多段パックされたマルウェアからのコード取得

中村 徳昭† 森井 昌克† 伊沢 亮一‡ 井上 大介‡ 中尾 康二‡

†神戸大学大学院工学研究科
657-850 神戸市灘区六甲台町 1-1
n.nakamura@stu.kobe-u.ac.jp
mmorii@kobe-u.ac.jp

‡独立行政法人情報通信研究機構
184-0015 東京都小金井市貫井北町４丁目２-１
{isawa},{dai},{ko-nakao}@nict.go.jp

あらまし 近年のマルウェアは，解析を困難にするためにパッキングが施されている場合が多く，解析を行う際は，パッキングを解除する必要がある．パッキングの手法には様々なものがあり，中でも，多段パックと呼ばれる手法はコード展開の処理が複雑になっており，オリジナルコードの取得が難しいことで知られている．本研究では，多段パックに対して汎用的にオリジナルコードを取得することを目的とし，代表的な多段パッカー (tElock,PESpin,yoda's Crypter) のアルゴリズムを調査したところ，パッキング解除に関して共通の動作があることが分かった．具体的には，パッキング解除時にはメモリ各所に対して操作を行なうが，最終的には実行コードが必ず text 部に展開される．本稿ではこの点に着目し text 部を監視することでオリジナルコードを取得する方法の詳細を与える．

Code Capture from Self-Modifying Malwares

Noriaki Nakamura† Masakatu Morii† Ryoichi Isawa‡ Daisuke Inoue‡
Koji Nakao‡

†Graduate School of Engineering, Kobe University
1-1 Rokkoudai-cho, Nada-ku, Kobe-shi, Hyogo,657-8501 JAPAN
n.nakamura@stu.kobe-u.ac.jp
mmorii@kobe-u.ac.jp

‡National Institute of Information and Communications Technology
Nukuikitamachi4-2-1, koganei-shi, Tokyo, 184-0015 JAPAN
{isawa},{dai},{ko-nakao}@nict.go.jp

Abstract Most malwares are packed or encrypted. In order to analyze such malwares, we have to unpack them and extract the original codes from them. There are many techniques of packing. It is difficult to extract original codes from malwares packed with multi-stage packers because their decryption algorithms is complicated. In this paper, we present our preliminary efforts toward generic binary unpacking against the three multi-stage packers, which are tElock, PESpin, and yoda's Crypter. We find out that the thee packers has a common part of their algorithms. The packers finally reveal the original code of a malware to a specific section named '.text section.' That is, we set a break point in the section and can get the original codes of any malwares packed with the three packers.

1　はじめに

マルウェアの持つ機能の全体像を把握するには逆アセンブラやデバッガなどを利用して静的解析するのが有効である．しかし，近年のマルウェアのほとんどは静的解析を妨害するためのパッカーと呼ばれるツールにより，難読化，解析妨害機能の具備が施されている．これらのマルウェアを解析する為には，解析妨害機能を回避しつつ難読化を解き，マルウェアの本来のプログラムコードであるオリジナルコードを取りだす作業，アンパックが必要となる．アンパックには複数の方法があるが，通常は既知のパッカーに対応した専用のアンパッカーを用いなければならない．数多くのパッカーに対応した RL!Depacker[1] といった強力なアンパッカーも存在するが，実行可能な状態までアンパックできないケースも多い．

本研究では，多段パックに対して汎用的にオリジナルコードを取得することを目的とし，代表的な多段パッカーである tElock[2]，PESpin[3]，yoda's Crypter[4] のアルゴリズムを調査したところ，パッキング解除に関して共通の動作があることが分かった．これらの多段パックは，パッキング解除時にはメモリ各所に対して操作を行なうが，最終的には実行コードが必ず text 部分に展開する．本稿ではこの点に着目し text 部分を監視することで実行コード全体を取得し，オリジナルコードを取得する方法の詳細を与える．パック以前のオリジナルコードを取得できているかどうかはメモリアドレス中のオペコードを比較することで評価を行う．しかしながら，評価の結果，我々の方法では Import Redirection により書き換えられたインポートアドレステーブル（IAT）を再構築することはできなかった．Import Redirection とは IAT の値をヒープに作成したダミーコードのアドレスに書き換える技術である．IAT が取得できないため，そのマルウェアが使用する Windows API を得ることができない．

