情報処理学会論文誌 Vol. 50 No. 4 1321–1333 (Apr. 2009)

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

堀合啓一[†1,†2] 今泉隆文[†1] 田中英彦[†2]

マルウェアの動的な挙動を多次元のベクトルとして数値化し，ベクトル間の距離からマルウェアの亜種を判定する手法について提案する．マルウェアの実行にともなって顕在化したプロセスの起動やファイルの変化および発生するトラフィックなどの出現頻度を基に 2 値のカテゴリ・データとして数値化し，距離の算出は，高速に演算が可能なハミング距離を用いる．提案手法の有用性を実験によって検証するとともにマルウェア自動解析システムへ実装した．

Proposal and Implementation of Automated Malware Variant Classification Based on Dynamic Behavior

Keiichi Horiai,[†1,†2] Takafumi Imaizumi[†1] and Hidehiko Tanaka[†2]

This paper describes a method for malware variant classification using the distance between multi-dimensional vectors which represent the dynamic behavior of the malware. The dynamic behavior, involving state changes of processes, files, and network traffic, is transformed into binary category data. The measurement of the distance is based on Hamming distance, which requires less processing power than measurements proposed in previous works. The proposed method is proved its efficiency by our experiments and we applied the method to the automated Malware analyzing system.

†1 防衛省技術研究本部電子装備研究所
Electronic Systems Research Center, Technical Research & Development Institute, Ministry of Defense
†2 情報セキュリティ大学院大学
INSTITUTE of INFORMATION SECURITY

1. はじめに

近年のマルウェアは，開発ツールの流通や難読化・暗号化などの一般化によって，非常に多くの種類が出現し，パターンに頼る検出だけでは困難となりつつあるといわれている．マルウェアによる被害を局限するためには，できるだけ早期に検知して対策を行うことが必要である．このため，筆者らは，定点観測の手法でマルウェアを収集し，捕獲したマルウェアの動的な挙動を自動的に解析するシステムを構築しているが[29]，捕獲したマルウェアの約 10〜20%程度は，市販のウィルス対策（以下 AV: Anti Virus）ソフトウェアではマルウェアとして検出できない．その後，しばらくして AV ソフトウェアのパターンが更新されると，ある種類の亜種として検出されることが多い．捕獲した時点では，市販の AV ソフトウェアでマルウェアとして認識できない場合でも，マルウェアを実行して得られる，プロセスの起動，レジストリの改ざん，通信パケットなどの情報を，既知のマルウェアの挙動と比較し，その類似性からマルウェアの名称を推定できれば，さらに詳細な解析を行う場合や，対策を検討する際に，既知の情報の活用が容易となり，これらの作業を効率化できる可能性がある．

本研究は，マルウェアの動的な挙動を解析して得られる情報を利用し，捕獲した段階では市販の AV 製品ではマルウェアとして検出できない場合でも，既知のマルウェアとの類似性を自動的に算出し，その名称を推定する手法の提案である．以下，次章では関連研究の状況，3 章で提案手法を説明し，4 章では実験結果および考察を，5 章でマルウェア自動解析システムへの実装について述べて，6 章でまとめる．

2. 関連研究

マルウェアの自動分類手法として，文献 1)–6)，27) などが知られている．これらは静的解析と動的解析の 2 種類に大別できる．静的解析の一例である文献 1) は，Windows のバイナリ・ファイルを対象とし，自己組織化マップ（SOM: Self Organizing Map）によって感染の有無を視覚的に確認する方式である．文献 2) では，Windows のバイナリ・ファイルのバイト列を n-gram で表現し，マルウェアと正常なプログラムの分類を行った実験についての報告である．また，文献 3) は，マルウェアのバイナリ・ファイルを対象として，正規化圧縮距離（NCD: Normalized Compression Distance）[7] で相互の距離を算出して分類する手法を提案している．

一方，文献 4) は，動的解析の例であり，マルウェアを実行した際に記録した API CALL

のシーケンスを解析の対象とし，このシーケンスが一致する行数の割合，連続一致最大行数，関数の一覧などを比較して亜種の判定を試みている．文献 5) はマルウェアを実行・記録したイベント・シーケンスを解析の対象とし，類似性の判定には編集距離を利用している．文献 6) では，マルウェアを実行して記録した動的な挙動を解析の対象とし，類似性の判定には NCD や単連結階層クラスタリングなどを利用して，分類すべきマルウェアの種類数とメモリの所要量，処理時間の関係などを分析し，既存のマルウェアの分類に依存しない，動的挙動に基づく新たなクラスタリング手法を提案している．

文献 27) は，マルウェアの挙動の動的解析を行う CWSandbox[28] を利用し，API CALL の文字列を処理の対象として，Support Vector Machine（SVM）の手法で分類を行う提案である．一般に，マルウェアの名称は科名（Family name）と亜種名（Variant name）を組み合わせた形式となっているが，この提案は，市販の AV 製品が付与した科名を基準として分類するものであり，マルウェアの科名 14 種類について実験した結果を報告している．この提案では，分類の対象としたマルウェアと，それ以外の種類のマルウェアを明確に区別できる利点があるものの，区別するための学習のフェーズが，分類対象のそれぞれの科名ごとに必要とされている．

本提案は，文献 29) の解析環境を使って取得した挙動の文字列の情報を数値化し，編集距離の算出との比較では高速な処理が可能とされている NCD よりもさらに計算コストが低い，ハミング距離（HMD: Hamming Distance）を利用し，市販の AV 製品の名称に基づいたマルウェアの分類と，その名称の推定を行う手法の提案である．本提案と同様に，市販の AV 製品に対応したマルウェア名を推定する手法として，前述の文献 27) が知られている．ただし，文献 27) では，推定の対象が科名までとなっている点と，分類対象のマルウェアの種類に応じた個別の学習フェーズが必要となる点で本提案とは異なっている．本提案では，科名 + 亜種名の推定が可能であり，また，分類対象の種類に応じた個別の学習フェーズも不要である．マルウェアの名称を推定することによって，対策を検討する場合やコードレベルのようなさらに詳細な分析を行う際に，その名称を手がかりとして既存の知見の活用が容易となり，市販の AV 製品では検知できないマルウェアへの対応を効率化することが可能となる．

3. 提 案 手 法

文献 29) では，定点観測によってマルウェアを捕獲し，捕獲したマルウェアを仮想マシン上の WindowsXP 内で実行して，その挙動を自動的に解析するシステムを提案している．本研究は，文献 29) の解析結果を利用し，解析の時点で AV 製品ではマルウェアとして検出されない対象に対し，挙動の類似性を自動的に算出して，マルウェアの名称を自動的に推定する手法の提案である．最初に，本提案手法の前提となるマルウェアの動的挙動の解析環境について，その概要を次に述べる．

