析と動的解析に大別される．静的解析はより深くマルウェアの挙動や手口を解析することを目的として行われるが，マルウェアは解析を回避するために様々な難読化が施されるため，静的解析のみでは対処できないケースが存在する．この問題を解決するひとつの手段は限りなく実際の感染者の環境と近い状況でマルウェアを動作させ，その結果を解析すること–すなわち動的解析を実施することである．

一方，マルウェアは日々驚くべき速度で大量生産されている．Kaspersky Lab の調査では 2012 年末時点において毎日 20 万種類のマルウェアが検出されていることが報告されている [1]．そのようなマルウェアの大量生産体制に抗するべく，動的解析を自動化する技術の進展がめざましい [2]．本研究で利用するデータセットを収集するために使われたオープンソースソフトウェアである Cuckoo Sandbox [3] はその一例であり，マルウェア解析における一連のデータ解析と記録をすべて自動化する．具体的には PE フォーマット形式ファイルの簡易な静的解析結果の出力，仮想マシン上での実行時の API コールの詳細な記録，プロセス情報，アクセスファイル，レジストリ，ファイルハッシュ値，ネットワーク通信の記録等を自動化している．

上記のような自動化された解析技術を大量に生産されるマルウェアに適用すると解析システムが出力するログデータも大量となる．したがって大規模なログデータから有効な対策につながる規則性を導くことが次の課題となった．本論文はそのような背景にもとづき，マルウェアの動的解析で得られる情報の中でも特に量が膨大となる API コールログのクラスタ分析を行う．API コールは情報量が膨大であるだけでなく，マルウェア解析上重要な指標であることが報告されている [4, 5, 6]．また，大量に収集される検体を自動的にクラスタ分類できれば，次のアクションを決定するための 1 次フィルターとして活用できると期待される．

本研究では約 2,600 のマルウェア検体に対して Cuckoo Sandbox を適用して収集したログを分析対象にする．ログから Win32 API の関数名と引数の情報を用いて特徴ベクトルを作成し，k-means 法および非負値行列因子分解 (NMF) を用いたクラスタ分析を行う．これらのクラスタリングアルゴリズムはそれぞれデータは必ず単一のクラスタに属するハードクラスタリング (k-means) と，データが複数のクラスタに属することを許容するソフトクラスタリング (NMF) を実現する手法である．

本研究の主要な貢献は下記のとおりである．

- 大量の Win32 API コールログを機械学習のフレームワークで分析するためのデータ前処理方法を詳細に提示した．
- マルウェア検体を分類する上で API の関数名だけでなく，引数値が重要であることを明らかにした．
- Adware，Backdoor，および Trojan の一部は他と明確に異なる特徴を有すること，および残りの Trojan と Worm についてはファミリー間の区別がつきにくいことが判明した．
- 互いに区別がつきにくいファミリー群が共通の特徴を有することを明らかにした．
- マルウェアファミリーに特有な API コールの組み合わせをトピックとして自動抽出可能なこと，およびトピックの応用例として類似検体の検索事例を示した．

本論文の構成は以下の通りである．はじめに 2 章ではデータセットの概要，およびデータの前処理としてスクリーニング，特徴ベクトル生成，特徴選択の具体的な方法を述べる．つづいで 3 では k-means 法と NMF の概要を簡潔に紹介する． 4 章でクラスタリング結果を述べた後， 5 章にて本論文のまとめと今後の展望を述べる．

2 分析データと前処理

本章では，分析に利用するデータの詳細と，クラスタ分析にかけるための前処理の手順を示す．

2.1 データの概要

本研究では MWS 2013 の研究用データセット [7] の一部として FFRI 社が提供している FFRI Dataset 2013 [8] を用いる．以下に FFRI Dataset の主な特徴を記す． FFRI Dataset は 2012 年 9 月から 2013 年 3 月にかけて様々なチャネルで収集されたマルウェア検体の内，PE 形式かつ実行可能な 2,644 検体を Cuckoo Sandbox を用いて動的解析したログである．ログファイルは JSON 形式であり，圧縮前のログファイルは合計 1.7 GB であり，テキストとしては約 4300 万行のデータである．Cuckoo Sandbox による 1 検体当りの実行時間は 90 秒であった．

Cuckoo Sandbox と VirusTotal [9] との連携機能により，各検体には最大で 66 種類のアンチウィルスソフトウェアによる検査結果がついている．本研究は後述する方法によって VirusTotal の検査結果を用いて各検体にマルウェア種別を示すラベルをつける．ラベル付きのマルウェアを分析することによって，API コールの情報がマルウェアの自動分類や識別に有効であるかを経験的なアプローチによって明らかにすることを狙いとする．

