Kullback-Leibler 情報量を用いた亜種マルウェアの同定

中村 燎太 †　　松宮 遼 †　　高橋 一志 †　　大山 恵弘 †

† 電気通信大学
182-8585 東京都調布市調布ヶ丘 1-5-1
{nakaryon, r.matsumiya}@ol.inf.uec.ac.jp　　{kazushi, oyama}@inf.uec.ac.jp

あらまし 本稿では，全マルウェア検体の中から選択した基準検体と他検体の Windows API コール列に関するディリクレ分布の類似度を Kullback-Leibler 情報量 (KL 情報量) によって示すことで，基準検体に類似する亜種を同定した実験について報告する．本実験におけるディリクレ分布は，検体の API コール列のパターンを多項分布と仮定し，ベイズ統計学における共役事前分布の計算によって得られる事後分布のことを指す．また，KL 情報量は確率分布 P，Q の統計的距離を示す値である．なお，実験では FFRI Dataset 2013 を検体として利用した．

Identification of Subspecific Malware by Utilizing Kullback-Leibler Divergences

Ryota Nakamura†　　Ryo Matsumiya†　　Kazushi Takahashi†
Yoshihiro Oyama†

†The University of Electro-Communications
1-5-1 Chofugaoka, Chofu-shi, Tokyo 182-8585, JAPAN
{nakaryon, r.matsumiya}@ol.inf.uec.ac.jp　　{kazushi, oyama}@inf.uec.ac.jp

Abstract We report the result of experiments in which we detected subspecific malware of a fiducial one included in FFRI Dataset 2013. Specifically, we found Kullback-Leibler divergences between two Dirichlet distributions of the base malware and every other malware in the dataset by calculating posterior probability distributions. Kullback-Leibler divergence represents a non-symmetric distance between two probability distributions P and Q. To find the posterior probability distribution, It is necessary to plug the multinomial distribution as patterns of Windows API calls and the conjugate prior distribution of the multinomial distribution into Bayes' theorem and to calculate this theorem.

1　はじめに

近年，マルウェアの発見数は増加傾向にあり [7]，それに伴って各種アンチウイルスソフトウェア (AV ソフトウェア) に登録されるパターンファイル数も増加の一途を辿っている．特に，マルウェアの簡易生成キット [12] 等を用いることで，亜種の関係にあるマルウェア群を大量に生成出来ることも，マルウェアの増加を助長する一要因であると考えられる．従って，発見された全てのマルウェアを一から分析し，パターンファイルを作成することは人的コストを考えると難しい．本問題は，アノマリ型のマルウェア検知システムを導入することで対処可能だが，誤検知や検知漏れの多さを鑑みると，パターンマッチング型のマルウェア検知システムも未だに重要である．

そこで，本稿では既知の検体 A と解析対象の検体 B の Windows API コール列を入力として与えると，A に対する B の類似度をベイズ統計学的手法によって算出する手法を提案する．もし，B が A に類似していることが判明した際には，A の分析結果を活用することで，B に対するパターンファイルの生成作

業を効率化することが出来る．また，解析対象の検体が複数存在する場合には，一定の境界値よりも類似度の高い検体を亜種として抽出し，それを検体の分析に活用することも可能である．

提案手法と Support Vector Machine (SVM) による亜種識別プログラムをそれぞれ開発し，FFRI Dataset 2013 [14] を対象として亜種の識別精度を比較した．SVM は識別器のデファクトスタンダードであり，亜種の識別精度を比較する対象として妥当である．結果，本手法は誤検知がやや多いものの SVM に相当する識別精度を示した．また，提案手法は SVM よりもメモリ消費量が少なく，高速な識別が可能であることも判明した．

なお，本稿は全 5 章によって構成されている．次章では提案手法の概要を説明し，3 章では各種実験の結果を示すとともに，提案手法の評価を行う．4 章では関連研究について説明する．そして，5 章では本稿のまとめと今後の課題を提示する．

2 提案手法

本章では，まず Kullback-Leibler 情報量の概要を示す。これは 2 つの確率分布間の距離を示す数値であり、提案手法にて求められる検体の類似度に相当する。次に提案手法の比較対象として利用する SVM について述べ、続けて提案手法の詳細を説明する．