3.1 挙動の解析環境と BDB（Behavior Data Base）

動的挙動解析のネットワーク環境は，Linux のカーネル・パケット・フィルタに使用されている iptables の機能を利用して構築している．この模擬ネットワークには，マルウェアを実行する仮想マシン上の感染 PC（OS: WindowsXP）のほか，仮想マシンのホスト OS 上に設定した模擬 DNS，IRC，SMTP，SMB，HTTP の各サーバ群が接続されている．ネットワークを模擬環境で構成する利点は，解析を安全に実行できる点に加え，挙動解析の再現性を確保しやすい点にある．仮に感染拡大の防止など，外部への影響を緩和する対策を行ったうえで，インターネットへ接続した状態で解析を行う場合では，マルウェアを実行する時期・時間帯によって通信の相手先の状態の影響を受ける可能性があり，マルウェアの種類とは必ずしも直結しない要素で，観測できる挙動が変化する可能性がある．インターネットへ接続しないことによって，ボットの Herder からの指令などを観測できないという欠点もあるが，マルウェアを実行した直後の数分間の挙動を自動的に解析し，マルウェアの種類を特定するための環境としては，模擬環境の方が適していると考えられる．また，Windows 内の挙動については，文献 4)，5)，27) のように API CALL の情報を解析するのではなく，マルウェアを実行する前の状態と実行後の状態を記録したログを比較し，それらの差分をマルウェアの挙動を示す情報として抽出している．これによって，デバッガの存在を検出して，その挙動を変化させるマルウェアへの対策としている．また，仮想マシンを検出するマルウェアの存在も知られているが，これについては今後の課題としている．

以上のような環境で取得できる情報は，AV 製品ベンダなどが公開している，マルウェアの特徴（レジストリの改ざん箇所と内容，マルウェアが作成・削除・改ざんするファイルの名称，起動または停止されるプロセスやサービスの名称，ルートキットの埋め込み，発生する通信のポート番号，通信のあて先，IRC サーバへログインする際のユーザ名やパスワードなど）とほぼ同等である．

以上の動的挙動解析の結果から，**図 1** にその一例を示す各項目を文字列の情報として抽出し，これを BDB（Behavior Data Base）へ蓄積している．

BDB には，各マルウェアに対する複数の AV 製品によるスキャン結果を含むが，解析対象のファイルからマルウェアが検出されなかった場合には，Unknown としている．分類に

[HASH] マルウェア・ファイルのハッシュ値.
04999957e3c78e03737cd55a61a7f3ca
[REGISTRY] レジストリファイルの変化
HKLM¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Run
c:¥windows¥system32¥logon.exe
[MD5SUM] Windowsシステム関連ファイルの変化
Created C:¥WINDOWS¥SYSTEM32¥LOGON.EXE
[PROCESS] 検知したプロセスの起動・停止
winlogon, services, logon
[HOSTS] Windows HOSTSファイルの変化.
No change Found.
[ROOT KIT] ルートキットの検出結果
Not Detected
[SERVICES] 検知したサービスの起動・停止
No change Found.
[TRAFFIC] 観測したトラフィックのポート番号
PORT(2), domain(2), 8998(16)
[MALWARE CLASSFICATION] ウィルススキャンの結果
C:Trojan.Lineage-80, T:Unknown, S:W32.IRCBot,
K:Trojan-PSW.Win32.Nilage.zh

図 1　BDB（Behavior Data Base）の一例
Fig. 1　An example of a BDB (Behavior Data Base).

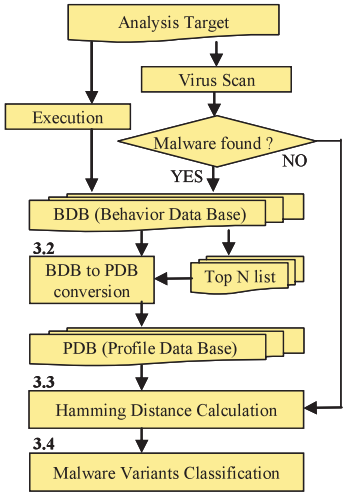

図 2　マルウェア分類処理の全体構成図
Fig. 2　An overall block diagram of malware variant classification.

あたり，同種類のマルウェアの挙動は類似していることを仮定しているが，実際の状況は実験結果とともに次章で示す．

3.2　分類の手順

BDB の各項目を，本論文では BDB の「要素」と呼ぶ．BDB の各要素は，不定長の文字列で表現された複数個のレコードで構成されている．本論文は，この BDB の各要素を利用して，マルウェアを自動的に分類する手法の提案である．本提案によるマルウェアの分類処理の全体ブロック構成を**図 2** に示す．また図 2 におけるトップ N リストの一例を **R**，**M**，**T** について**表 1** に示す．ここでトップ N リストとは，マルウェアを実行して観測した挙動の各要素に出現する文字列の出現頻度の高い順から N 番目までのリストを意味している．

分類の各ステップは次のとおりである．

［STEP-1］マルウェアを実行し，ファイルの改ざん，プロセスやサービスの起動，トラフィックの発生など，マルウェアの実行にともなって顕在化する動的挙動を記録したデータベース（BDB）を生成する．

［STEP-2］マルウェアを市販の AV 製品でスキャンし，判明したマルウェアの名称を BDB へ加える（検出されない場合は Unknown とする）．

表 1　**R**，**M**，**T** に関する Top N（$N = 5$）リストの一例
Table 1　An example of Top N list for **R**, **M** and **T** for $N = 5$.

N	R:REGISTRY(Value)	M: MD5SUM	T: PORT
1	%win%system32\browseui.dll	%win%system32\vcmgcd32.dll	80
2	%win%system32\shdocvw.dll	%win%system32\spoolsvc.exe	65520
3	%win%system32\explorer.exe	%win%system32\lssas.exe	25
4	%win%system32\urlmon.dll	%win%system32\algs.exe	1863
5	%win%explorer.exe	%win%system32\winiogon.exe	8585

［STEP-3］STEP-1 で作成した BDB から要素ごとの出現頻度リスト（図 2 における Top N リスト）を生成する（このステップは，BDB の初期生成時と大幅な更新時のみ実行）．

［STEP-4］Top N リストを使って，BDB の各要素をカテゴリ・データへ変換し，これらを結合して PDB（PDB: Profile Data Base）へ追加する．BDB は複数レコード，不定長の文字列の情報であるが，これを固定長のカテゴリ・データへ変換し，複数のレコードを 1 レコードにまとめて PDB を生成する．

［STEP-5］STEP-4 で生成した PDB を利用し，分類対象の検体と，検体自身を除く PDB

```
# hash_value_of_Malware_binary   categorized_data
049c244101170faa7c743e1ac5494ede 0001000000000600
04af7239845601e9d785a7824b6ca34e 0000001200000600
04cb88703c78a3ffa6f678a9a77c66c7 0000000008002740
04ecc1d94e27cd9f8822dc69b8e78d8a 1400000000000700
0502329d4e1c5bf4fe370e3a9e246454 0000000000002600
052b23dd1320692f6508e7c24b519d0e 8000008000000e00
05534778b7e1652237be1ca9497bd230 8000000100000e00
```

図 3　PDB（Profile Data Base）の一例
Fig. 3　An example of a PDB (Profile Data Base).