2.2 データの前処理

本節では API コールをクラスタ分析するために必要なデータの前処理方法を詳細に示す，はじめにラベル付きのマルウェアを各ラベル毎に偏りなくランダム抽出するデータスクリーニングの手順を説明する．次に選択されたマルウェアに対して API コールを「単語」に変換し，出現単語集合にもとづいて特徴ベクトルを作成する方法を示す，最後に単語の出現頻度にもとづく特徴選択方法を示す．

2.2.1 データスクリーニング

前述したように Cuckoo Sandbox のログは最大で 66 種のアンチウィルスソフトウェアの検出結果を含む．本研究ではその内最も検知率が高かった Kaspersky [10] の検知結果を利用してラベルを作成する．以下ではデータスクリーニングをした上でラベルを作成する手順を述べる．

文献 [11] の記載によると Kaspersky によるマルウェア命名則は下に示す通りである．

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

Prefix と Variant はオプションであり，それぞれ検体を検出したサブシステム，および検体の亜種を示す．Prefix が HEUR である検体はなんらかのヒューリスティックにより検知したケースであるため，一般にその素性は明確でない．Behaviour は検体の動作を示し，Viruses，Worms，Trojans，Malicious Tools，Adware，Riskware，Pornware などがある．Platform は OS 種別であり，Win32，Linux，multi などがある．Name は検体の公式なファミリー名である．

Kaspersky の検知結果では 1,051 種類のユニークなマルウェア種別を得た．すなわち各マルウェア種別に属する検体数は平均して 2 から 3 程度である．上記は亜種を区別した場合であるが，亜種を集約することでマルウェア種別を 277 種類にまで減らすことが出来る．種別を減らすことで，種別ごとのサンプル数を増やす効果が期待できるため，以下の分析では亜種を区別しない．なお，ここでは亜種は同様の API コールを持つと暗黙に仮定しているが，後に見るように同一種別であっても API コールの特徴に差異が生じるケースは存在する．

表 1 に Kaspersky のマルウェア検出結果の一部を示す．表より検出されたマルウェアの約 25% が HEUR:Trojan.Win32.Generic に分類されていることがわかる．前述したようにこの検体の実体は Prefix が HEUR であり，かつ Name が Generic であるため明確ではない．本研究の目的は素性が明確なマルウェアを対象にすることでクラスタ分析が有効に働くことを示すことであるので，以下ではこれらの検体を分析対象から除外する．

表 1: Kaspersky の検知結果 (Top 10).

マルウェア名	検知数
HEUR:Trojan.Win32.Generic	547
Worm.Win32.WBNA	275
Trojan.Win32.Jorik.Vobfus	110
Worm.Win32.Vobfus	84
Trojan-PSW.Win32.Tepfer	55
Trojan-Spy.Win32.Zbot	43
Trojan.Win32.SelfDel	43
Trojan.Win32.Agent	37
HEUR:AdWare.Win32.iBryte.heur	36
Worm.Win32.VBNA	36
合計	2021

表 2: スクリーニング後の分析対象マルウェア種別．括弧内数字はラベル ID．

(1) B.Azbreg, (2) B.Buteraft, (3) B.Shiz, (4) B.Simda, (5) T-Do.Agent, (6) T-Dr.Dorifel, (7) T-Dr.VB, (8) T-PSW.Tepfer, (9) T-Spy.Zbot, (10) T.Agent, (11) T.Diple, (12) T.Jorik.Vobfus, (13) T.Midhos, (14) T.SelfDel, (15) T.Skillis, (16) T.VB, (17) T.VBKrypt, (18) T.Vobfus, (19) T.Yakes, (20) DangerousObject.Multi.Generic, (21) W.AutoRun, (22) W.VBNA, (23) W.Vobfus, (24) W.WBNA, (25) A.Gamevance, (26) A.iBryte, (27) A.iBryte.heur

さらに各マルウェア種別毎のサンプル数を平滑化することで特に数が多いクラスのオブジェクトがクラスタリングアルゴリズムに与えるバイアスを排除する．また，各マルウェア種別の特徴を統計的に捉えるために必要な検体数の下限値を設定する．具体的には最低 10 個の検体が属するマルウェア種別を選別し，それぞれの種別毎に 10 個の検体を無作為抽出した．この結果，表 2 に示す 27 のマルウェア種別について，それぞれ 10 個の検体を得た．本研究では以降，27 のマルウェア種別をラベルとして用い，合計で 270 個のマルウェア検体を分析の対象する．なお紙面の節約のため，下記のような省略をした．Backdoor → B, Trojan → T, Downloader → Do, Dropper → D, Worm → W, Adware → A, Downloader → Do, Dropper → Dr. platform が Win32 の場合は省略．

2.2.2 特徴ベクトルの作成

つぎに API コールから特徴ベクトルを作成する手順を示す．図 1 は単一の API コールを記録した

表 3: 単語の作成レベルの定義と具体例.