2.1 Kullback-Leibler 情報量

Kullback-Leibler 情報量 (KL 情報量) は相対エントロピーの一種であり，確率分布 P から同分布 Q までの統計的距離を表す数値である．例えば，連続関数分布 P, Q 間における KL 情報量 $D_{KL}(P||Q)$ は，P, Q の確率密度関数 $p(x), q(x)$ を用いることで次のように表すことが可能である．

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) ln \frac{p(x)}{q(x)} dx \quad (1)$$
$$(p(x) > 0 \text{ かつ } q(x) > 0)$$

本数値は，一般的に $D_{KL}(P||Q) > 0$ を満たすが，$P = Q$ の時には 0 となる．また，$D_{KL}(P||Q) \neq D_{KL}(Q||P)$ が常に成立する．

2.2 Support Vector Machine

Support Vector Machine (SVM) は，教師あり学習による 2 クラスのパターン識別器の一種である．本機械は，学習機械の中でも高い識別能力を誇ることで知られている [13]．SVM を利用するためには，まず学習用データを識別器に学習させ，続けて判定対象データ (識別用データ) を入力する必要がある．例えば，A， B というラベルが付加された各特徴ベクトル群を識別器に予め学習させておけば，その後に入力した判定対象の特徴ベクトルについて，当該ベクトルが A, B のいずれに属するのかを識別器が自動的に判定する．

また，SVM は「学習用データとして入力された全ての特徴ベクトルを線形分離する識別関数 $f(x)$ が，$f(x) = 0$ を満たす超平面」を探索することで学習を行うため，特徴ベクトルが多次元であるほど正確な識別を行うことが出来るという特徴を持つ．従って，SVM はマルウェアによる API コール列のように，多くの情報 (次元) を含むデータを学習・識別する用途との相性が良い．

但し，学習させる特徴ベクトルが多次元であっても，全ベクトルの線形分離が可能であるとは限らない．また，学習用ベクトルの次元数が極度に大きい場合には，識別器が過学習に陥り，識別用データを正しく分類できないことがある．それらのケースでは，線形分離が不能な特徴ベクトルに対して一定のペナルティ C を科すことで，$f(x)$ の決定要件を緩和する必要がある．当該方針を採用した SVM はソフトマージン SVM (C-SVM) と呼ばれる．また，C は学習用データに依存するため，それを更新した場合には C を再設定する必要がある．

ところで，SVM の学習精度を測る際には，k-交差検定という手法を利用する．これは「学習用データとして入力した特徴ベクトル群を k 個のブロックに分割し，うち $k-1$ 個のブロックを識別器に学習させ，残る 1 個を評価に使う」作業を全てのブロックについて繰り返すことで学習精度を算出する手法である．本手法は学習用データに対する最適な C を求めるために用いられる．具体的には，とある学習用データ L を入力とした k-交差検定によって「C と，それに対応する学習精度」の組を多量に生成する．そして，全組の中で最も学習精度が高い組の C が，L に対する最適値に相当する．

2.3 提案手法の詳説

始めに，全検体の API コール列から N-gram を抽出する．N-gram は Shannon によって提唱された概念であり,「一塊の情報内に含まれる構成要素を，先頭から順番に N 個の要素群へと分割し直したもの」のことを指す．本稿では，こうして生成した N 個の構成要素から成る情報群を「ウィンドウ」，N を「ウィンドウサイズ」と表記する．例えば，文字列 “ABBCCCCD” を 3-gram に分割すると，“ABB”, “BBC”, “BCC”, “CCC”, “CCD” という 5 種類のウィンドウを得られる．なお，本プロセスにおけるウィンドウサイズ N には適切な値を設定すべきだが，それについては実験結果から求めるものとする．