内のすべてのマルウェア個体間のハミング距離を算出する．

［STEP-6］距離が最短となるマルウェアを求めてこれを検体の種類の候補として出力する．ここで，距離が最短のマルウェアが複数種類存在する場合には，種類ごとの個体数が多い種類を候補として出力する．

以下，提案手法の中心である BDB から PDB への変換および距離の算出とこれに基づくマルウェアの分類について述べる．

3.3　BDB から PDB への変換

マルウェアの挙動の類似性をハミング距離で測定するには，不定長の文字列で表現された BDB から数値で表現した PDB へ変換が必要となる．このため，図 1 で示した BDB 要素の {**R**, **M**, **P**, **S**, **K**} については，BDB 内に記録されたファイル名やプロセス名などの文字列を，要素 T については，観測したパケットのポート番号について，それぞれ出現頻度の高いトップ N のリストを参照して，カテゴリ・データへ変換する方式とした．すなわち，BDB 内の各「要素」に記録された文字列などに注目し，出現頻度の多い文字列のトップ N を $\{0\,|\,1\}$ の 2 値へ対応させてカテゴリ・データへ変換する．たとえば，レジストリの変化については，$R = \{r_1, r_2, r_3, \cdots r_n, r_{n+1}\}$ ただし，$r_i = \{0\,|\,1\}$ とする方式である．ここで r_{n+1} は，文字列がトップ N 番目以内に入らなかった場合に 1 をセットする「その他」のカテゴリである．

この BDB から PDB の変換は，BDB の各要素をパラメータとして関数 $\boldsymbol{F}$ を使って PDB へ変換する過程と考えることができる．この方法では，関数の選び方すなわち，利用する要素の種類，その組合せ，およびトップ N の値（= カテゴリの種類数）などの影響で PDB が変化し分類の精度が変化するが，これについては次章で述べる実験で検証した．本提案における PDB の一例を**図 3** に示す（カテゴリ・データは 16 進数表記）．図 3 の各行を PDB の構成要素と呼ぶ．ここで，PDB の構成要素数を |PDB| と表現すると，|PDB| = 分類対象のマルウェアの個体数である．

3.4　距離の算出

本提案では，マルウェアの挙動の類似性を判定する手段として，ハミング距離を利用する．ハミング距離（HMD: Hamming Distance）は，n 次元のベクトル $X = (x1, x2, \ldots, xn)$, $Y = (y1, y2, \ldots, yn)$ において要素が等しくない個数であり次式で表現される．

$$HMD(X,Y) = \sum_i^n \delta(Xi, Yi) \tag{1}$$

ただし，$\delta(X,Y)$ は $X = Y$ なら 0，そうでなければ 1 をとる関数である．各要素の値は 2 値の場合が多いが，多値の場合でもかまわない．特に X，Y が 2 値の場合には，排他的論理和（XOR）演算の結果から，ビットが 1 となっている数をカウントすることで距離の算出が可能であることから，少ない計算量で結果が得られる点に特徴がある．提案のハミング距離を利用した分類の精度を比較する意味で，文献 3)，6) が利用している NCD についても検証を行った．NCD は圧縮度に基づいた，汎用な類似度測定の算出手法であり，次の式で定義される．

$$\begin{aligned} NCD(X,Y) &= \frac{\max\{(C(XY) - C(X), C(YX) - C(Y)\}}{\max\{C(X), C(Y)\}} \\ &= \frac{C(YX) - \min\{C(X), C(Y)\}}{\max\{C(X), C(Y)\}} \end{aligned} \tag{2}$$

ただし，$C(X)$，$C(Y)$ は，それぞれ情報 X，Y の圧縮サイズであり，$C(XY)$ は，X と Y を結合した情報の圧縮サイズを表す．圧縮 C の演算には，zlib，bzlib など既存の圧縮ライブラリを利用することができるが，式 (1) との比較では，汎用に類似度の測定ができる利点があるものの，少なくとも圧縮の演算が 3 回必要である．この中の 1 回は，情報を結合したファイルが対象となるため，処理対象のファイルのサイズ換算では，4 回相当となる．したがって，仮に同サイズの情報のハミング距離の計算と圧縮演算の所要時間が同等とした場合でも，HMD の計算量は NCD と比較して 4 分の 1 以下となる．

3.5　未知のマルウェアの分類

市販の AV 製品では，マルウェアとして認識されなかったマルウェアを分類するフェーズでは，分類対象のカテゴリ・データと，蓄積した PDB 内のすべてのカテゴリ・データ間の距離を算出し，距離が最短となるカテゴリ・データを持ったマルウェアとして分類する．ただし，距離が最短のマルウェアが，複数種類存在する場合には，データベース内のそれぞれの種類ごとに，ハッシュ値が異なるマルウェアの検体の数をカウントし，その数をスコアとして，スコアが最大となる種類として分類する．ただし，本提案の方式では，既存のマル

ウェアとはまったく異なる新種についても，既存のマルウェアの挙動との類似性をもとに種類を判定することとなる．したがって，新種の挙動と最も類似している既存のマルウェアとして誤判定する可能性があることに注意が必要である．

4. 実験結果と考察

本章では，実験に利用したデータ，実験の手順および実験結果の説明とその考察を行う．分類の精度を評価する指標として，不正侵入検知システムの評価などに利用されている ROC（Receiver Operating Characteristic）分析[8] を用いた．ROC 分析における AUC（Area Under Curve）値を指標として，良好な分類結果を得るための要素ごとのトップ N の値や要素の組合せを検討した．

4.1 実験データ

BDB から PDB へ変換したハッシュ値を異にするマルウェアの個体数は 3,665 個である．これらの検体は，インターネットへ接続したハニーポット Nepenthes[22] で捕獲したものであるが，その大半はボット関連のマルウェアである．これらのファイルを市販の 4 種類のウィルス対策製品でスキャンして判明したマルウェアの種類数を**表 2** に示す．

マルウェアの名称は，科名（Family name）と亜種名（Variant name）の組合せとなっているが，この表から，同じ検体を対象としても，分類の種類数が製品によって大きく異なっていることが分かる．文献 4)，6) においても，あるマルウェアの個体が製品によって別の種類として分類されている例を指摘している．このような，製品による名称の一貫性の欠如の影響を低減させるために，分類に適したトップ N の値や，最適な要素の組合せを得るための ROC 分析に使うデータは，次の方法で選別した．

［STEP-1］4 種類の AV 製品でマルウェアをスキャンし，4 種類の名称を結合したラベルとそれぞれのマルウェアのハッシュ値に対応した名称のリストを生成．

［STEP-2］4 種類の名称を結合したラベルの出現頻度（= ラベルに対応したマルウェアの個体数）を算出し，ラベルに対応した個体数が多い（= 4 社の分類が一致している可能性が高い）順のリストを生成．

［STEP-3］同ラベルの個体数の最低基準（10 個）を設定し，同基準以上に含まれる個体を ROC 分析の対象として利用する．

この方法で，自動解析によって何らかの挙動の変化を観測できた 3,665 個の中から 1,503 個を抽出し，第 1 段階の実験データとした．

以下本論文では，表 2 に示したように，実験対象の検体に対し，4 種類の AV 製品の中では最も多くの種類に分類した，ウィルスバスター 2007[9] のスキャン結果を，マルウェアの名前として利用した．この製品のスキャン結果による，実験対象の種類別検体数の上位 5 を**表 3** に示す．

表 2　実験対象のマルウェアの AV 製品ごとの種類
Table 2　Number of families & variants classified by four AV products.