レベル	定義	具体例
Level 0	api	"LdrGetDllHandle"
Level 1	api:n1:n2:...	"LdrGetDllHandle":"FileName":"ModuleHandle"
Level 2	api:n1:m(v1):n2:m(v2):...	"LdrGetDllHandle":"FileName":"C:\\WINDOWS\\system32\\rpcss.dll":"ModuleHandle":MASK
Level 3	api:n1:v1:n2:v2:...	"LdrGetDllHandle":"FileName":"C:\\WINDOWS\\system32\\rpcss.dll":"ModuleHandle":0x00000000

```
"category": "system",
"status": "FAILURE",
"return": "0xc0000135",
"timestamp": "2013-02-28 12:03:55,656",
"thread_id": "432",
"repeated": 1,
"api": "LdrGetDllHandle",
"arguments":
  [{"name": "FileName",
    "value": "C:\\WINDOWS\\system32\\rpcss.dll"},
   {"name": "ModuleHandle",
    "value": "0x00000000"}]
```

図 1: API コールログの例.

ログの抜粋である (紙面の都合上実際の JSON を変形した). 図中の api, arguments はそれぞれ API の関数名と引数である. arguments は更に引数名 name と値 value を含む. 本研究ではこれら 3 つの変数を用いて各 API コールを「単語」に変換する. なお, 他にも API コール が持つ特徴的な情報として戻り値やスレッド ID があるが, 分析が煩雑になるため今回は対象から外している. これらの効果に関する分析は残課題とする.

単語の変換方法を, 情報量のレベルによって表 3 に示すような 4 パターンに分類した. 表中の n, v はそれぞれ name, value の略である. また Level 2 の m() は値が 16 進数の数値であるときにマスクをかける関数であり, メモリアドレス等の差異を考慮しない効果を期待している. Level 0 は API 関数名のみで単語を生成する方式であるが, 予想されるようにこの方式は情報量が少ない.

各検体に含まれるすべての API コールから単語を抽出し, それらの単語集合を Bag-of-Words (BoW) モデルで表現することで特徴ベクトルを作成する. 具体的には 270 個の検体のログデータから抽出した単語集合を $\mathbf{W} = \{w_1, w_2, \ldots, w_m\}$ とし, i 番 目の検体の特徴ベクトルを $\mathbf{X}_i = \{\theta(w_1), \theta(w_2), \ldots, \theta(w_m)\}$ と定義する. ここに $\theta(w)$ は検体が単語 w を含む時に 1 , 含まない時に 0 となる関数である. すなわち単語の出現有無によって特徴ベクトルを構成する.

自然言語処理の分野では単なる単語の有無ではなく, TF-IDF 等を用いて単語に重み付けをする手法が用いられるが, TF および TF-IDF のいくつかのバリアントを試行した結果, 単語の有無で特徴ベクトルを構成した上で次節に示す特徴選択を行う方式が経験的に最も有効であったので, 本研究では単語有無による特徴ベクトルを採用した.

2.2.3 特徴選択

以上で示した手順にしたがってスクリーニング, 特徴ベクトル作成を行った後, クラスタ分析が有効に働くような特徴選択を行う. 基本的な考え方は分類に貢献しない単語を除外することであり, 以下の 2 つの方法を用いる. 第一の方法は大多数のラベルに共通して出現する単語は分類には寄与しないとみなし, 出現頻度が高い単語を除去する. これを高頻度フィルターと呼ぶことにする. 具体的には $0 < H \leq 1$ を満たす閾値 H を用い, ある単語が出現したマルウェアサンプル数の割合が全体の H 以上であったら除外する. この手法は DF-thresholding として知られるものであり, テキスト分類タスクにおける単語選択などで使われる [12]. 第二の方法は出現頻度が極端に低い単語を除去することであり, これを低頻度フィルターと呼ぶことにする. 具体的には閾値を $L = 1, 2, \ldots$ とし, ある単語が出現したマルウェアサンプル数が L 未満のとき, その単語を除外する.