IAT が取得できないオリジナルコードは実行可能な状態ではないが，それを用いてマルウェアの解析支援に役立てることは可能である．取得できたオリジナルコードの有用性を示すため，これらの部分的なコードを用いた類似度判定を行う．類似度判定では，コードを「ベーシックブロック」と呼ばれる一定の基準に基づいて分割し，各検体のベーシックブロックの一致率を調べることにより算出する．

本提案手法の評価のため，マルウェア 492 検体と 3 種類の多段パッカーを用いて，2 つの実験を行った．実験 1 では，マルウェア 168 検体に対してそれぞれ 3 種類の多段パッカーでパッキングし，提案手法を用いた結果，全てのパッカーに対して元々のプログラムのオリジナルコードのベーシックブロックを 9 割以上取得することができた．また実験 2 では MWS2012 データセット [5] のうち 324 検体をランダムに選び，取得したコードをベーシックブロックに分割，各検体同士のブロック一致率を調べることにより，マルウェア検体間の類似度を算出することに成功した．

本稿の構成は以下の通りである．まず，第 2 章で既存研究では多段パックに対して完全なオリジナルコードを取得するのが困難であることを説明し，第 3 章で多段パックに対して有効と考えられる手法を述べる．第 4 章で提案手法を用いて行った実験結果を示し，最後に第 5 章でまとめる．

2　関連研究

近年，コードに着目した類似度算出手法が数多く提案されており，コードから得られる制御フローをグラフ等に変換し構造の比較を行う手法 [6] や，バイナリコードを処理ブロックに分割し，ブロック単位で比較を行う手法 [7] などがある．これらの手法では，パッカーによる圧縮・暗号化処理に対しては既に何らかの方法でアンパックが成功しており，解析対象となるコード本来の動作を示すオリジナルコードが取得済みであることが前提となっている．

そこで，ステルスデバッガや DBI (Dynamic Binary Instrumentation) により，書き込みや実行といったメモリアクセスを監視することでオリジナルコードを推定する手法 [8, 9, 10] が数多く検討されている．また，DBI による実行命令トレースをシグネチャマッチングすることに

よってパッカーの種類を特定する手法 [11] も提案されている．しかし，DBI を用いた手法では実際にマルウェア検体自体を実行する必要があるため，tElock や PESpin といった，自己を書き変える機能を持つ強力なパッカーに対しては実行命令を全く取得できないという課題がある．加えて，マルウェアが解析時に必ず特徴的な挙動を示すという保障がない点や，マルウェアによる様々な解析環境検知機能により，重要な挙動が隠蔽される可能性がある点も問題として挙げられる．文献 [12] では，このような多段パッカーに対して，オリジナルエントリポイントを発見する手法が提案されているが，IAT の再構築を行うことはできない，また，DBI エンジンを用いているため，実行命令トレースを検知されるリスクが高いといった問題がある．

3 多段パックされたマルウェアに対するコード取得

本章では，多段パックされたマルウェアに対してオリジナルコードを取得する手法を述べる．まず 3.1 節で多段パックに用いられている「Import Redirection」という手法について説明し，次に 3.2 節で多段パックからコードセクションをメモリダンプする手順を述べる．

3.1 Import Redirection

IAT のリビルド時や逆アセンブラでの解析時に，インポートしている API をわかりにくくする手法が Import Redirection と呼ばれている手法である．通常，IAT には図 1 に示すようにインポートした API のアドレスが格納されている．したがって，IAT を参照すれば，どのような API をインポートしているかを把握することができる．しかし，Import Rediredtion が行われている場合，IAT は API のエントリポイントではなく，API を呼びだすための関数のアドレスに書き換えられており，そのアドレスを見ただけではどのような API が呼び出されるかわからないといった問題がある．