次に，抽出済みの全ウィンドウに N-gram ID という $1 \ldots n$ の番号を割り振る．すなわち，当該ウィンドウ群には，少なくとも 1 つの検体に含まれるウィンドウが全て網羅されている．そして，検体毎の API コール列に含まれる N-gram ID の出現数 $x_i\,(1 \leq i \leq n)$ をカウントすることで生成したパターンデータを，尤度関数としての多項分布 $Pr(X|\theta)$ $(X \in [x_1 \ldots x_n], \theta \in [0, 1])$ と仮定する．具体的には,「m 列から成る検体の API コール列について，先頭から順番に N-gram を生成する際に，生成されたウィンドウがどの N-gram ID $(1 \ldots n)$ に該当するのかを予想する問題」を,「人間が n 面を持つ賽子を振った時に，いずれの面が出るかを確かめる問題」へと捉え直し，賽子を $m - N$ 回振るという独立試行において各々の面が出る確率を $Pr(X|\theta)$ として設定する．そして，当該分布を式 2 に示したベイズの公式へ代入する．

$$Pr(\theta|X) = \frac{Pr(X|\theta)Pr(\theta)}{Pr(X)} \tag{2}$$

ここで，多項分布の共役事前分布にあたるディリクレ分布 $Dir(\alpha)$ を事前分布 $Pr(\theta)$ に設定すると，事後分布 $Pr(\theta|X)$ もまたディリクレ分布となる．α はディリクレ分布のパラメータであり，今回は事前分布が一様分布となるように $\alpha = 1$ とした．

各検体は API コール列についてのディリクレ分布 $Pr(\theta|X)$ を 1 つずつ持つことになるため，既知の検体と解析対象の各検体の分布について KL 情報量を算出する．KL 情報量は 2 つの分布間における差異の大きさを表すため，本値が相対的に小さければ，当該検体が既知検体の亜種/変異種である可能性が高いと言える．

表 1: 実験環境

CPU	Intel Core i7-860 2.8GHz
Memory	DDR3-1600 8GB
OS	Microsoft Windows 8 (64bit)
Runtime	.NET Framework 4.5 (64bit)
	Python 2.7.5 (64bit)

表 2: 識別プログラム内で使用したライブラリ

System	System.Collections
System.Data	System.IO
System.Linq	System.Text
System.Threading	System.Diagnostics
LingPipe API[1]	Libsvm[2] 3.1.6 [5][9]
Libsvm.net [6]	IKVM.NET[2] [4]

3 実験

本稿における実験の目的は，提案手法と C-SVM のそれぞれを用いた亜種の識別実験を行い，その結果を比較することで提案手法を評価することである．本章では，実験の手順，実験結果，提案手法の評価について記述する．

3.1 全実験における共通事項

実験では表 1 に示す環境を使用し，検体の識別結果，同環境下における処理時間，最大メモリ消費量を記録・分析した．実験にて開発した亜種の識別プログラムは C#で記述しているものの，C-SVM における k-交差検定の処理には Python スクリプトを利用した．また，C#にて制作したプログラムでは，表 2 に挙げるクラスライブラリを利用した．

実験にて使用する検体群の収集と，各検体による API コール列の収集については，FFRI Dataset 2013 [14] を用いることでこれに代えた．本データセットには，国内のハニーポットにて採取した 2,644 個のマルウェア検体を Cuckoo Sandbox [3] にて解析した結果が JSON として収録されており，各検体の API コール列もそれに含まれている．但し，実験対

[1]本来は Java 向けの API だが，KL 情報量の計算に関するメソッドを C#へ移植して利用した．

[2]Libsvm.net が利用する．本研究にて開発したプログラムからは直接参照していない．

象の検体は，以下の要件を全て満たす 1,974 検体に限定した．

- JSON ファイルのフォーマットが破損していないもの
- Windows API の動的解析結果が誤りなく記録されているもの
- Trend Micro, Symantec, McAfee, Kaspersky が販売する AV ソフトウェアによる検体の検査結果が含まれるもの

Trend Micro, Symantec, McAfee は日本市場におけるシェアの約 75%を占めている [17] ことから，多数の国産マルウェアを解析する機会に恵まれていると考えられる．Kaspersky は AV-TEST [2] や AV-Comparatives [1] によってマルウェアの検査能力が保証されている．従って，本実験ではこれらのベンダーによる検体の検査結果を正しいものと仮定した．