図 2, 3 は特徴選択をした結果をサイズ $I \times J$ の行列 $\mathbf{X}$ で表現したものである. 行列の要素が 1 を黒, 0 を白でプロットしている. 行は単語を出現順にソートしたものであり, 単語生成レベルによってそのサイズ I は異なる. 列は個々のマルウェア検体を示し, ラベル順にソートされている. データスクリーニングの結果, $J = 270$ である.

図の基本的な見方として, 同一のラベルを持つ 10 個のマルウェア検体の集合 (薄い線の矩形で囲まれた部分) に属する単語が矩形内で水平方向に連続して出現しており, かつそれらの単語が他のラベルには出現しないときに分類精度が高くなる. つまり他のラベルには登場しない黒いマス目 (ただしバーコードのような細いマス目もある) のみを持つラベルは他とは明確に弁別されると期待できる. このように

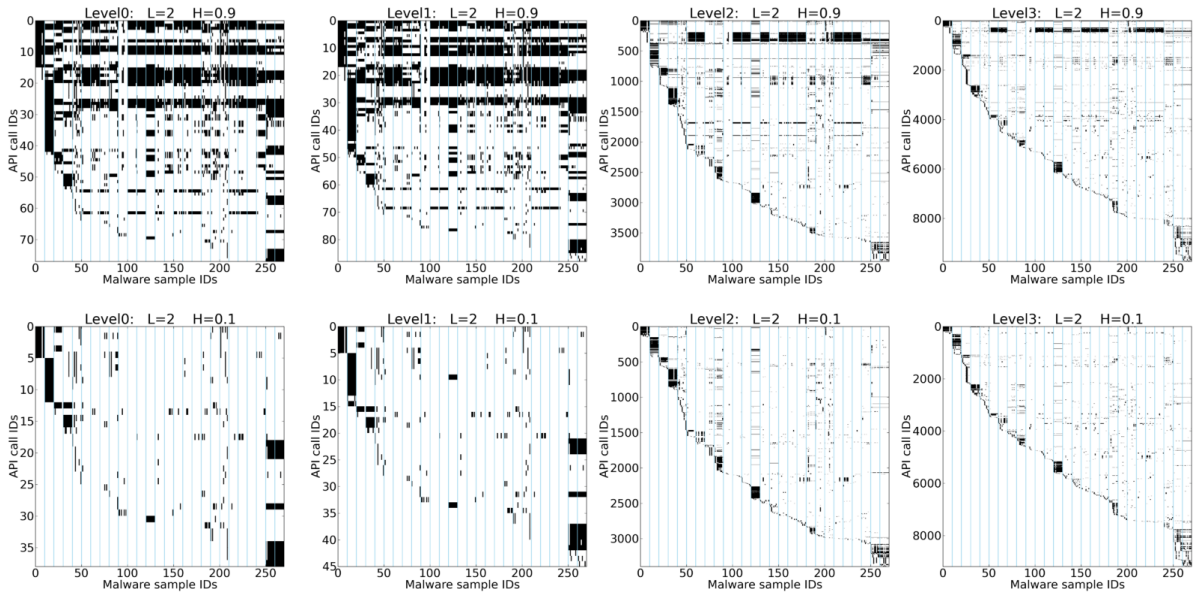


図 2: 単語生成レベルと高頻度フィルターによる特徴ベクトルの差異.

ラベル付きの特徴ベクトルを行列として表現することにより，データの特徴やクラスタ分析の成否を大雑把にとらえることができる．

図 2 は $L = 2$ と固定した時に，単語生成レベルと高頻度フィルターの閾値を $H = 0.9, 0.1$ と変化させたときの行列を示す．はじめにレベル 0,1 とレベル 2, 3 はそれぞれ類似していることがわかる．また，レベルが 1 から 2 にあがるに伴い，生成される単語数が飛躍的に拡大することがわかる．レベル 0,1 については部分的に分類がうまくいくと考えられる箇所があるものの，全般的に単語が複数のラベルにまたがるか，単語がほとんど存在しないラベルが出てきてしまい，分類はあまりうまくいかないことが定性的にわかる．すなわち，*API* **コールにおいては引数の値が重要な役割を果たす**ことが示された．一方，レベル 2, 3 はレベル 0,1 と比較して分類がうまくいきそうなことがみてとれる．高頻度フィルターの効果に関しては，閾値を $H = 0.9$ から $H = 0.1$ に変更することで複数ラベルに共通する単語を削除する効果がでるため，結果として分類がシャープになる効果が認められる．その他の H についても同様の方法で調査したが，紙面の都合で割愛し，以降では経験的に決めた $H = 0.1$ を採用する．

