実行ファイルに含まれる文字列の学習に基づくマルウェア検出方法

戸部 和洋 †　　森 達哉 †‡　　千葉 大紀 ††　　下田 晃弘 †　　後藤 滋樹 †

† 早稲田大学 基幹理工学研究科 情報理工学専攻
169-8555 東京都新宿区大久保 3-4-1
{tobe, tatsuya, shimo, goto}@goto.info.waseda.ac.jp

†† 早稲田大学 基幹理工学部 情報理工学科
169-8555 東京都新宿区大久保 3-4-1
chiba@goto.info.waseda.ac.jp

‡NTT サービスインテグレーション基盤研究所
180-8585 東京都武蔵野市緑町 3-9-11
mori.tatsuya@lab.ntt.co.jp

あらまし 本研究ではマルウェア実行ファイルの構造やパッキングの有無に依存しない軽量かつ高速なマルウェア検出方法を提案する．中心となるアイディアは実行ファイル集合に含まれる文字列情報を抽出し，得られた情報から統計的特徴ベクトルを作成して教師付き機械学習を適用することである．合計で 1511 の通常ファイルと 1449 のマルウェアを用い，既存方式との性能比較を行った結果，ファイルの先頭部分にあたる非常に少ない情報を利用することで高速かつ高精度な検出が達成可能であることがわかった．

Detecting malware by learning character strings in executable files

Kazuhiro Tobe†　　Tatsuya Mori†‡　　Daiki Chiba††　　Akihiro Shimoda†
Shigeki Goto†

†Graduate School of Fundamental Science and Engineering, Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555 JAPAN
{tobe, tatsuya, shimo, goto}@goto.info.waseda.ac.jp

††School of Fundamental Science and Engineering, Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555 JAPAN
chiba@goto.info.waseda.ac.jp

‡NTT Service Integration Laboratories, NTT Corporation
3-9-11 Midori-cho, Musashino-shi, Tokyo 180-8585 JAPAN
mori.tatsuya@lab.ntt.co.jp

Abstract This paper develops a simple and efficient technique that can detect malware without having knowledge about the structure of malware executable files nor existence of packing. The key idea is to make use of character strings included in executable files and construct statistical feature vector for each file. We then apply supervised machine learning method. Through the analysis of 1,511 and 1,449 of benign and malware files, we validate that our proposed method can achieve fast and accurate malware detection.

1　はじめに

ウイルス，トロイの木馬，ワーム，スパイウェアなどの悪意あるソフトウェア—マルウェア —による被害が拡大・深刻化している．マルウェアがもたらす被害は迷惑メールやフィッシングメールの大量送信，サーバへの不正な大量アクセス，個人情報の抽出などの脅威にとどまらず，航空機の障害監視システムへの感染が大きな要因となり，航空機墜落事故による 154 名もの死亡事故まで発生している [1]．このようなマルウェアによる被害を未然に防ぐためにはマルウェア本体，あるいはマルウェア本体を送受信している通信を検出する技術が必要となる．

マルウェアを検出するための解析手法は，動的解析と静的解析の二つのアプローチに大別される．動的解析はマルウェアをコントロールされた環境において実行し，その挙動を調べることによって解析を行う手法である．静的解析は，マルウェアを実行せずに，マルウェアを構成するバイナリ実行ファイルを逆アセンブルするなどして解析を行う手法である．いずれの解析手段も分析に要するコストが高いという課題がある．今日観測されるユニークなマルウェアの種類はおよそ 1000 万に達する事が報告がされており [2]，今後もこの傾向が続くことが予想される．したがって，マルウェアの被害を未然に防ぐためにはより軽量なマルウェア検出手段が必要である．

本研究は以上の点に着目し，マルウェアあるいはマルウェアを送信する通信を高速に検出するための軽量な手段を提案する．中心となるアイディアはマルウェアおよび通常の実行ファイルに含まれる文字列から統計的特徴量を抽出し，機械学習を適用することによって両者を識別することである．

本研究の貢献は既存の動的解析および静的解析を置き換えるものではなく，それらの分析をサポートする軽量な手段を提供することである．本研究の提案手法は軽量であるため，ネットワークレイヤで実行可能であることが期待できる．多数のネットワークで検出された情報を用いることにより，動的解析および静的解析を用いて詳細に解析すべきマルウェアを高速に同定できる．ネットワークレベルでの分析に対応することにより，マルウェアを送信する通信をリアルタイムに遮断することも可能である．