	TrendMicro[9] VirusBuster2007	Kaspersky[10] Internet Security6.0	Symantec[11] Internet Security 2007	ClamAV[12] 0.91
Variants	1,095	400	65	765
Families	95	63	35	74

表 3　種類別検体数
Table 3　Number of specimens for family & variant names.

No.	Family & variant names	samples	Family names	samples
1	PE_BOBAX.AH	550	PE_BOBAX	761
2	PE_VIRUT.A	424	PE_VIRUT	709
3	PE_SALITY.AS	370	WORM_RBOT	516
4	PE_VIRUT.B	270	PE_SALITY	388
5	PE_BOBAX.AK	204	WORM_SDBOT	176
	Other	1,847	Other	1,115

表 4　BDB の要素ごとで挙動を観測できた割合
Table 4　The ratio of observed changes of the dynamic behavior across the BDB for each element.

S	H	K	R	P	M	T
3%	17%	19%	77%	96%	67%	82%

次に，マルウェアを実行した前後で，各要素において何らかの挙動の変化が観測された割合を**表 4** に示す．

この表から，**P**，**R**，**M**，**T** の各要素については，挙動の変化が観測された割合が多いことから，これらの項目が分類の有力なキーとして利用できる可能性を示している．以下，これらの要素を挙動の主要素と呼ぶ．また，主要素以外の **S**，**H**，**K** については，マルウェアの実行前後で挙動の変化を観測できた割合が少ないことから，これらの要素単独の情報だけで，マルウェアの分類を行うことは困難と考えられる．ただし，主要素と組み合わせることによって，分類の精度を向上させることができる可能性があるので，これらについてはそれぞれの要素についての変化の有無を $\{0|1\}$ の 1 ビットで表現した．以下，これらの要素を補助要素と呼ぶ．

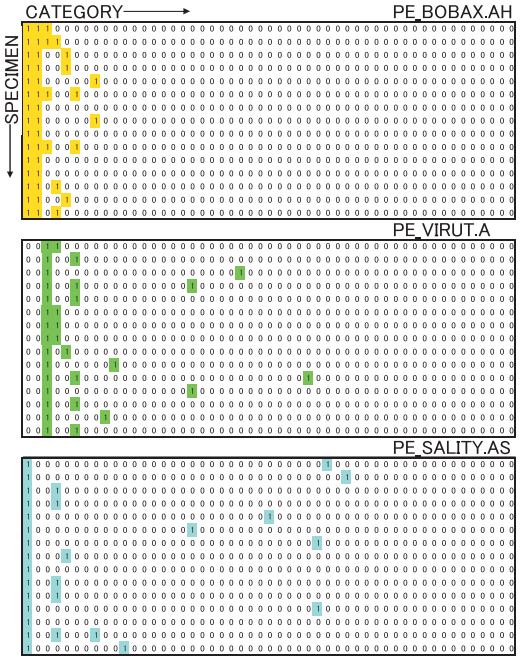

図 4　マルウェアの挙動の変動の一例
Fig. 4　An example of behavioral variations of malware.

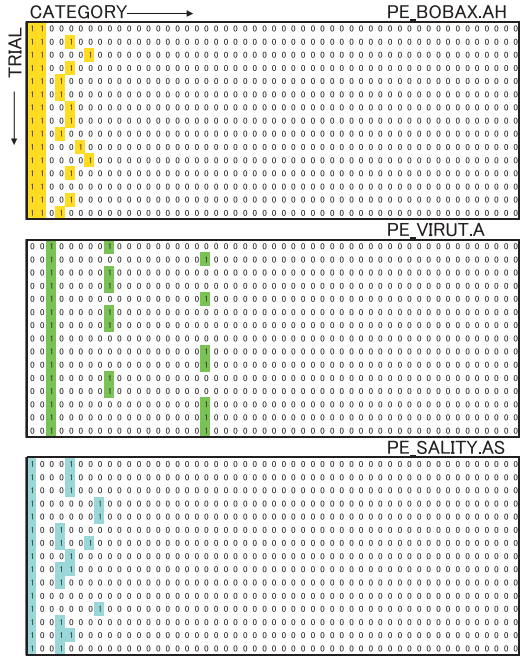

図 5　試行の繰返しによる挙動の変動の一例
Fig. 5　An example of behavioral variations during repeated malware execution.

4.2　マルウェアの挙動

本提案の手法は，同種類として分類されるマルウェアの挙動が類似していることを仮定している．ここで，実験データの中で同種類の検体数が多かった 3 種類のマルウェア（PE_BOBAX.AH，PE_VIRUT.A，PE_SALITY.AS）について，ファイルのハッシュ値が異なるそれぞれ 15 個体を例とし，解析対象のマルウェアを実行することによって発生した通信のポート番号をカテゴリ・データとして視覚化した例を**図 4** に示す．この図における横軸は，通信ポートのトップ 50 をカテゴリ・データとしたものであり，観測された通信ポートに対応したカテゴリ部分が着色されている．縦方向の各行は，3 種類のマルウェアそれぞれについて，ハッシュ値が異なる 15 個体を示す．この図から，同種類として分類されているマルウェアの挙動は一定ではないものの，利用ポートの状況は，種類ごとに特徴が認められる．また，実験の結果，カテゴリ・データ化したレジストリの変化やプロセスの状況についても，同様な傾向が確認され，「同種類として分類されるマルウェアの挙動は類似している」という仮定は，おおむね実態に即しているといえる．

次に，マルウェアの挙動をもとにして分類するためには，観測できる挙動の再現性も重要である．近年のマルウェアの中には，解析を妨害する意図で自身の挙動を変化させる種類の存在が知られている．この状況を確認した結果を**図 5** に示す．この図は，3 種類の個体各 1 個を，同じ環境下で各 15 回繰り返し実行した際に観測された通信ポートの状況を，図 4 と同様な手法でプロットしたものである（ただし，縦軸方向は，試行番号）．この図から，試行ごとに挙動の揺らぎが認められるものの，それぞれの種類に応じ，図 4 で示した挙動（ポート数とその分布）と類似した特徴が見られることが分かる．以上の実験で，マルウェア実行時の挙動をカテゴリ・データ化したパターンから，マルウェアの種類ごとの特徴の存在を定性的に確認できる．以下，挙動の類似性をハミング距離の算出で定量化し，分類性能の良否の評価を行う．

	A_1	A_2	A_3	B_1	B_2	B_3	C_1	C_2	C_3
A_1	–	5	4	11	11	12	14	13	13
A_2	5	–	5	10	6	11	13	12	12
A_3	4	5	–	11	11	12	14	13	13
B_1	11	10	11	–	4	5	9	8	6
B_2	11	6	11	4	–	5	9	8	6
B_3	12	11	12	5	5	–	8	7	5
C_1	14	13	14	9	9	8	–	1	3
C_2	13	12	13	8	8	7	1	–	4
C_3	13	12	13	6	6	5	3	4	–

図 6　距離マトリクスの一例
Fig. 6　An example of a distance matrix.