図 3 は $H = 0.1$ と固定した時に，単語生成レベル 2, 3 と低頻度フィルターの閾値を $L = 2, 5, 10$ と変化させたときの行列を示す．$L = 2$ の場合はよ

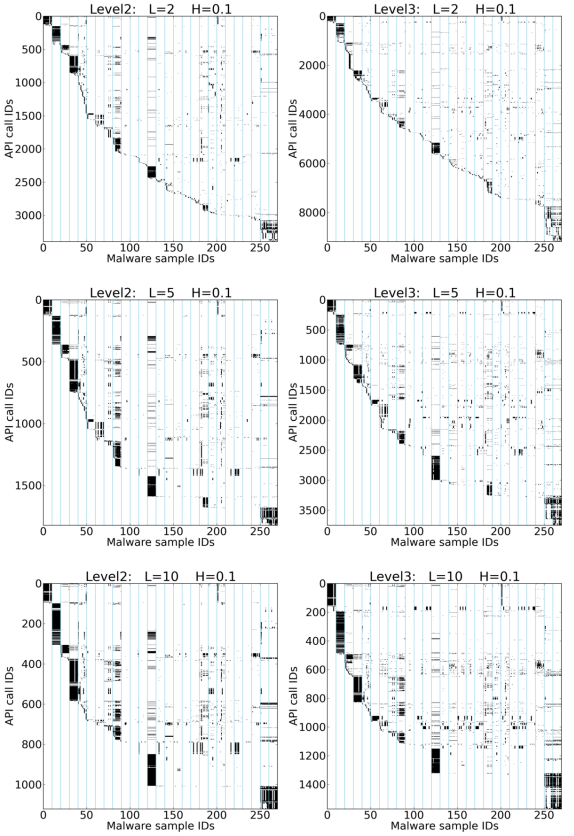


図 3: 単語生成レベルと低頻度フィルターによる特徴ベクトルの差異.

り多くの単語を拾うためラベルと無関係な点が多数あるが，L を増加させることにより，それらの「ゴミ」を除去する効果が出る．また今回のデータはラベル毎に 10 個の検体があるため，理想的な単語はある特定のラベルにのみ 10 個存在する．実際には同一ラベルには亜種が含まれているため，かならずしも 10 個すべてが同じ単語を持たないケースもあり，$L = 10$ とするとそのようなケースを見逃してしまうことになる．本研究では両者の中間的な値として $L = 5$ を採用する．単語レベルの差異はこの図からのみでは甲乙をつけがたいが，経験的に Level 3 の方が良い結果を与えたため，次章の分析では Level 3 を採用する．

3　クラスタリングアルゴリズム

本研究ではハードクラスタリング手法である k-means およびソフトクラスタリング手法である NMF を用いてクラスタ分析を行う．以下ではそれぞれのアルゴリズムの概要を簡潔に述べる

k-means　k-means は与えられたデータを距離の近さにもとづいて k 個のクラスタに割り当てるアルゴリズムである．アルゴリズムの動作は以下の通りである．はじめにランダムな重心を与え，各データとの距離 (例えばユークリッド距離やコサイン類似度など) を計算し，データと最も近いクラスタをそのデータの属するクラスタとする．その後，各クラスタの重心を更新し，新たな重心と各データの距離を計算し，属するクラスタを更新する．この操作を収束するまで繰り返す．

NMF　非負値行列因子分解 (NMF) は非負値からなる行列を分解するアルゴリズムであり，画像認識，音響信号処理，文書データの分類，ネットワーク監視データの分析等の応用に利用されている [13, 14]．前章で導入したように各マルウェア検体の特徴ベクトル集合が $I \times J$ の非負値行列 $\mathbf{X}$ で表現されているとする．NMF は $\mathbf{X}$ を $I \times K$ 非負値行列 $\mathbf{T}$ と $K \times J$ 非負値行列 $\mathbf{V}$ の積に分解するアルゴリズムである．K は基底の数であり，文書分類の場合はトピック数と解釈される．一般にはクラスタ数に相当するものであり，アルゴリズムの利用者がデータに合わせて適切に設定する量である．