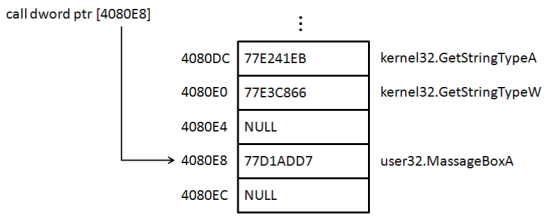

図 1: IAT を用いた通常の API 呼び出し

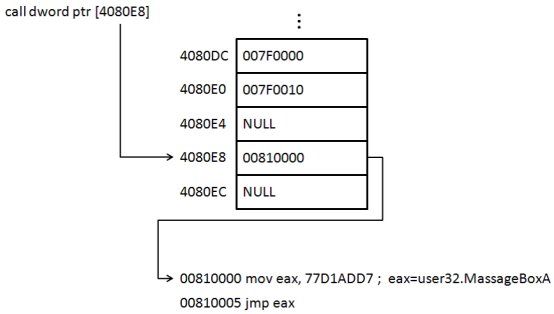

図 2: Import Redirection の例

また，API を呼びだすための関数内では，単純に call 命令だけで API に制御を移すのではなく，いったん汎用レジスタに API のアドレスを格納してから jmp 命令で関数を呼び出す，などのやりかたをとる場合があり，書き換えの手法は一様ではない．

図 2 の例では mov 命令を使った後にジャンプ（jmp）する形で API 呼び出しを行っているが，push 命令で API のアドレスをスタックに積み，ret 命令で API のエントリポイントに飛ぶ方法など，多段パッカーの種類によって様々なやり方が考えられる．このため，多段パックされたマルウェアに対して IAT のリビルドを完全に行うことは困難である．

3.2 多段パックからのオリジナルコード取得

ここでは，多段パックされた EXE ファイルから OEP を検出し，オリジナルコードをメモリダンプする手法を示す．以下の 3 つ Step を行うことにより，オリジナルコードを取得する．

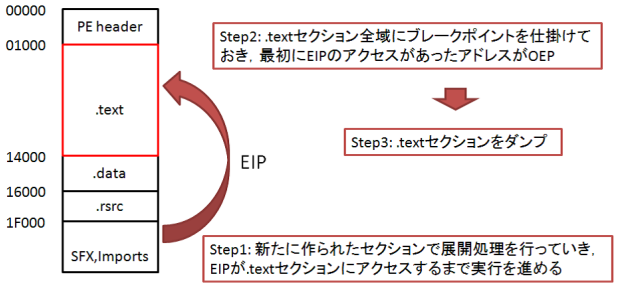

図 3: 多段パックからのオリジナルコード取得

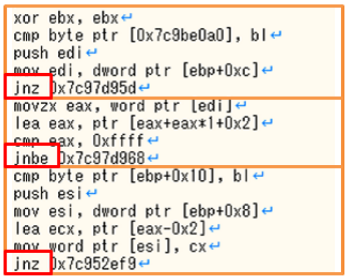

図 4: ベーシックブロック分割

オリジナルコード取得手順

Step1 展開ルーチン処理の実行

Step2 コードセクション上での実行検出

Step3 コードセクションのメモリダンプ

3 つの Step の実行の様子を図 3 に示す．Step1 では，多段パックによって新たに作られたセクションにエントリポイントが移されているので，このセクション内で 1 ステップずつ展開処理を行う．展開ルーチンによる処理が終了後，コードセクションにオリジナルコードの書き込みが行われるので，書き込みが終了した時点でコードセクション全域にメモリアクセスブレークポイントを仕掛ける．Step2 では，インストラクションポインタ（EIP）の制御がコードセクションへと移る瞬間をブレークポイントによって捉える．ここで停止した地点がオリジナルエントリポイントとなる．Step3 では，Step2 で停止した地点をエントリポイントとしてコードセクションのメモリダンプを行い，展開されたオリジナルコードを取得する．

なお今回の実験では，デバッガには OllyDbg v1.10[13] を，メモリダンプには OllyDbg のプラグインである OllyDump[14] を用いた．

4 オリジナルコードのオペコード部分を用いたマルウェア間の類似度判定

我々の多段パックに対するオリジナルコードの取得する方法では IAT の再構築ができない．本章では IAT の値が正しくないオリジナルコードに対応した類似度判定法を提案する．4.1 節でオリジナルコードを扱う単位となる「ベーシックブロック」の概念について説明し，4.2 節で類似度判定の手法について説明する．

