6X-07

実行ファイルの階層的な構造を考慮したマルウェア検知手法

堀江 健太† 彌冨 仁†

†法政大学理工学部応用情報工学科

概 要

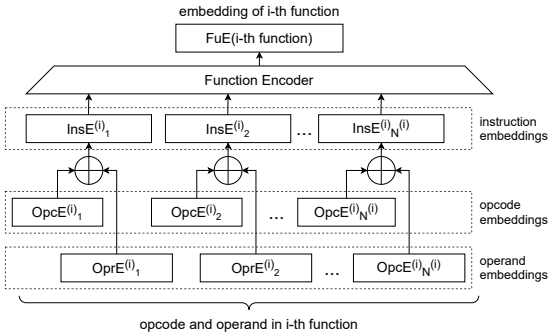
近年，機械学習モデルによるマルウェア検知手法が数多く提案されているが，検知の根拠を提示できる手法は限られている．本研究では実行ファイルを階層的に解析，統合することで根拠提示が期待できる新しいマルウェア検知モデルを提案する．本報告では検知の根拠提示の可能性について評価を行う前段階として，実データを用いたマルウェア検知能の検証を行い，提案手法の有効性を確認した．

1 はじめに

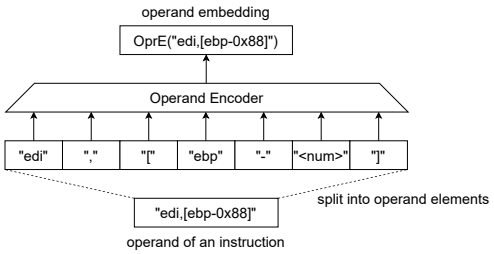
近年マルウェアを高速かつ高精度に検知する機械学習手法が数多く提案されているが，検知の根拠を提示できる手法は限られている．検体の主な解析方法である表層解析，動的解析，静的解析のうち，静的解析は検体を安全かつ網羅的に調べることができるというメリットがあり，Jeon ら [1] は検体を実行した際に実行される命令のオペコードを抽出して解析し，検知に加え命令レベルでの根拠提示も可能な手法を提案した．しかし，実行される命令列のみ解析するため解析の網羅性に課題が残されている．

本研究では，図 1 に示す関数内の命令，検体中の関数，検体全体と階層的に解析していく bottom-up 手法を提案する．本手法は検体そのものから診断結果までを end-to-end で学習するモデルであり，モデルの勾配情報に基づく隠れ層の解析により関数レベルの根拠提示を行えることが期待できる．全ての関数を解析対象とするため検体の網羅的な解析も期待できる．

本報告では，提案法の有効性を検証するために，32bit の exe および dll ファイルからなる学習用データおよびテストデータを構築し，検体を固定長の断片に分割して convolutional neural networks (CNN) を適用していくことにより，根拠提示可能ではないものの検体全体を解析し高い精度でマルウェア検知を行える手法である MalConv [2] および提案法のテストデータでの性能を測定した．



(a) 関数のベクトル表現の獲得方法



(b) オペランドのベクトル表現

図 1: 提案手法の全体像

2 手法

2.1 手法の概要

検体が入力されると，2.2 節に述べる方法で検体内に存在する N_f 個の各関数が検出され，関数ごとのベクトル表現を求める処理が行われる．まず i 番目 ($i = 1, 2, \cdots, N_\mathrm{f}$) の関数内の j 番目 ($j = 1, 2, \cdots, N^{(i)}$) の命令に対応するオペコードおよびオペランドがそれぞれ 2.3 節に述べる方法で固定長のベクトル表現 $\mathrm{OpcE}_j^{(i)}$ および $\mathrm{OprE}_j^{(i)}$ に変換される．次に，図 1a に示すように各命令に対応する $\mathrm{OpcE}_j^{(i)}$ および $\mathrm{OprE}_j^{(i)}$ を結合することにより，固定長のベクトル表現 $\mathrm{InsE}_j^{(i)}$ が獲得される．それらを recurrent neural networks (RNN) の一種である QRNN [3] および全結合層から構成される `Function Encoder` に入力することで，i 番目の関数に対応する固定長の関数のベクトル表現である $\mathrm{FuE}^{(i)}$ に変換する．最後に，N_f 個からなる $\mathrm{FuE}^{(i)}$ 列を QRNN および全結合層から構成される `File Encoder` に入力することで検体の固定長のベクトル表現である `FiE` に変換し，最

Malware detection considering the hierarchical structure of executable files

Kenta HORIE†, Hitoshi IYATOMI†

†Applied Informatics, Science and Engineering, Hosei University, 184-8584, Koganei, Tokyo, Japan