NMF のアルゴリズムは $\mathbf{X}$ と $\mathbf{TV}$ の距離 $D(\mathbf{X}, \mathbf{TV})$ を最小化する．距離関数はいくつかの選択肢があるが，本研究では Kullback-Libler divergence を適用

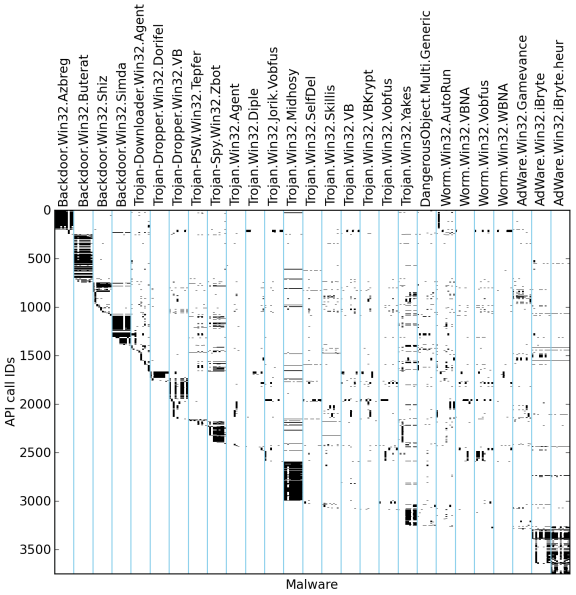


図 4: クラスタ分析対象の特徴ベクトル行列 $\mathbf{X}$

する．距離関数は下記の更新式を繰り返すことによって最小化される．ただし $\widehat{x_{ij}}$ は行列 $\mathbf{T}$ の i 番目の行と行列 $\mathbf{V}$ の j 番目の列の内積である．

$$t_{ik} \leftarrow t_{ik} \frac{\sum_j \frac{x_{ij}}{\widehat{x_{ij}}} v_{kj}}{\sum_j v_{kj}}, \quad v_{kj} \leftarrow v_{kj} \frac{\sum_i \frac{x_{ij}}{\widehat{x_{ij}}} t_{ik}}{\sum_i t_{ik}}. \quad (1)$$

4　クラスタ分析

本章ではクラスタ分析の結果を示す．クラスタリング結果の評価には様々な手法を適用できるが，本研究では個々のラベルの分類結果を直感的に解釈可能なクロス表を採用する．いずれのアルゴリズムもクラスタ数 K を設定する必要があるが，今回は経験的に決めた値 $K = 20$ を利用する．図 4 にクラスタ分析対象の特徴ベクトル行列を可視化した図を示す．前述したように，単語レベルを 3，DF 閾値を $L = 5$，$H = 0.1$ とした．

4.1　k-means によるクラスタ分析

図 4 の行列に対応する特徴ベクトル集合に k-means を適用した結果を図 5 に示す．k-means アルゴリズムにおける更新式の反復回数は最大 30 回とし，30 の異なる初期値によるクラスタリング結果の内，もっともクラスタ内距離総和が最小となる結果を採用した．

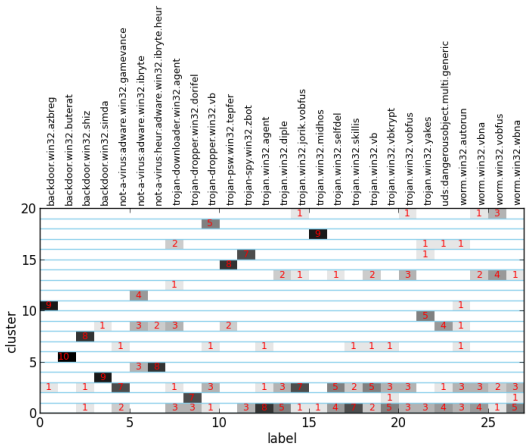
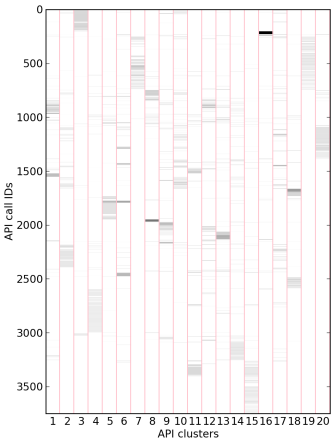


図 5: k-means のクロス表.