4.1 ベーシックブロックの導入

本研究で取得できたオリジナルコードは 1 検体につき数万行で構成される検体がほとんどである．特に多段パックされたマルウェアから取得したコードは，IAT の呼び出し部分がパッカーによって書き換えられているため，本来のオリジナルコードとは異なる部分も存在する．また，このコードが特定の命令長で一致しても同じ機能を有しているとは言い難い．そこで，図 4 に示すように，このオペコード列をベーシックブロック単位で分割して比較を行う．ベーシックブロックとは,「分岐命令・分岐先命令で区切られた連続した命令列」と定義する．このブロック単位で比較を行う理由としては，分岐命令にまでの連続した命令が行われることで，マルウェアの一つの機能に関する一連の動作を行っていると考えられるからである．以後，本手法の類似度判定ではベーシックブロック単位でコードを分割して扱う．

4.2 ベーシックブロックを用いた類似度判定

オリジナルコードのオペコード部分をベーシックブロック単位で用いて，マルウェア間の類似度判定を行う手法を説明する．マルウェア間の類似度はベーシックブロックの一致率によって算出する．マルウェア A に対するマルウェア B の類似度を $sim(A,B)$ とし，$sim(A,B)$ を以下の式 (1) で定義する．

$$sim(A,B) = \frac{C_A \cap C_B}{C_A} \quad (1)$$

ここで，C_A と C_B はマルウェア A とマルウェア B の全コードブロックをそれぞれ示している．$sim(A,B)$ の値が大きいほど，マルウェア A はマルウェア B のコード部分を多く含むことになり，マルウェア B の機能を多く有していると考えられる．

5 評価実験

本章では提案手法の評価のために行った 2 つの実験手法およびその結果について述べる．

5.1 実験 1：多段パックからのオリジナルコード取得

実験 1 では，多段パックされたマルウェア検体から提案手法によって元のオリジナルコードのオペコード部分を取得できるかどうかを評価する．そのため，168 検体のマルウェア検体に対して 3 種類の多段パッカーを用い，すべての組み合わせでパッキングを行い，パックする以前のオリジナルコードとオペコード部分で比較する．なお，コード比較は提案手法の 3 章で説明したベーシックブロック単位で行った．図 5 に各多段パックに対するオリジナルコード取得率を示す．横軸は検体番号，縦軸は元検体と比較したときのオリジナルコード取得率である．図 5 に示す通り，ほぼすべてのパッカーと検体の組み合わせでパック以前のオリジナルコードのベーシックブロックを取得できていることがわかる．しかしながら，数検体についてはパック以前のオリジナルコードを十分に取得できなかった．特に全くコードを取得できなかったパターンも存在したが，このようなマルウェアは通常環境においても EXE ファイルとして全く動作しなかった．これはパッキングによってマルウェア自体の実行可能な EXE 構造を壊してしまったことが原因であると考えられる．

5.2 実験 2：ベーシックブロックを利用したマルウェアの類似度判定

実験 2 では，MWS2012 データセットからランダムに 324 検体選び，パッキングの種類および有無に関わらず，提案手法によってオリジナルコードが取得できることを検証する．また，取得した各オリジナルコードをベーシックブロックに分割し，検体間のブロック一致率を総当たりで算出することで，類似度判定が行えるかどうかを評価する．

実験の結果，すべての検体に対してオリジナルコード候補を取得することができた．また，これらをベーシックブロックに分割して総当たりで類似度判定を行ったところ，実験 2 で用いた 324 検体の Symantec[15] によるウイルス科名はすべて「W32/RAHack」であったのに対して，取得できたオリジナルコードにはいくつかのバリエーションがあることがわかった．これらのブロック一致率を基準にしてクラスタリングを行うことができた．検体同士が同一クラスタに含まれるための条件を式 (2) に示し，クラスタリング結果を表 1 に示す．

$$sim(X,Y) \geq T_d \quad or \quad sim(Y,X) \geq T_d \quad (2)$$

ここで，X と Y はそれぞれ同一クラスタに含まれる任意の検体を示し，T_d は閾値を示す．

6 まとめ

本稿では，多段パックされたマルウェアから IAT をリビルドする以前の全ての実行コードを取得し，オペコード部分をマルウェア間で比較することで類似度判定を行う手法を提案した．