2　関連研究

ファイルとしての特徴を元にマルウェアを検出する研究として，[3–5] などがある．文献 [3] はマルウェアに含まれる可読な文字列を分析し，バギングとサポートベクターマシン (SVM) を組み合わせることによってマルウェアを検出する手法を提案している．この研究は我々のアプローチに最も近いものであるが，以下に示す点で我々の研究との相違点がある．すなわち，(1) 文献 [3] は取得した文字列統計から特徴ベクトルを構成する方法が明確に示されていない．(2) 文献 [3] はファイル全体を用いたオフラインでの検出方法を提案しているのに対し，我々の研究ではより高速な検出を実現するためにファイルの先頭 n バイトのみの情報を用いたオンラインでの検出を目指している．

文献 [4] では実行ファイルを構成する固定長のバイナリの集合を用い，情報エントロピーによる特徴づけを行っている．すなわち，バイナリをいくつかの区間に分割し，それぞれの区間ではさらに小さなビット列のパターンの出現頻度を算出する．この出現頻度を元に区間ごとに情報エントロピーを計算し，ファイル全体での統計から通常のファイルと暗号化あるいはパッキングされたマルウェアを識別する．本アプローチのアイディアは，マルウェアの多くはパッキングあるいは暗号化されているため高いエントロピーを持つ性質を利用することにある．本研究ではこの手法と提案手法の性能比較を行う (詳細は 3.4 節を参照)．

文献 [5] は Portable Executable (PE) フォーマットのプログラムを分析し，そのファイルがパッキングされているか否かをパターン認識の手法を適用することによって判定する手法を提案している．パッキングされたファイルをアンパックする技術の一つであるユニバーサル・アンパッカーは，暗号化アルゴリズムに関する事前知識を使うことなくオリジナルのプログラムを

抽出可能な技術である．しかしながらユニバーサル・アンパッカーの計算コストは大変高い．したがって，軽量な手法によって事前にファイルがパッキングされているか否かを知ることにより，ユニバーサル・アンパッカーの効率的な利用が実現出来る [5]．中心となるアイディアはコード・データセクションの名前，書き込み実行可能セクションの数，コードとデータのエントロピーを特徴とし，ナイーブベイズ，決定木，K 最近傍法などを適用することである．文献 [4] の手法は実行ファイルの構造を利用するため，オンラインでの検出には適さないという性質がある．

3　マルウェア検出方法

本章は我々が提案するマルウェアの検出方法について詳細を述べる．はじめに通常及びマルウェアの両方を含む実行ファイル集合に含まれる印字可能な文字列を抽出する．次に抽出された文字列の出現パターンを用いて統計的な特徴ベクトルを構成する．最後に特徴ベクトルに対して教師あり学習の一つであるサポートベクターマシンを適用することにより，ある実行ファイルが通常，あるいはマルウェアのいずれかであるかを高速かつ高精度に識別する．本手法はマルウェアの構造には直接的に依存しないため，未知のマルウェア（特に亜種に対して）もロバストな判定が可能であると期待できる．

このアプローチにおいて鍵となるのは統計的特徴量の構成方法である．後に示すように抽出された文字列のすべてを単純に採用する場合，次元数が非常に増大するため，機械学習の訓練に必要となる計算量のコストが高くなる．この制限を緩和するためには，(1) 文字列情報のダウンサイジング，および (2) ファイル全体から一部の情報をサンプリングするアプローチが考えられる．特に (2) のサンプリングにおいてファイルの先頭部のみ利用すればオンラインでの通信遮断等に適用可能であると期待できる．また，実行ファイルの先頭部分にはマジックナンバーや識別子等各種の特徴的なヘッダが入るため，より識別に有効な情報が含まれていると予想される．そのため，先頭部分を利用することは他の部分をサンプルするよりも有効であることが予想される．

3.1　文字列情報の抽出

本節では実行ファイルに含まれる文字列情報の抽出と辞書を利用したダウンサイジング方法について述べる．文献 [3] と同様に，実行ファイルに対して GNU Binary Utilities [6] に含まれる strings プログラムを適用することにより，印字可能な文字列の集合を抽出する．ここで得られる文字列の多くは印字が可能ではあるものの，意味を成さないものが大多数である．また次章で示すように単純に抽出した文字列の数は膨大になるため，限られた時間内で機械学習の訓練を完了するためには何らかの方法で数を削減する必要がある．

そこで得られた印字可能な文字列集合のうち，辞書に含まれる単語のみを抽出することによって可読な文字列集合を抽出する．このヒューリスティックにより，特徴空間の大幅なダウンサイジングが可能になる．また，Windows API の関数名などバイナリ実行ファイルに含まれる可読な文字列集合はファイルの特徴として有効な情報であると考えられる．