図より，Backdoor 4 種は全般的にきれいにクラスタに分離していることがみてとれる．また，Adware.iBryte 2 種は片方が 2 つのクラスタに分かれてしまっているが，概ね他のファミリーと独立したクラスタに分類された．Trojan については Tepfer, Zbot, Midhos がそれぞれ独立したクラスタに分類されている．その他，Worm 系全般を含む Vobfus, VB, VBNA, WBNA に関してはファミリー毎の明確なクラスタは存在せず，特定のクラスタ (ID=1,3,14) のいずれかに集中していることがみてとれる．一方，これらのファミリー名をもつマルウェア検体は亜種によっては相互に別のファミリー名で検知されることが報告されている [15]．したがってクラスタ分析の結果，単純に検知結果のファミリー名では判別できない内部動作の潜在的な類似性を抽出することができたと解釈できる．

4.2 NMF によるクラスタ分析

図 4 の行列に対応する特徴ベクトル集合に NMF を適用した結果を図 6 に示す．NMF アルゴリズムにおける更新式の反復回数は 50 回とした．特に行列 $\mathbf{V}$ は各々の検体がどの基底 (クラスタ) の線形和として表現されるかを示すものであり，クラスタ分析の結果に対応する．クラスタ分析の結果は k-means と概ね同様の結果を得た．すなわち，Backdoor 4 種と Adware.iBryte 2 種，Trojan の一部 (Zbot, Midhos, Yakes) はきれいにクラスタに分離していること，および Worm 系全般を含む Vobfus, VB, VBNA, WBNA に関してはファミリー毎の明確なクラスタは存在せず，特定のクラスタ (ID=6,18) のいずれかに集中していることから，相互に類似性を有してい



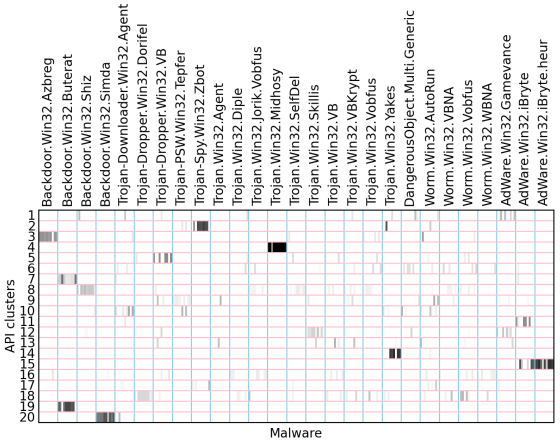


図 6: NMF による行列因子分解の例．行列 $\mathbf{T}$ (上) および行列 $\mathbf{V}$ (下).

る考えられる．API コールの類似性は同様の上位関数の存在が示唆される．そのような仮説を検証するアプローチとして行列 $\mathbf{T}$ を参照し，当該クラスタに対応する API コールを抽出して調査する方法が考えられる．これは今後の課題としたい．

4.3 NMF によるトピック抽出と応用

図 6 で示した行列 $\mathbf{T}$ は同時生起しやすい単語を「トピック」として抽出したものと解釈できる．トピックを構成する単語に対応する API コールはそれぞれのマルウェア種別の特徴を代表するものであるため，類似の検体を検索することに使えることが期待できる．表 4 はクラスタ番号 4 の `Trojan.Win32.Midhos` に関して行列要素の数値が高い単語上位 5 個をトピックとして抽出した結果である．以下では API 関数と

表 4: Trojan.Win32.Midhos のトピック単語抽出結果 (Top 5).


```
"RegOpenKeyExW":"Registry":"0x80000002":"SubKey":"Software\Microsoft\COM3":"Handle":"0x000000f4"
"LdrGetProcedureAddress":"ModuleHandle":"0x77cf0000":"FunctionName":"CallNextHookEx":"Ordinal":"0"
"NtCreateSection":"SectionHandle":"0x00000070":"DesiredAccess":"0x00000004":"ObjectAttributes:":"FileHandle":"0x0000006c"
"LdrLoadDll":"Flags":"522168":"FileName":"uxtheme.dll":"BaseAddress":"0x58730000"
"LdrLoadDll":"Flags":"522348":"FileName":"C:\WINDOWS\system32\uxtheme.dll":"BaseAddress":"0x58730000"
```