4.3　分類精度の評価方法

ハミング距離によるマルウェア分類の精度を，定量的に評価するため，ROC 分析における AUC 値を利用した．このため，分類対象のマルウェアの PDB から，相互のハミング距離を算出して距離マトリクスを生成する．一例として，3 種類のマルウェアをそれぞれ 3 個体，合計 9 個の距離マトリクスの例を**図 6** に示す（A，B，C がそれぞれ PE_VIRUT.A，PE_SALITY.AS，PE_BOBAX.AH に対応）．

この例では，距離 5 が分類の最適な閾値であるが，異種類を同種と誤判定する組合せを 1 件（C3 と B3 を同種と誤判定）含んでいる．この距離マトリクスから，距離の遠近とマルウェアの種類の関係を，グラフ化ツールの Graphviz [13] を利用して視覚的に表示した例を**図 7** に示す．この図は，HMD による計算結果を Graphviz の入力となる DOT 言語[25] 形式へ変換するスクリプトを作成して処理した結果の一例である．この図では，種類が同じものどうしのハミング距離は小さく（距離 $\leqq 5$），種類の異なるマルウェアどうしの距離は大きい（距離 > 5）ことが視覚的に示されている．ただし赤色の線で示した箇所は，異種類間の距離にもかかわらず距離 $= 5$ となっていて，異種類を同種類と誤判定するケースである．このようにグラフ化によって，距離の遠近を基準として，マルウェアの種類を判定できることが，視覚的に確認できる．

次に，分類性能の定量的な評価に利用した ROC 分析について述べる．ROC 分析は，同種を同種と判定（TP: True Positive）した割合（TPR: TP Rate）と異種を同種と判定（FP: False Positive）した割合（FPR: FP Rate）について，閾値をパラメータとして曲線を描画し，この曲線の下側の面積を AUC 値として，この面積の大小で分類の性能を評価するものである．実験で得た ROC 曲線の一例を**図 8** に示す．

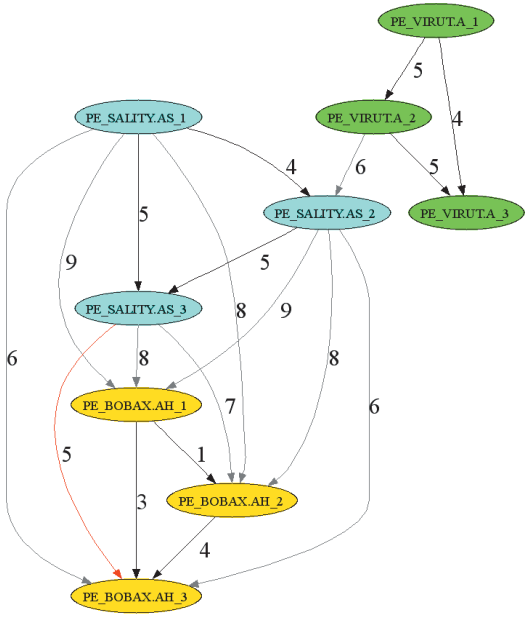

図 7　ハミング距離によるマルウェア分類の視覚化例
Fig. 7　An example of visualization of malware classification based on hamming distance.

4.4　実験の手順

最初に，4.1 節で述べた主要素について，それぞれを単独にカテゴリ・データとした場合の評価を行った．この際，出現頻度トップ N における最適な N を得るため $N = \{10, 20, 50, 100\}$ について評価を行い，この中から最適な N を求めた．続いて，主要素を中心として複数の要素を組み合せた場合の評価を行った．

実験の手順は，次のとおりである．

［STEP-1］1,503 個の実験データの中から，10 種類のマルウェアについて，それぞれの種類からランダムに 10 個ずつ，合計 100 個の個体を抽出．

［STEP-2］STEP-1 で抽出した合計 100 個について相互のマルウェアの距離マトリクスを算出し，距離の最小値（Rmin）と最大値（Rmax）を得る（距離の算出はハミング距離と NCD の 2 種類を用いて比較を行った）．

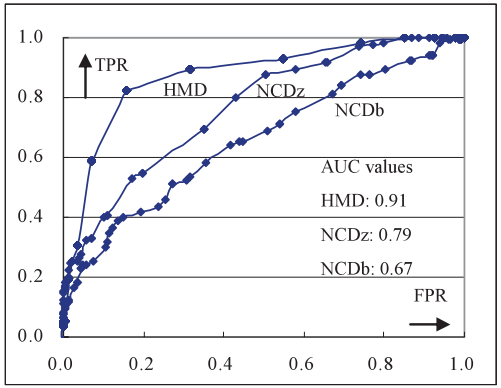

図 8　ROC 曲線と AUC 値の一例
Fig. 8　An example of a ROC curve and AUC values of malware classification.

表 5　要素単体による AUC 値
Table 5　The AUC values in the case of a single major element.

R	N10	N20	N50	N100
HMD	0.68	0.75	0.82	0.81
NCDz	0.67	0.72	0.81	0.82
NCDb	0.70	0.72	0.78	0.72

M	N10	N20	N50	N100
HMD	0.72	0.80	0.86	0.85
NCDz	0.64	0.75	0.76	0.77
NCDb	0.64	0.68	0.78	0.75

T	N10	N20	N50	N100
HMD	0.80	0.82	0.82	0.83
NCDz	0.53	0.57	0.59	0.61
NCDb	0.70	0.70	0.74	0.70

P	N10	N20	N50	N100
HMD	0.66	0.65	0.68	0.68
NCDz	0.58	0.36	0.69	0.65
NCDb	0.60	0.63	0.66	0.68

［STEP-3］同種類として判断する閾値 TH を Rmin から Rmax まで変化させて，TPR（TH）と FPR（TH）を得る．

［STEP-4］STEP-3 の結果から，TH をパラメータとした ROC 曲線をプロットし，この図から AUC 値を算出する．

以上のステップを 20 回繰り返し，AUC 値の平均値を求めて，分類精度の指標とした．

4.5　要素単体による分類

前節で述べた手順により，要素単体で分類した場合の AUC 値の平均値を**表 5** に示す（NCDz は NCD に利用した圧縮アルゴリズム zlib を，NCDb は同 bzlib を表す）．出現頻度トップ N における N の選び方によって，AUC 値が変化するが，**R** および **M** については

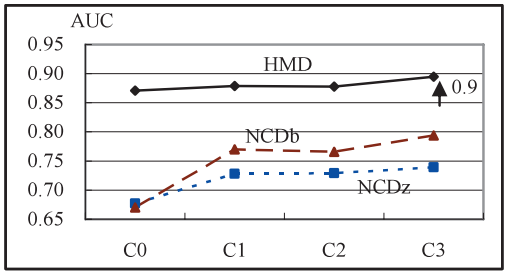

図 9　複数要素の組合せによる AUC 値
Fig. 9　AUC values in the case of combination of multiple elements.