本研究では可読な文字列を抽出する辞書コーパスとして英語版 wikipedia の全ページのアーカイブ [7] に存在した単語集合および標準的な英単語辞書の和集合を採用した．多言語の辞書対応は今後の課題である．また，`getURLbyID` のような複数単語の合成で，各々の単語の最初の一文字を大文字で表現しているような文字列に対しては，辞書のチェックを行う前に `get`，`URL`，`by`，`ID` のように分かち書きを実行するヒューリスティックを適用した．

さらに本研究では実行ファイルオブジェクトの先頭 n バイトから文字列を抽出したケースについて性能を比較する．

3.2　統計的特徴量の抽出

本研究では前節の方法によって抽出した文字列に対して下記の 2 つの方法で特徴ベクトルを

定義し，性能を比較する．

tf-idf　term frequency-inverse document frequency (tf-idf) はテキストマイニングの分野において単語の重み付けに使われる手法であり，以下のように定義される．

$$\phi_{ij} \quad = \quad \frac{n_{ij}}{\sum_k n_{kj}}(\ln M - \ln m_i)$$

ここで n_{ij} は文字列 i がファイル j において出現した回数である．M はすべての実行ファイルの数であり，m_i は文字列 i を含む実行ファイルの数である．実行ファイル j の特徴ベクトルを $\{\phi_{1j}, \phi_{2j}, \ldots\}$ と定義する．

出現の有無　tf-idf よりも簡便な統計的特徴量として，実行ファイル j において文字列 i が存在したら 1，それ以外の場合は 0 をとるバイナリの変数 δ_{ij} を導入し，実行ファイル j の特徴ベクトルを $\{\delta_{1j}, \delta_{2j}, \ldots\}$ と定義する．

3.3　SVM のセットアップ

サポートベクターマシン (SVM) は教師あり機械学習法の一つであり，優れた識別機能を有することで知られている．SVM の特徴は入力データを高次元に写像した上でデータを識別する超平面を構築することであり，写像された高次元空間において線形分離のアプローチを試みる．教師データを用いた訓練では超平面による識別境界と訓練データ間の距離–マージンを最大化するようパラメタを最適化する．SVM ではソフトマージンへの緩和を採用している．このため，マージン最大化条件および境界面によって生じる誤りに関するトレードオフを制御するパラメタ C を経験的に求める必要がある．同様に，高次元空間に写像する基底関数 $\phi(x)$ の内積として定義されるカーネル関数についてもパラメタを経験的に決定する必要がある．本研究では代表的なカーネル関数であるガウスカーネル $e^{-\gamma||\mathbf{x}-\mathbf{y}||}$ を採用する．

本研究では SVM の実装として広く利用されている LIBSVM [8] を用いた．パラメータチューニングおよび精度評価は文献 [9] の手法を適用した．すなわち，特徴量に対して線形なスケーリングを施した後，制御パラメタ C，およびカーネル関数のパラメータ γ のグリッド探索を行い，最適な境界面を与えるパラメータ集合を得る．得られたパラメータを用いて 5 分割の交差検定法を適用し，正解率の平均値を算出する．

3.4　エントロピー法

本研究では文献 [4] の方法をエントロピー法と呼び，検出精度の比較対象として採用する．基本的なアイディアはパッキングされたマルウェアを構成するビット列は暗号化によってより乱雑なパターンとなるため，エントロピーが高くなるという性質を利用することである．はじめに実行ファイルを 256 バイトからなるバイト列に分割する．それぞれのバイト列 j において 1 バイトからなるビット列のパターン (00–FF のいずれかの値) の出現頻度を算出し，情報エントロピー $H_j = -\sum_{i=1}^{n} p_j(i) \log_2 p_j(i)$ を計算する．ここに $p_j(i)$ はバイト列 j においてあるビット列パターンが出現した経験確率であり，$n = 2^8$ である．次にバイト列毎に計算した情報エントロピーの集合 $H_1, H_2, \ldots$ を用いて通常ファイルとマルウェアを識別する．

文献 [4] では最大値あるいは平均値が通常のファイルと暗号化あるいはパッキングされている実行ファイルで明らかな差が出るため，適切に定めた閾値によって両者を高い精度で識別可能であることを示している．しかしながら本研究で対象とする実行ファイルを使った予備実験では単純な最大値あるいは平均値ではきれいに分離しないことが判明した．この原因は本研究で分析の対象としたマルウェアが必ずしもパッキングされているとは限らないことにあると予想される．