そして，選別した検体のうち，検体 ID が 306 番のものを基準検体とした．なお，基準検体として如何なる検体を選択したとしても，実験の本質に影響を及ぼすことはない．残りの各検体は，当該検体を基準検体の同種/亜種であると検出したベンダー数 H(一致ベンダー数) に基づき，表 3 に示す通りに分類した．判定対象の検体を基準検体の同種と判定した条件は「対象検体の科名及び亜種名が基準検体のそれと一致すること」であり，亜種の判定条件は「対象検体の科名が基準検体と一致すること」である．便宜上，各分類には H に基づくグループ名を付けており，例えば $H = 4$ の検体群は G4 と命名した．実験では，所属グループ毎に実験プロセスにおける処理対象を選択した他，亜種の識別結果が適当であるか否かを判定するためにも所属グループを利用した．詳細については後述するが，例えば実験にて亜種と認定されたにも関わらず，当該検体の所属グループが G0 の場合には，検体の識別に失敗した可能性が疑われる．なお，G2 に属する検体については，当該検体が実際に亜種であるか否かを推測する術がなく，性能評価の指標として利用出来ないため，実験の対象外とした．G1 と G3 に属する検体については，ベンダー間にて検出結果の統一が取れていないものの，今回は多数決によって G1 の検体を基準検体の非亜種，G3 の検体を亜種として取り扱った．

また，C-SVM の実験にて学習用データと識別用データを用意しなければならない都合上，G0 と G4 についてはグループ全体を T(学習用データ) と P(識別用データ) の小グループに細別した．基準検体は G4T に含めたものの，他検体については実験の恣意性を排除するため，無作為に分類した．T と P の検体数比は 3 : 7 で，本比率は学習用データ及び識別用データの分量に偏りが生じることを防ぐために設定したものである．

表 3: 検体のカテゴライズ結果

グループ名	該当検体数
G4	54 (G4T) / 126 (G4P)
G3	107
G2	276
G1	129
G0	388 (G0T) / 894 (G0P)

なお，両実験における N-gram の生成プロセスは 3-gram から 7-gram のそれぞれについて実施し，最終的な識別結果も各ウィンドウサイズについて求めた．それらの結果は，手法の評価にて適切なウィンドウサイズを決定する際に利用した．

3.2 提案手法による実験

基準検体及び G4P, G3, G1, G0P に属する各検体の N-gram におけるディリクレ分布について，KL 情報量を算出した．本実験では，検体を判定するために，KL 情報量の境界値を「G4 の検体群のうち，最も KL 情報量が大きいもの」と定めた．そして，基準値よりも小さな KL 情報量を持つ検体を亜種，基準値よりも大きな KL 情報量を持つ検体は非亜種と識別した．

最後に，検体の所属グループと識別結果の関係を “True Positive (T-P)”，“False Negative (F-N)”，“True Negative (T-N)”，“False Positive (F-P)” の 4 類型に区分した．まず，G4P, G3 の検体 (亜種) の中で正常に判定されたものを “T-P”，非亜種と誤判定されたものを “F-N” へ分類し，続けて G1, G0P に属する検体 (非亜種) を正しく判定したものを “T-N”，亜種と誤判定したものを “F-P” へと分類した．そして，各分類に含まれる検体数を集計し，それを実験の最終結果とした．

以上の実験手順を纏めると，図 1 の通りとなる．

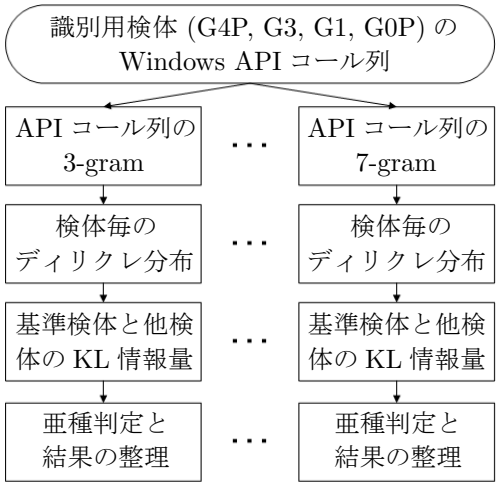

図 1: 提案手法による実験の概要図

3.3 C-SVM による実験

本実験では，Libsvm.net [6] というライブラリを利用して，C-SVM による機械学習を行った．Libsvm.net は Chang らが製作した Libsvm [5] [9] という C++/Java 向けの SVM ライブラリを C#向けにラッピングしたものである．