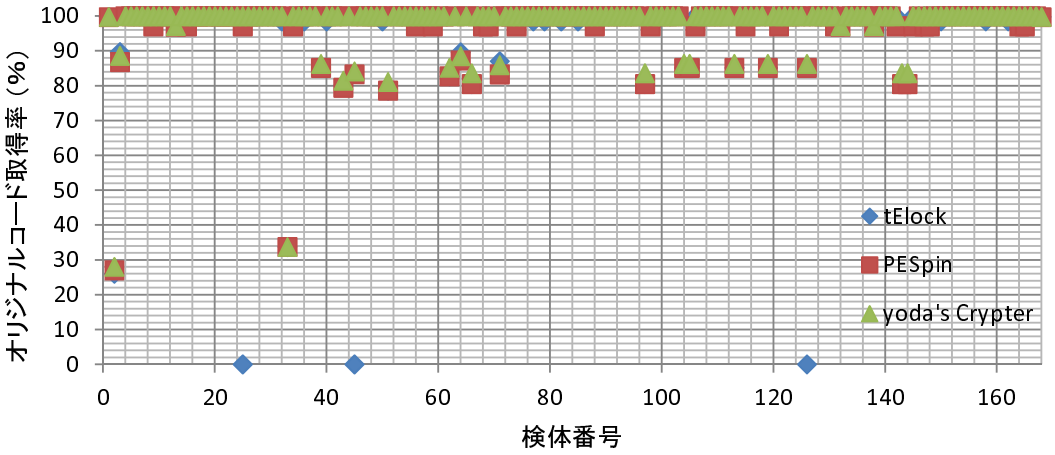

図 5: 多段パックに対するオリジナルコード取得率

表 1: 閾値 $T_d = 0.9$ でクラスタリングした結果

クラスタ No	検体数
1	85
2	85
3	83
4	26
5	26
6	6
7	4
8	4
9	3
10	1
11	1

提案手法の評価方法として，マルウェア 168 検体に対して多段パックを施し，パックする以前のオリジナルコードが取得できるかを検証した．また MWS2012 データセットからランダムに選んだ 324 検体に対して，パッカーの影響を受けることなくオリジナルコードを取得することができた．これらのコードをブロックに分割して比較することで，大まかに定義されているウイルス科名より詳細に類似度判定を行うことができた．我々のオリジナルコードの取得方法は IAT の再構築には対応できなかった．しかしながら，取得したオリジナルコードによりマルウェアの類似度判定が可能であることを示した．

謝辞

有益な御討論を頂いた（独）情報通信研究機構衛藤将史氏をはじめとする（独）情報通信研究機構ネットワークセキュリティ研究所サイバーセキュリティ研究室関係各位に感謝する．

参考文献

[1] RL!Depacker, http://www.reversinglabs.com/forum/index.php?topic=11.0

[2] tElock 0.98, http://www.telock.com-about.com/

[3] PESpin v1.33, http://pespin.w.interia.pl/

[4] Yoda's Crypter, http://www.yodas-crypter.com-about.com/

[5] MWS2012 実行委員会, 研究用データセット, http://www.iwsec.org/mws/2012/about.html#datasets

[6] 岩本一樹, 和田克己, “コンピュータウイルスのコード静的解析による特徴抽出と分類について, ” 信学技報, vol. 107, no. 397, ISEC2007-127, pp. 107-113, (2007).

[7] Marius Gheorghescu, “AN AUTOMATED VIRUS CLASSIFICATION SYSTEM”

[8] 織井達憲, 吉岡克成, 四方順司, 松本 勉, 金 亨燦, 井上大介, 中尾康二, “パッキングされたマルウェアの類似度算出手法とその評価,” 信学技報, ICSS, 109(285), pp. 7-12, (2009).

[9] H. C. Kim, D. Inoue, M. Eto, Y. Takagi, and K. Nakao, “Toward Generic Unpacking Techniques for Malware Analysis with Quantification of Code Revelation, ” JWIS, (2009).

[10] 岩村誠, 伊藤光恭, 村岡洋一, “コンパイラ出力コードの尤度に基づくアンパッキング手法, ” MWS2008, pp.103-108, (2008).

[11] 川古谷裕平, 岩村誠, 針生剛男, “実行命令トレースに基づく動的パッカー特定手法, ” Computer Security Symposium 2011

[12] Piotr Bania, “Generic Unpacking of Self-modifying, Aggressive, Packed Binary Programs. ”

[13] OllyDbg, `http://www.ollydbg.de/`

[14] OllyDump, `http://www.openrce.org/downloads/details/108/OllyDump`

[15] Symantec, `http://www.symantec.com/index.jsp`