そこで本研究ではエントロピーから得られる統計値として最大値，最小値，平均値，標準偏差を採用し，それらの値の組み合わせを特徴ベクトルとする．この特徴を用いて、SVM によって通常のファイルとマルウェアを識別する．

4 性能評価

4.1 実行ファイルの収集

提案手法の有効性を示すため，マルウェア検体 1449 個と，通常の実行ファイル 1511 個を用いて性能評価を行った．これらの検体はすべて異なるハッシュ値を有することを確認している．

マルウェアの入手方法は次の 3 通りである．

- MWS 2010 Datasets (CCC DATAset 2008, 2009, 2010, D3M DATAset 2010) [10] に含まれるマルウェア検体.
- ある複数のネットワークに設置した低対話型ハニーポット Nepenthes [11] で収集したマルウェア検体.
- スパムメールに添付されたマルウェア検体.

通常の実行ファイルの入手方法は次の 2 通りである．

- 初期状態の Windows XP SP3 上に存在した実行ファイル
- ソフトウェアライブラリ Vector [12] で収集した実行ファイル[1]

4.2 検出精度の比較

図 1 に各手法の精度を比較した結果を示す．縦軸の accuracy は 3.3 節で述べた手法を用いて得られた正答率である (5 分割交差検定の平均値)．例えば accuracy が 95%は，マルウェアあるいは通常のいずれかに正しく識別できたファイルの割合が 5 回交差検定法を試行した結果，平均で 95%となることを意味する．横軸は各手法をそれぞれ実行ファイルの先頭 n バイトに対して適用する実験を行った結果である．n の値は 512, 1024, 2048，4096，8192 バイトとした．ただし，後に示すように n が大きい場合に特徴ベクトルの次元数が非常に高くなるケースがある．そのようなケースではグリッドサーチによる SVM のパラメタ最適化が一定期間内に収束しなかった為，本研究では割愛している．以下で示すように次元数を上げたからといって必ずしも精度が向上するわけではないため，実質的な問題はない．

図中の naive_str，dict_words はそれぞれ単純な strings コマンドの結果を採用したケース，および辞書を用いてダウンサイジングしたケースである．括弧内の tf-idf，bin は 3.2 節で示した統計的特徴量の構成方法の違いを示す．entropy_2d および entropy_4d はそれぞれエントロピー法において最大値，平均値および最大値，最小値，平均値，標準偏差を特徴ベクトルとしたケースである．

図より以下のことが読み取れる．全般的に提案手法はエントロピー法よりも精度が良い．統計的特徴量は tf-idf を採用したほうが精度が高い．tf-idf を採用する場合，単純な文字列よりも辞書を適用したほうが精度が向上する．tf-idf を採用する場合，より小さい n の方が精度が高い傾向がある．この最後の結果はオンラインでの検出を行う場合に非常に有利な性質である．小さい n でも高精度な結果が得られるのは，ファイルの先頭部分により重要な情報が集中しているためと予想される．

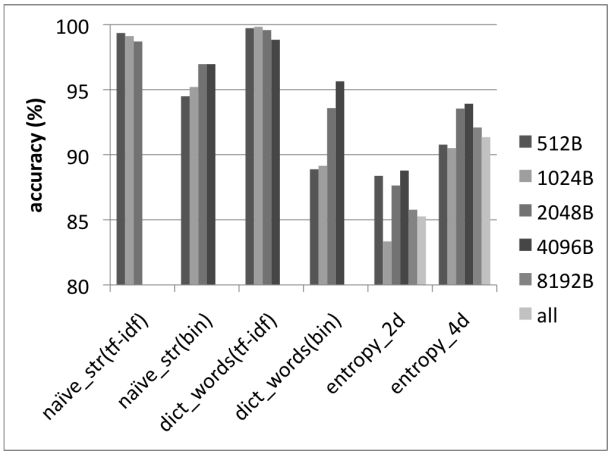

図 1: 各手法の精度比較.