まず，G4T 及び G0T の検体に関するディリクレ分布にそれぞれ “1 (亜種)” と “−1 (非亜種)” のラベルを付与し，それを特徴ベクトルに変換したものを学習用データとした．そして，Libsvm に付属する grid.py という Python スクリプトを用いることで，最も学習精度が高くなるようなペナルティ C を求めた．grid.py は C の値を次々と変更しつつ，学習用データについて k-交差検定を行うスクリプトであり，出力として学習精度の実測値と最適な C を得ることが出来る．表 4 は実際に grid.py を使って学習用データから C と学習精度を求めた結果である．本表より，パラメータ C が適切な場合には，全てのウィンドウサイズ N について識別器の学習精度が 99%前後となることが分かる．従って，識別器の学習は正常に行われており，学習の不備によって実験結果に何らかの影響が及んだ可能性は低い．

次に，表 4 に示した C の下で学習用データを識別器へ入力し，続けて G4P, G3, G1, G0P に属する各検体を識別器で判定したところ，全検体が “1 (亜種)”，もしくは “−1 (非亜種)” のいずれかに分類された．最後に，識別結果と検体の所属グループを基に，提案手法と同じ要領で各検体を “T-P”，“F-N”，“T-N”，“F-P” の 4 類型に区分し，各分類の検体数

表 4: SVM のペナルティ C と学習精度

N	C	学習精度
3	32768	98.87%
4	32768	99.55%
5	32768	99.10%
6	32768	99.32%
7	32768	99.32%

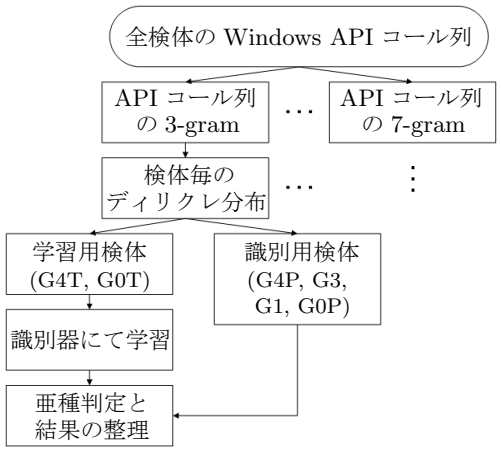

図 2: C-SVM による実験の概要図

を集計したものを実験の最終結果とした．

以上の実験手順を纏めると，図 2 の通りとなる．

3.4 実験結果

本節では，実験の結果を示し，それに関する考察を記述する．なお，3.1 節の説明にあるように，C-SVM における学習用データと識別用データの区分は無作為に行ったため，実験結果の公正性は確保されている．

3.4.1 検体の識別結果

提案手法及び C-SVM を用いた検体の識別実験の結果を，表 5 と表 6 にそれぞれ示す．

“T-P” (亜種を亜種として正しく検出したケース) の割合は，提案手法では 7-gram，C-SVM では 3-gram, 4-gram, 7-gram が最も高いことから，提案手法における適切なウィンドウサイズ N は 7 であることが分かる．また，両手法の “T-P” の割合に差は見られないため，提案手法の亜種識別精度は C-SVM

表 5: 提案手法による識別結果

	分類	検体数	割合
3-gram	T-P	220	17.50%
	F-P	83	6.60%
	T-N	940	74.78%
	F-N	13	1.03%
4-gram	T-P	225	17.90%
	F-P	70	5.57%
	T-N	953	75.82%
	F-N	8	0.64%
5-gram	T-P	225	17.90%
	F-P	80	6.36%
	T-N	943	75.02%
	F-N	8	0.64%
6-gram	T-P	226	17.98%
	F-P	81	6.44%
	T-N	942	74.94%
	F-N	7	0.56%
7-gram	T-P	228	18.14%
	F-P	90	7.16%
	T-N	933	74.22%
	F-N	5	0.40%