$N = 50$ の場合に最大で，AUC 値はそれぞれ 0.82，0.86 となった．**T** については $N = 100$ の場合に AUC 値が 0.83 で最大となった．**P** による分類では，他の主要素と比較して AUC 値が低い値となり，分類の主要素としては適さないことが判明した．これは，プロセス名として，実行のつどランダムな文字列が使われることが多いためである．距離算出の手法による違いとしては，ほとんどの場合，HMD の方が NCD よりも高い AUC 値となった．

4.6　複数要素の組合せによる分類

AUC 値をさらに高めるために，複数の要素を組み合せた場合の実験を行った．複数要素の組合せは厖大な数となるため，**R**，**M**，**T** について単一要素の場合に最大の AUC 値となった N を用いて，**K**，**S**，**H** の各補助要素を順次追加して，AUC 値を算出した．組合せの中で最も高い AUC 値を得た例を**図 9** に示す．**M**（$N = 50$）と **T**（$N = 100$）を組み合せた場合に AUC 値が最大となり，補助要素を追加することによって逐次 AUC 値が増加し，最大で 0.9 を得た．また，HMD と NCD との比較では，HMD の方がつねに高い AUC 値となった．

4.7　目隠しテストによる未知のマルウェアの分類

前節では AUC 値を指標として，マルウェアの分類に適した PDB の要素の構成を求めた．本節では，これまでの実験で，最大の AUC 値となった要素の組合せ $\{\mathbf{M}\,(N = 50), \mathbf{T}\,(N = 100), \mathbf{K}, \mathbf{S}, \mathbf{H}\}$ を用いて，ハミング距離によるマルウェアの分類を試みた結果について述べる．ここでいう「目隠しテスト」とは，4.1 節で述べた検体 3,665 個のすべてを使い，個々の検体の名称が不明だったものと仮定し，提案の手法でマルウェアの名称を推定した結果が，ウィルス・スキャン結果と一致する割合を算出したものである．一致の割合を算出する際に，名称が完全に一致する場合と，科名までが一致する場合の 2 種類について確認を行っ

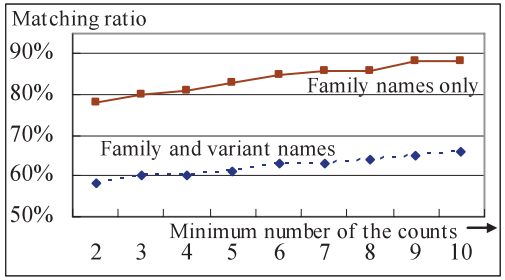

図 10　分類の一致率
Fig. 10　Matching ratio of the classification.

表 6　個体数の下限に対応したマルウェアの種類数
Table 6　Number of malware names corresponding to the minimum number of the counts.

Minimum number of the counts	2	3	4	5	6	7	8	9	10
Number of family & variant names	181	104	83	64	50	41	35	31	27
Number of family names	78	64	53	47	38	31	29	25	22

た．また，提案手法では，距離が最短となるマルウェアの種類が複数の場合には，PDB の中の種類ごとの個体数が多い種類として分類する．このため種類ごとの個体数が，分類の精度に影響を与える．この影響を確認するため，個体数の下限を横軸として一致率をプロットした結果を**図 10** に示す．図 10 において，上側のプロットは，提案手法で推定したマルウェアの科名までが一致した割合を示し，この値が 78%〜88%となる結果を得た．また，同図の下側のプロットは，科名と亜種名が完全に一致した割合を示し，その値は 58%〜66%の範囲となった．

この目隠しテストに使われた，マルウェアの種類数を**表 6** に示す．この表は，図 10 と同様に，同じ種類の個体数の下限を 2〜10 としたときに分類対象となった種類数であり，科名では 78〜22 種類，亜種名を含む場合では 181〜27 種類であった．

4.8　HMD と NCD の処理所要時間の比較

HMD と NCD の計算式上の違いは，3.3 節で示したが，実際の処理所要時間を比較した結果を**図 11** に示す．この図は 20 種類のマルウェアをそれぞれ 10 個，合計 200 検体間の相互の距離を算出したものであり，縦軸は処理所要時間 [秒]，横軸はカテゴリ・データのビット数を表している．ここで，NCD は C 言語のコンパイラで記述された Libcomplearn [24] のパッケージを利用し，HMD はスクリプティング言語 PHP5.1.6 [26] で記述した場合の結果である．使用言語の違いがあるものの HMD の方が NCD よりも短時間で処理が可能であることが示されており，HMD の処理を C 言語などで記述することによって，さらに高速化が可能である．

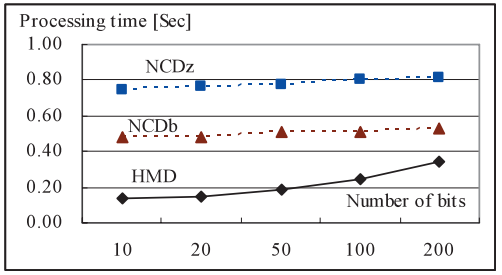

図 11　HMD と NCD の計算所要時間の比較
Fig. 11　The comparison of the classification time required between HMD and NCD.

4.9　実験結果のまとめ

以上の実験から，主要素単体では，マルウェアの実行によって顕在化した **M**（システム関連ファイル）の変化を示す文字列の出現頻度トップ 50 をカテゴリ・データとした場合に，AUC 値が最大となるという結果を得た．また主要素へ補助要素を付加することで AUC 値が増加し，2 種類の主要素の組合せでは，**M** ($N = 50$) と **T** ($N = 100$) の場合に高い AUC 値となった．さらに，これに補助要素 **K**，**S**，**H** を加えた場合に，AUC 値が 0.9 で最大となった．HMD と NCD との比較では，提案の PDB 形式では HMD の方が高い AUC 値となった．NCD 法で利用した圧縮アルゴリズム zlib と bzlib との比較では，表 4 に示したように一方がつねに高い AUC 値とはならず，データによって良否が変化した．

また，処理速度の比較では HMD をスクリプト言語で実装した場合でも，コンパイラで実装した NCD よりも数分の 1 以下の時間で処理が可能であった．HMD による分類は NCD と比較して，高速に演算が可能でかつ高い AUC 値が得られることから，本提案の PDB のデータ様式を利用した分類法としては，NCD よりも HMD 法の方が良い結果を得ることができる．

また，分類精度に関し文献 27) では，SVM 法で 14 種類について実験し，科名の一致率としてほぼ 70%を達成したことが述べられている．一方，筆者らの目隠しテストによる実

践的な評価では，提案手法で推定した名称とトレンドマイクロ社製のウィルス・スキャン結果が完全（科名＋亜種名）に一致する割合は 60%程度，科名までが一致する割合は 80%以上という結果を得た．

ただし，これらの分類精度は，HMD，NCD，SVM などの分類の手法だけでなく，マルウェアを実行する環境，特徴の抽出手法，挙動の数値化の手法あるいは分類の対象となった検体の種類とその数の分布などによって影響を受ける．したがって，これらの数値は，必ずしも同一条件下での比較ではないことに注意が必要である．

5. マルウェア自動解析システムへの実装

提案の手法をマルウェアの自動解析システムへ実装した．システムの概要を**図 12** に示す．このシステムは，OS として Linux（FedoraCore 6）を利用し，このホスト OS へインストールした仮想マシン（VMware Server）上の WindowsXP（サービスパックなし）でマルウェアを実行して挙動を観測する．