{kenta.horie.9x@stu., iyatomi@}hosei.ac.jp

後に FiE を全結合層からなる File Classifier に入力することで検体の識別を行う．本手法の，命令の解析，関数の解析，検体全体の解析という階層的構成の各層のパラメータは end-to-end で学習されるため，以後の階層ごとの解析可能性が期待できる．

2.2 関数の定義および抽出方法

本報告では，検体を objdump[1] により逆アセンブルした結果において，push ebp と mov ebp,esp の 2 つの命令から始まり ret で終了するまでに存在する一連の命令の系列を関数とし，ジャンプ命令などによる制御フローは無視した．

2.3 OpcE および OprE の獲得方法

OpcE は，学習に伴い要素が更新される固定長のベクトル表現を与えることで獲得する．OprE は，図 1b に示すように，はじめにオペランドを記号，演算子，数値，レジスタ名のオペランド要素に分解して各要素に OpcE と同様に固定長のベクトル表現を与え，それらを QRNN および全結合層から構成される Operand Encoder に入力することにより獲得する．

2.4 モデルの具体的な構成

Operand Encoder は，オペランド要素の 8 次元ベクトル表現の可変長系列を双方向 QRNN に入力し，順方向と逆方向の最終状態を結合したベクトルをさらに 2 層の全結合層に通すことで 16 次元の OprE を出力する．共に 16 次元ベクトルである OpcE と OprE を結合した InsE は 32 次元ベクトルである．Function Encoder および File Encoder の構成は Operand Encoder と同様であり，前者の出力は 64 次元の FuE，後者の出力は 128 次元の FiE である．File Classifier は，1 層の全結合層および Softmax 関数から構成され，検体がマルウェアである確率，検体がグッドウェアである確率の 2 要素からなるベクトルを出力する．

3 実験

データセットの構築　実験には A と B の 2 つのデータセットを用意した．データセット A は，フリーウェアの配布サイト[2] から得られたグッドウェア，MalShare[3] から得られたマルウェアで構成されたデータセット[4] のうち，objdump で解析可能なグッドウェアとマルウェアそれぞれ 739 体を抽出して構成した．データセット B は，Windows10 が搭載された PC の C:\Program Files (x86) 以下から収集したグッドウェア，VXHeaven [5] から収集したマルウェアのうち，objdump で解析可能なグッドウェアとマルウェアそれぞれ 330 体を抽出して構成した．

表 1: 各手法のテストデータでの識別結果

手法	正解率	適合率	再現率	F1 値
MalConv [2]	**0.788**	**0.910**	**0.639**	**0.751**
提案法	0.727	0.861	0.542	0.665

実験方法　提案法，および 1 章で述べた MalConv の 2 つの手法について，Windows10 由来の検体が多く含まれるデータセット B よりも多様性があると考えられ，検体数も多いデータセット A を用いて学習し，データセット B を用いて正解率，適合率，再現率，F1 値を測定し識別性能を検証した．

4 結果と考察

提案法および MalConv の識別能を表 1 に示す．提案法は一定のマルウェア検知能を持つが，MalConv には劣っていた．この原因として，提案法は MalConv と異なり検体の関数以外の部分が考慮されていないことに加え，関数の制御フローを無視しているため関数の特徴を十分に捉えられていないこと，長い関数列や命令列を QRNN に入力した場合に系列の前方の情報が失われていることが考えられる．一方で前述の通り，提案法は階層構造を持つ end-to-end 構成のため，勾配情報に基づきどの部位の悪性コードが含まれているかなどの根拠の可視化などが期待できると考えられる．今後の精度向上を図るとともに評価を行いたい．

5 おわりに

本報告では，関数内の命令，検体中の関数，検体全体と階層的に解析していく bottom-up 手法を提案し，マルウェア検知能を検証した．今後モデルを改良し検知能を向上させることとともに根拠提示の可能性についても評価を行う．

参考文献

[1] S. Jeon and J. Moon, “Malware-detection method with a convolutional recurrent neural network using opcode sequences,” *Information Sciences*, vol. 535, pp. 1 – 15, 2020.

[2] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. Nicholas, “Malware detection by eating a whole exe,” *Proc. of AAAI Workshops*, pp. 268–276, 2018.

[3] J. Bradbury, S. Merity, C. Xiong, and R. Socher, “Quasi-recurrent neural networks,” *Proc. of ICLR*, 2017.

[1] https://sourceware.org/binutils/docs-2.26/binutils/objdump.html

[2] https://www.portablefreeware.com/

[3] https://malshare.com/

[4] https://www.reddit.com/r/datasets/comments/exhy38/malware_and_benign_windows_pe_cuckoo_reports/

[5] https://archive.org/details/vxheavens-2010-05-18