4.3 実行コストの比較

本研究では特徴ベクトルの次元数 d を実行コストの評価尺度とする．これは，カーネル関数

[1]Windows 向けソフトウェアのダウンロードランキング上位のものを入手した．ただし，この中にはインストーラやダウンローダも含まれるが，それらはそのまま評価実験に用いた．一方，アーカイブされているソフトウェア（拡張子が.zip や.lzh のファイル）は解凍して，その中に入っている実行ファイルを評価実験に用いた．

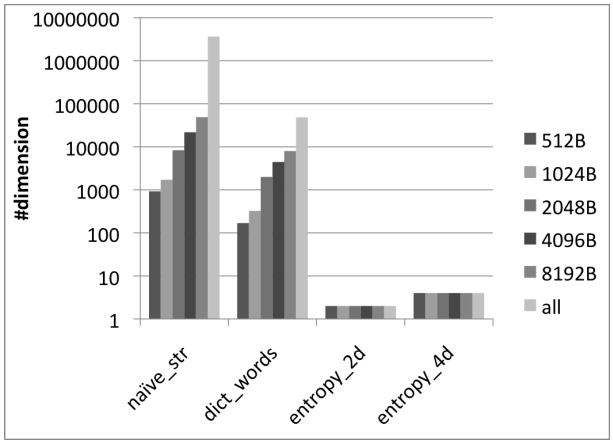

図 2: 各手法のコスト比較.

の計算量のオーダーが $O(d)$ であるため，d の増加に対して SVM の計算量が線形に増えるためである．entropy_2d, entropy_4d の次元数は n によらずそれぞれ 2，4 である．一方で，文字列を特徴ベクトルとして用いる場合はファイル内で検出される文字列の数は可変であるため，特徴ベクトルの生成方法に応じて次元数が大きく変動する．

図 2 に各手法の特徴ベクトルの次元数を比較した結果を示す．naive_str および dict_words については対象とする先頭バイトのサイズ n に対して指数関数的に次元数が増大することがわかる．特に単純な文字列を使うケースの方がより急激に次元数が増大することが読み取れる．前節の結果とあわせると n が小さいほど精度が向上し，かつ計算コストが削減されるという結論を得る．

5 まとめ

本研究は，マルウェアあるいはマルウェアを送信する通信を高速に検出するための軽量な手段を提案した．中心となるアイディアはマルウェアおよび通常ファイルの実行ファイルに含まれる文字列集合から統計的特徴量を抽出し，教師あり機械学習を適用することによって両者を識別することである．実データを用いた性能評価の結果，提案方法がきわめて高い検出精度を達成することを示した．さらに実行ファイルの先頭部分のみを利用した場合に高い精度を達成可能であるため，オンラインでの識別にも適用可能である．文字列に加えてバイト列の n-gram などを利用することによるさらなる精度の向上，通常・マルウェアの二値識別からマルウェアの種別を複数クラスに識別する拡張，機械学習に必要な計算量の大幅な短縮が今後の課題である．

謝辞 本研究の初期段階において貴重なご助言を頂いた NTT 情報流通プラットフォーム研究所の岩村誠氏に感謝致します．

参考文献

[1] L. Meredith, “Malware implicated in fatal Spanair plane crash,” TechNewsDaily, Aug. 20, 2010. http://www.msnbc.msn.com/id/38790670/ns/technology_and_science-security/.

[2] F. Paget. “Malware at Midyear: a Summary,” http://www.avertlabs.com/research/blog/index.php/2010/07/07/malware-at-midyear-a-summary/

[3] Y. Ye, L. Chen, D. Wang, T. Li, and Q. Jiang, “SBMDS: an interpretable string based malware detection system,” Journal in Computer Virology, Vol. 5, No. 4. pp. 283–293. 2009.

[4] R. Lyda and J. Hamrock, “Using Entropy Analysis to Find Encrypted and Packed Malware,” Security & Privacy, IEEE, Volume 5, Issue 2, 2007 pp. 40–45.

[5] R. Perdisci, A. Lanzi, and W. Lee, “Classification of packed executables for accurate computer virus detection,” Pattern Recognition Letters, Volume 29, Issue 14, 2008, pp. 1941–1946.

[6] GNU Binutils, http://www.gnu.org/software/binutils/.

[7] Wikimedia Downloads, http://download.wikimedia.org/.

[8] C. C. Chang and C. J. Lin, “LIBSVM : a library for support vector machines,” 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[9] C. W. Hsu, C. C. Chang, C. J. Lin, “A practical guide to support vector classification,” http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[10] 畑田 充弘, 中津留 勇, 秋山 満昭, 三輪 信介. マルウェア対策のための研究用データセット 〜 MWS 2010 Datasets 〜. マルウェア対策研究人材育成ワークショップ 2010.

[11] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling. “The Nepenthes Platform: An Efficient Approach to Collect Malware,” In Proceedings of the 9 th International Symposium on Recent Advances in Intrusion Detection (RAID), pp. 165–184. Springer. 2006.

[12] Vector： ソフトライブラリ& PC ショップ - 国内最大級のフリーソフト ダウンロードサイト http://www.vector.co.jp/