引数値が持つ意味に関する詳細な分析は行わず，この情報によって未知の検体を検知できることを示す簡易な実験を行った結果を示す．

表 4 に示す 5 つのトピック単語すべてを API コールとして利用した検体をスクリーニング前の全データから抽出したところ，15 検体が該当し，この内，10 検体はスクリーニング後に利用した検体であった．新たに発見された 5 検体の内訳はいずれも HEUR:Trojan.Win32.Generic であった (つまりスクリーニング後の検体数が閾値の 10 とたまたま一致していた)．これら 5 検体について他のアンチウィルスソフトウェアの検知結果を調べたところ，4 検体が Midhos ファミリーと検知されていた．さらに残りの 1 検体については Medfos ファミリーと検知されており，Medfos は Midhos ファミリーとして検知される場合がある [16]．以上のように NMF で抽出したトピック単語を利用して類似した特徴を持つ検体を検索可能であることを示した．

5 まとめ

本研究では自動化されたマルウェア動的解析システムから出力される大量の API コールを機械学習のフレームワークを用いて分析するための前処理方法を提示した．API コールを単語に変換し，BoW モデルで表現するアプローチは考えうる最も単純な方式であり，さらに高度なアプローチのベースラインを与えるものである．前処理したデータに対して k-means および NMF を用いてクラスタ分析を行った結果，API コールで分類可能なマルウェア種別とそうでない種別が存在すること，および類似していると考えられる複数のマルウェアファミリーに共通する潜在的特徴の存在が示された．また，NMF を利用することでマルウェアファミリーに特有な API コールの組み合わせをトピックとして自動抽出し，類似検体の検索に応用可能であることを示した．

本研究ではクラスタ分析の評価を有効にするためにラベル間のバランスをとるスクリーニングを行った．今回の分析から外れた検体に関しても原理的に同様のアプローチを用いた分析が可能であり，今後の課題としたい．また API コールの時空間データ構造を考慮したモデルの拡張，分類結果に基づくマルウェア検知技術の検討は今後の課題である．

謝辞 貴重なデータセットを研究コミュニティに貢献頂いた株式会社 FFRI の諸氏に感謝します．また，NMF の適用に関して有用なアドバイスを頂いた NTT 研究所の木村達明氏に感謝します．

参考文献

[1] Kaspersky Lab, “2012 by the numbers: Kaspersky Lab now detects 200,000 new malicious programs every day.” http://www.kaspersky.com/about/news/virus/2012/2012_by_the_numbers_Kaspersky_Lab_now_detects_200000_new_malicious_programs_every_day.

[2] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, “A survey on automated dynamic malware-analysis techniques and tools,” *ACM Comput. Surv.*, vol. 44, pp. 6:1–6:42, Mar. 2008.

[3] “Cuckoo Sandbox.” http://www.cuckoosandbox.org.

[4] C. Willems, T. Holz, and F. Freiling, “Toward automated dynamic malware analysis using cwsandbox,” *Security & Privacy, IEEE*, vol. 5, no. 2, pp. 32–39, 2007.

[5] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Krgel, and E. Kirda, “Scalable, behavior-based malware clustering.,” in *NDSS*, The Internet Society, 2009.

[6] M. Alazab, S. Venkataraman, and P. Watters, “Towards Understanding Malware Behaviour by the Extraction of API Calls,” in *Cybercrime and Trustworthy Computing Workshop (CTC), 2010 Second*, pp. 52–59, 2010.

[7] 神薗雅紀他, “マルウェア対策のための研究用データセット ～MWS Datasets 2013～.” MWS 2013 http://www.iwsec.org/mws/2013/, Oct 2013.

[8] “FFRI Dataset 2013.” http://www.iwsec.org/mws/2013/files/FFRI_Dataset_2013.pdf.

[9] “VirusTotal.” http://www.virustotal.com.

[10] “Kaspersky.” http://www.kaspersky.com.

[11] SECURELIST, “Rules for naming detected objects.” http://www.securelist.com/en/threats/detect?chapter=136.

[12] Y. Yang and J. O. Pedersen, “A comparative study on feature selection in text categorization,” in *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML ’97, pp. 412–420, 1997.

[13] 澤田宏, “非負値行列因子分解 NMF の基礎とデータ/信号解析への応用,” **電子情報通信学会誌**, vol. 95, pp. 829–833, sep 2012.

[14] 木村, 竹下, 豊野, 横田, 西松, 森, “Syslog+SNS 分析によるネットワーク故障検知・原因分析技術,” *NTT* **技術ジャーナル**, vol. 25, pp. 20–24, Jul 2013.

[15] Microsoft Malware Protection Center, “Win32/Vobfus.F.” http://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Worm:Win32/Vobfus.F#tab=2.

[16] Microsoft Malware Protection Center, “Win32/Medfos.” http://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Win32/Medfos.