表 6: C-SVM による識別結果

	分類	検体数	割合
3-gram	T-P	228	18.14%
	F-P	111	8.84%
	T-N	912	72.61%
	F-N	5	0.40%
4-gram	T-P	228	18.14%
	F-P	77	6.13%
	T-N	946	75.26%
	F-N	5	0.40%
5-gram	T-P	225	17.90%
	F-P	75	5.97%
	T-N	948	75.42%
	F-N	8	0.64%
6-gram	T-P	227	18.06%
	F-P	78	6.21%
	T-N	945	75.18%
	F-N	6	0.48%
7-gram	T-P	228	18.14%
	F-P	81	6.44%
	T-N	942	74.94%
	F-N	5	0.40%

と同等である．ところで，実験にて入力した亜種検体 (G4P 及び G3 属する検体) の総数が 233 個であるのに対して，実験では 228 個の検体を誤りなく検出しているため，提案手法及び C-SVM による亜種の検知率は $\frac{228}{233} * 100 \simeq 97.85\%$と求められる．

しかし，その一方で，“T-P”の割合が高い 4 区分の N-gram における “F-P”(非亜種を亜種として誤検出したケース) の割合は C-SVM の 4-gram が最も低い．つまり，亜種の識別精度が高く，かつ誤検出の少ない手法は 4-gram を用いた C-SVM であることが分かる．なお，“F-N”(亜種を非亜種と判断して検知漏れを起こしたケース) の割合については両手法で同一である．

3.4.2 識別に要した時間

次に，亜種の識別に要した時間を表 7 に示す．本表の結果は 3 回分の測定値の平均を取ったものである．また，公正を期するため，双方の実験で共通する動作 (N-gram やディリクレ分布の生成時間) やファイルの I/O に要した時間は測定対象から除外した．

C-SVM の測定結果には grid.py を含むものと含まないものを併記したが，C-SVM におけるペナルティC は学習用データに依存するため，新規の学習用データを入力する場合には前者，既存のデータを流用する場合には後者の計測結果が，実際に検体を識別する際の所要時間に相当する．

ここで，双方の計測結果を比較すると，ウィンドウサイズの大小に関わらず，提案手法は grid.py を併用しない場合の C-SVM と比べて約 2 倍高速である．特に grid.py を併用した際の C-SVM と比較すると，提案手法はおよそ 100 倍のスピードで演算を終えている．これは，C-SVM では必須であるデータの学習プロセス及びパラメータの設定プロセスを，提案手法では必要としないためであると推測される．よって，提案手法は速度の面で C-SVM よりも優れていると言える．

3.4.3 最大メモリ消費量

最後に，亜種の識別時にプログラムが消費したメモリの最大値を表 8 に示す．本表の結果もまた，3

表 7: 識別に要した時間

		計測時間
提案手法	3-gram	4.57s
	4-gram	10.33s
	5-gram	16.98s
	6-gram	23.44s
	7-gram	30.02s
C-SVM + grid.py	3-gram	417.89s
	4-gram	914.71s
	5-gram	1750.97s
	6-gram	3053.62s
	7-gram	3942.43s
C-SVM	3-gram	9.98s
	4-gram	21.82s
	5-gram	36.73s
	6-gram	50.11s
	7-gram	64.74s

回分の測定値の平均を取ったものである．なお，本測定は，亜種識別プログラムが生成したプロセスのメモリ使用量を，コマンドプロンプトの tasklist コマンドによって 1 秒ごとに測定し，その最大値を取ることで実施した．

双方の計測結果を比較すると，各ウィンドウサイズにおいて，提案手法の最大メモリ消費量が C-SVM の約 1/9 に抑えられていることが分かる．これは，C-SVM が処理過程において，全学習用データと識別用データの 1 つをメモリ上に展開する必要があるのに対して，提案手法では基準検体と識別対象検体の 2 つを展開すれば処理を継続出来るためであると考えられる．従って，C-SVM と比べて，提案手法はメモリの利用効率が優れていると言える．

4 関連研究

提案手法のように，検体の振る舞いの記録情報より生成した N-gram に基づくマルウェア分類を行った例としては Santos ら [11] による実験が挙げられる．当該実験では k-近傍法によって分布を比較しているため，SVM と同様に学習用データとしての基準検体を複数用意する必要がある．一方で，提案手法は基準検体を 1 つ用意すれば利用可能なため，検体解析者の分析方針に手掛かりを与える手法としては，