ホスト OS 上へインストールしたスクリプトが解析処理全般の制御，各種サーバの模擬およびネットワーク環境の模擬を行うとともに，マルウェアの実行後に挙動の変化の解析と類似性の判定を実行する．類似性の判定は，WEB ブラウザをユーザインタフェースとし，インタラクティブな操作によって実行される．

解析の対象として，たとえば研究用データセット CCC DATAset2008 の攻撃元データ[30]では，このログに 52,465 種類のハッシュ値の異なる検体が記録されている．このような規模のデータを想定し，一般的な PC 1 台を使って，インタラクティブな操作（応答時間数秒以下）でマルウェアの類似性の判定が可能なシステムへの適用を考えた場合，現状では所要時間が数十秒以上となることが予想されるため，ハミング距離による類似性の判定は，NCD との比較では高速であるが，さらに高速化の工夫が必要である．

3.2 節で述べたように PDB の構成要素数はマルウェアの個体数と同一であり，多種類のデータが蓄積された場合には，これに比例した計算量となる．一方，マルウェアの個体が違っても挙動が同じ場合には，カテゴリ・データに変換されたバイナリのパターンは等しいものとなる．そこで PDB の構成要素から重複したパターンを除去したリストを生成し，これを N_PDB（Normalized PDB）と呼ぶ．仮に同じ種類のマルウェアの挙動はつねに同じであり，かつ異種類のマルウェアの挙動とは必ず異なるものであれば，N_PDB のサイズは，マルウェアの種類数と一致する．しかし実際には，4.2 節に示したように同じ名称を持つマルウェアの中にも相当なバラツキや揺らぎが存在し，マルウェアの種類数を κ とすると次の関係となっている．

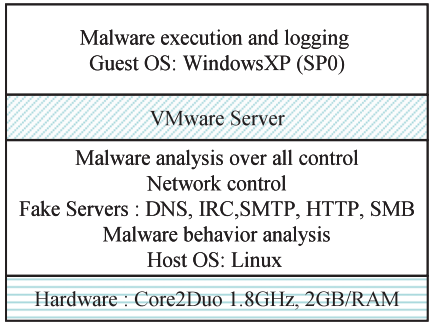

図 12　マルウェア自動解析システムの概要図
Fig. 12　An overall block diagram of automated malware variant classification system.

$$|\mathrm{PDB}| > |N\_\mathrm{PDB}| > \kappa \tag{3}$$

N_PDB のリストを生成する際に，カテゴリ・データが同じ値となる個体をマルウェアの種類別にカウントし，これをスコア値として，最もスコアの高いマルウェアの種類を N_PDB の構成要素ごとに割り付けるものとする．これは，提案手法を説明した 3.1 節［STEP-6］において，「距離が最短のマルウェアが複数種類存在する場合には，種類ごとの個体数が多い種類を候補として出力する」と等価である．すなわち，あらかじめ PDB から N_PDB を生成しておくことによって，分類対象のマルウェアに対応したカテゴリ・データの値が，N_PDB に含まれている場合には，距離の計算をすることなく，テーブルの参照だけで，高速に分類が可能となる．また，新種のマルウェアを解析する場合などでカテゴリ・データの値が，N_PDB に含まれない場合でも，PDB よりもサイズが小さい N_PDB を対象としたハミング距離の算出で済むため，処理時間を大幅に短縮が可能である．4.7 節で述べた目隠しテストを対象として，PDB のみの場合と N_PDB を併用した場合について，処理所要時間を実測した結果を**図 13** に示す．カテゴリ・データのビット数が 50 ビット以下では，PDB 単独の場合よりも N_PDB を併用した方が 1/1000 以下の時間で処理が完了している．ここで，ビット数が多い場合に，処理時間の差が縮小している．この原因は，ビット数が多くなるとテーブル参照だけでは分類ができず，距離の計算を必要とする対象が増加しているためである．このように，N_PDB を併用する工夫によって，インタラクティブな操作によるマルウェアの自動解析システムへの適用が可能となる．

以上の手法をマルウェア自動解析システムへ適用し，市販の AV 製品では，マルウェアを

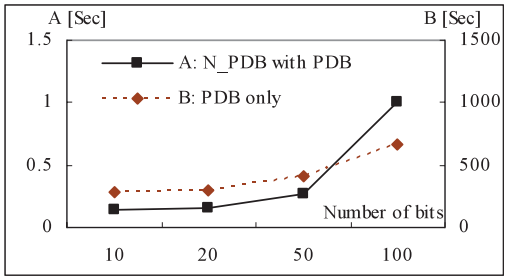

図 13　分類所要時間の比較
Fig. 13　The comparison of the classification time required.

検出できなかったファイルを解析した場合の表示例を**図 14** に示す．

図 14 において ① は，解析対象ファイルのハッシュ値と 4 種類の AV 製品でスキャンした結果を示し，N.A. の文字列は，マルウェアを検知しなかったことを示している．② は，提案手法でマルウェアの名称を推定した結果であり，この場合は距離がゼロ（Distance 0）でスコアが最大（Score 3）となっている PE_BOBAX.AH である可能性が高いことを示している．③〜⑥ は，このマルウェアの挙動として観測されたレジストリやシステム関連ファイルの改ざん，起動プロセスの情報などである．

近年のマルウェアは，ポリモーフィック型や難読化によって非常に多くの種類が発生しており，従来のように，シグネチャベースのパターンマッチングによる検知だけでは対応が困難であるといわれている．したがって，市販の AV 製品でスキャンし，マルウェアが検出されなかったということだけでは，安心できない状況となりつつある．このような背景にあって，本システムを利用することによって，市販の AV 製品ではマルウェアとして検出できない場合でも，自動解析で得られた通信の相手先やポート番号を参考として，感染が疑われる PC の発見や，ネットワーク機器の設定変更などの対策に役立てることができる．また，感染が疑われる PC 内のレジストリやその他のファイルの状態を確認することによって，感染した PC を特定することが可能となる．さらに，感染時の挙動の類似性からマルウェアの名称を推定することが可能となり，この名称を手がかりとして，コード解析のような，より詳細な解析を行う場合や，対策を検討する際に，既知の情報の活用が容易となり，マルウェア対策の効率化に寄与できるものと考えられる．

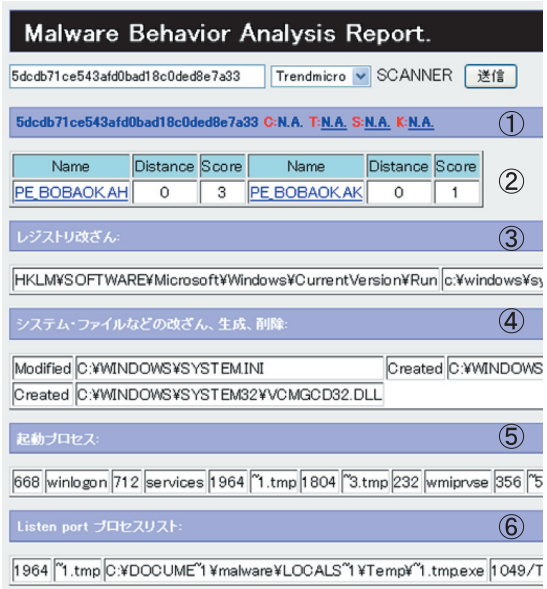

図 14　マルウェア自動解析システムの表示例
Fig. 14　An example of screen shot of the automatic malware analyzing system.