表 8: 最大メモリ消費量

		計測時間
提案手法	3-gram	129.13MB
	4-gram	290.98MB
	5-gram	434.70MB
	6-gram	570.54MB
	7-gram	688.28MB
C-SVM	3-gram	1003.26MB
	4-gram	2702.52MB
	5-gram	4115.26MB
	6-gram	5381.70MB
	7-gram	5939.35MB

提案手法の方が利便性の面で優位である．また，提案手法における亜種識別率が 97.85%であるのに対して，Santos らの手法における識別率は 91.25%に留まっている．それぞれの実験で異なる検体群を用いていることから，識別精度を単純に比較することはできないが，提案手法では k-近傍法に基づく手法に比べて高い精度で亜種を識別できる可能性があると推測している．

堀内ら [15] は，基準検体及び識別対象検体の動的解析結果における分布間のハミング距離を求めることで，基準検体の亜種を検知している．しかし，本研究では 1 つの AV ソフトウェアによる検体の検査結果を利用しており，提案手法では複数の AV ソフトウェアによる検査結果を利用する．従って，提案手法の方が AV ソフトウェアの誤検知による影響を受けにくい．

Forrest ら [8] らや島本ら [16] による研究では，プロセスのシステムコール列を用いてプログラムの特徴を検査しているものの，プログラムの異常検知を目的としている点が本稿の提案手法と異なる．

Rieck ら [10] の手法は，本稿における C-SVM を用いた亜種の識別実験に類似している．但し，当該手法では時系列順に並べた検体の API コール列について N-gram を利用せず，API コール列をそのまま用いている点が本稿の実験と異なる．

5 まとめと今後の課題

本稿では，基準検体と分析対象検体による API コール列に関する分布の類似度を KL 情報量によっ

て示す手法を提示し，実験を通じて当該手法を評価した．その結果，提案手法は C-SVM と比べて誤判定が多いものの，亜種の識別能力は互角で，計算時間やメモリの利用効率は C-SVM に比べて良好であることが判明した．

今後の課題は主に 2 つ存在する。第一の課題は，提案手法による亜種の誤判定を極小化することである．第二の課題は，本手法に広範な汎用性を持たせることである．本手法は，基準検体に対する亜種の識別を想定したものであり，現時点における用途は必然的に限定される．今後は，検体を全自動でクラスタリングする等，より幅広い用途に適するように手法を拡張したい．

参考文献

[1] AV-Comparatives. http://www.av-comparatives.org/.

[2] AV-TEST. http://www.av-test.org/en/home/.

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

[4] IKVM.NET. http://www.ikvm.net/.

[5] Libsvm. http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[6] Libsvm.net. http://code.google.com/p/libsvm-net/.

[7] McAfee 脅威レポート：2013 年第 1 半期. http://www.mcafee.com/japan/security/report/download.asp?no=78.

[8] Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff. A Sense of Self for Unix Processes. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pp. 120–128, 1996.

[9] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A Practical Guide to Support Vector Classification. Technical report, Department of Computer Science, National Taiwan University, 2003.

[10] Konrad Rieck, Thorsten Holz, Carsten Willems, and Patrick Düssel. Learning and Classification of Malware Behavior. In *Fifth Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 08)*, Vol. 5137, pp. 108–125, 2008.

[11] Igor Santos, Yoseba K. Penya, Jaime Devesa, and Pablo Garcia Bringas. N-grams-based File Signatures for Malware Detection. In *ICEIS (2)*, pp. 317–320, 2009.

[12] Trend Micro Incorporated. TSPY_SPYEYE.EXEI. http://about-threats.trendmicro.com/us/malware/TSPY_SPYEYE.EXEI.

[13] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[14] 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～. *MWS2013*, 2013.

[15] 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.

[16] 島本大輔, 大山恵弘, 米澤明憲. System Service 監視による Windows 向け異常検知システム機構. 情報処理学会論文誌：コンピューティングシステム, Vol. 47, No. 12, pp. 420–429, 2006.

[17] 岩上由高. 2012 年中堅・中小企業における「クライアント PC セキュリティ」の利用実態とユーザ評価. http://www.norkresearch.co.jp/pdf/2012itapp_pcs_rel.pdf.