6. おわりに

仮想マシン上の Windows 環境で実行・取得した，マルウェアの動的挙動に関する情報を，カテゴリ・データへ変換し，相互のハミング距離を算出することによって，マルウェアの種類を推定する手法についての提案を行い，その有効性を実験によって検証した．さらに提案手法をマルウェアの自動解析システムへ実装し，マルウェアの実行にともなって顕在化する，レジストリの改ざん，プロセスの起動などの情報とともに，市販の AV 製品では，マルウェアとして検出できない検体についても，蓄積された PDB 内のマルウェアの挙動との類似性から推定したマルウェアの名称を，自動的に提示できることを示した．マルウェアの種類ごとの個別の学習フェーズなしに，亜種名を含むマルウェアの名称を自動的に提示できる点が，文献 27) と比して本提案特有の機能である．本提案の手法は，NCD などよりも計算量が少なく，かつ分類の精度も高いことから，マルウェア分類の自動化，対策の立案およびコード解析など，さらに詳細な解析の初期フェーズに有効と考えられる．

本提案では，仮想マシン上の Windows でマルウェアを実行・解析したが，仮想マシン上では挙動を観測できない種類のマルウェアについても解析を可能とするための，実マシン上の自動解析環境の構築が今後の課題である．また，本提案では，インターネットの定点観測で捕獲したマルウェアを解析の対象とし，人手を介さずに自動的に解析・分類を行うシステムとして実装したが，メール添付や WEB サイトからダウンロードしたマルウェアなどで，対話形式でインストールの操作を必要とする対象についても解析を可能とする機能の拡張や，推定精度に影響する既存のマルウェアの挙動データベースの充実が今後の課題である．

参 考 文 献

1) Yoo, I.-S.: Visualizing Windows Executable Viruses Using Self-Organizing Maps, *VizSEC/DMSEC'04*, Oct. 29, 2004, pp.82–89 (2004).
2) Kolter, J.Z. and Maloof, M.A.: Learning to Detect and Classify Malicious Executables in the Wild, *Journal of Machine Learning Research*, Vol.7, pp.2721–2744 (2006).
3) Wehner, S.: Analyzing Worms and Network Traffic using Compression, *Journal of Computer Security*, Vol.15, No.3, pp.303–320 (2007).
4) 星澤裕二，太刀川剛，山村元昭：マルウェアの亜種等の分類の自動化，情報処理学会研究報告，2007-CSEC-38, pp.271–278 (2007).
5) Lee, T. and Mody, J.J.: Microsoft Corp., Behavioral Classification, EICAR Conference (May 2006).
6) Bailey, M., Andersen, J., Mao, Z.M., Jahanian, F. and Nazario, J.: Automated Classification and Analysis of Internet Malware, *Proc. 10th Symposium on Recent Advances in Intrusion Detection* (*RAID'07*), pp.178–197 (2007).
7) Vitányi, P.（著），渡辺　治（訳）：圧縮度にもとづいた汎用な類似度測定法，数理科学，No.521 (Nov. 2006).
8) Mell, P., Lippmann, R., Harines, J. and Zissman, M.: An Overview of Issues in Testing Intrusion Detection Systems, *NIST IR* (2002).
9) TrendMicro. http://jp.trendmicro.com/jp/home/index.html
10) Kaspersky. http://www.kaspersky.co.jp/
11) Symantec. http://www.symantec.com/ja/jp/index.jsp
12) ClamAV AntiVirus. http://www.clamav.net
13) Graphviz-Graph Visualization Software. http://www.graphviz.org/
14) http://www.cwsandbox.org/
15) http://www.norman.com/microsites/nsic/
16) http://www.cyber-ta.org/
17) Gheorghescu, M.: An Automated Virus Classification System, *Virus Bulletin Conference* (Oct. 2005).
18) 井上大介，衛藤将史，吉岡克成，星澤裕二，伊沢亮一，森井昌克，中尾康二：Micro Analysis for Analyzing Malware Code and its Behavior on NICTER, *Proc. SCIS2007*, 2F2-1 (Jan. 2007).
19) Jiang, X., Xu, D., Wang, H.J. and Spafford, E.H.: Virtual Playgrounds For Worm Behavior Investigation, *8th International Symposium on Recent Advances in Intrusion Detection*, Vol.3858/2006, pp.1–21 (2006).
20) Honeynet project: Know your Enemy: Tracking Botnets. http://www.honeynet.org/papers/bots/
21) VMware. http://www.vmware.com/
22) Nepenthes. http://nepenthes.mwcollect.org/
23) Stewar, T.J.: Truman, The Reusable Unknown Malware Analysis Net. http://www.lurhq.com/truman/
24) Libcomplearn. http://www.complearn.org/
25) The DOT Language. http://www.graphviz.org/doc/info/lang.html
26) PHP. http://www.php.net/
27) Rieck, K., Holz, T., Willems, C., Dussel, P. and Laskov, P.: Learning and Classification of Malware Behavior, *DIMVA2008*, *Detection of Intrusions and Malware and Vulnerability Assessment*, Vol.5137/2008, pp.108–125 (2008).
28) CWSandbox – Behavior-based Malware Analysis. http://www.cwsandbox.org/
29) 堀合啓一，今泉隆文，田中英彦：定点観測によるボットネットの観測と Malware の動的挙動解析システムの提案，情報処理学会論文誌，Vol.49, No.4, pp.1680–1691 (2008).
30) 永田　大，堀合啓一，田中英彦：OLAP（多次元データ分析）を利用した攻撃元データの分析と検体の自動解析，MWS2008 論文集，pp.61–66 (Oct. 2008).

(平成 20 年 9 月 5 日受付)
(平成 21 年 1 月 7 日採録)

堀合　啓一（正会員）

1973 年防衛大学校電気工学科卒業．同年航空自衛隊入隊．1979 年防衛大学校理工学研究科電子工学専攻修了．2005 年から防衛省技術研究本部電子装備研究所勤務．2009 年情報セキュリティ大学院大学博士課程修了，博士（情報学）．

今泉　隆文

2004 年東京大学工学部計数工学科卒業．2006 年同大学大学院情報理工学系研究科博士前期課程修了．同年防衛省技術研究本部入省．現在，同省電子装備研究所勤務．

田中　英彦（フェロー）

1970 年東京大学大学院博士課程修了，工学博士．東京大学大学院情報理工学系研究科教授・研究科長を経て，2004 年情報セキュリティ大学院大学教授．計算機アーキテクチャ，分散処理，知識処理，デペンダブル情報システム等に興味を持つ．著書に『非ノイマンコンピュータ』『計算機アーキテクチャ』『Parallel Inference Engine』等がある．日本学術会議会員，電子情報通信学会，情報処理学会，人工知能学会，IEEE 各フェロー．