特集
マルウェア

{研究用データセット：マルウェア検体編}

機械語命令列の類似性に基づく自動マルウェア分類システム

岩村　誠　伊藤光恭（NTT 情報流通プラットフォーム研究所）

急増するマルウェア

昨今の多くのマルウェアは，C や C++ といった高級言語で開発され，頻繁にバグ改修や機能追加といった改良が繰り返されている．またアンチウイルスソフトによる検出から逃れるために，パッカーと呼ばれる一種の圧縮ツールにより，同じ機能を持ちながらも外観が異なる形式で大量に生産されている．このような開発サイクルにより，近年のマルウェア数は増加の一途を辿り，それらを網羅的に解析することは難しくなってきている．

一方でこうした状況を打開すべく，マルウェアの分類技術の研究が進んでいる．本稿では，これまでに提案されてきたマルウェア分類技術について概説した後，マルウェアの分類を自動化するための要件と，我々が提案した自動マルウェア分類システムについて紹介する．続いて，我々のシステムを用いて実際に収集されたマルウェアの分類結果を示し，最後に今後の展望について述べる．

マルウェア分類技術

マルウェア分類技術には大きく分けて，挙動に着目する手法[1]と，プログラムコードに着目する手法[2), 3)]がある．

挙動に着目する手法では，まずネットワークやファイルシステム等のシステムリソースへのアクセスを監視できる環境を用意する．次に，その環境上でマルウェアを動作させ，得られたアクセス履歴を抽出し，このアクセス履歴の特徴に基づきマルウェアを分類する．この手法の利点は，マルウェアの挙動を把握できる実行環境さえ用意できればよく，人手によるリバースエンジニアリング等の作業を必要としないことにある．一方この手法は，挙動を網羅的に抽出することが困難なマルウェアに関しては，適切な分類結果を得られないといった問題を抱えている．たとえば，攻撃者の指令なしには動作しないボットや，ある時刻にしか動作しないマルウェアでは，マルウェアを特徴付ける挙動を観測することが難しい．

こうした挙動に基づく分類手法の問題点を解決するために，マルウェアのプログラムコードから抽出した特徴に基づきマルウェア間の類似度を算出する手法も提案されている．プログラムコードを分類の材料とすることで，マルウェアが潜在的に備える機能を踏まえた分類が可能になる．この方法に関する従来研究は，プログラムコードの特徴として何に着目しているかで整理できる．

N-gram に基づくアプローチでは，まずマルウェアの逆アセンブル結果からオペレーションコード列を抽出し，連続する N 個のオペレーションコード列（N-gram）がそれぞれ何回出現したかを特徴ベクトルとする．そして，この特徴ベクトルのコサイン距離をマルウェア間の類似度とする．また，N-gram の代わりに N-perm と呼ばれる順序関係を考慮しない特徴を利用する方法も提案されている．N-gram/N-perm を利用した手法の特長は，ベクトル間のコサイン距離としてマルウェアの類似度が定義されるため，非常に高速に処理可能なことである．

ほかにもプログラムのベーシックブロックに着目した手法も提案されている．ベーシックブロックとは，分岐命令・分岐先命令を含まない連続した機械語命令列である．まずマルウェアの逆アセンブル結果から，プログラムコードをベーシックブロックに分割する．そしてベーシックブロック単位で編集距離を算出し，それをマルウェアの非類似度とする．加えて，ベーシックブロック単位での並べ替えに対応するために，転置インデックスまたはブルームフィルタを用いる手法も提案されている．この手法もまた高速に類似度を算出することが可能である．

さらに，プログラムのコールツリーを特徴とし，類似性を求める手法も存在する．コールツリーは，ソースコード上のプログラム構造が反映されるため，コンパイラの変更に強く実際に備える機能に則した分類結果が得やすいとされている．

また我々も LCS（Longest Common Subsequence：最長一致部分列）に基づくマルウェアの類似度算出手法を提案している[8)]．この手法では，機械語命令からオペランド情報を取り除いた縮約命令を定義し，その縮約命令

列に関して LCS を求める．そして求まった LCS の長さに基づき類似度を定義する．この手法の特長は，命令単位でのきめ細かい変化を類似度に反映できることである．ベーシックブロック単位での処理と比較すると速度の低下が懸念されるが，LCS 算出アルゴリズムのビットベクトル化により，この問題の解決を図っている．また，逆アセンブル結果(機械語命令とデータを識別した結果)さえあれば，類似度を算出できることや，機械語命令単位で変更があった個所を抽出できるといった利点もある．

このように，マルウェアのプログラムコードの特徴に焦点を当てたマルウェア分類技術は，マルウェアが潜在的に備える機能に基づく分類が可能である．次章では，こうした技術に基づき自動的にマルウェアを分類するシステムについて述べる．

自動分類システム

昨今の多くのマルウェアはパッカーにより難読化されているため，その機能を表す機械語命令列を取得するにはアンパッキングと呼ばれる工程が必要となる．また，アンパッキングの結果得られるのは機械語命令とデータが混在するバイト列であるため，プログラムコードに基づく分類を行うためには，さらに機械語命令とデータを識別する逆アセンブルも必要となる．

本章では，まずアンパッキングおよび逆アセンブルを自動化する手法について説明する．その後，アンパッキング手法，逆アセンブル手法，最長一致部分列を利用した類似度算出手法を組み合わせた自動マルウェア分類システムについて述べる．

●アンパッキング手法

パッカーの一種であるランタイムパッカーは，実行ファイルを入力として，その実行可能形式を保ったまま難読化を施すことができる．つまり，既存の開発環境とは独立して利用可能であり，マルウェアに多く適用されている．また，こうしたランタイムパッカーは非常に多くの種類があるため，従来から個々のランタイムパッカーに依存しない汎用的なアンパッキング手法の研究が進められている．その中でも Saffron[4] は，OS のページフォールトハンドラを独自の処理に置き換えることで，全メモリアクセスをフックし，動的に生成されるコード，つまりアンパッキングされたマルウェアのオリジナルコードの抽出に成功している．Saffron はすべての仮想アドレスに対応する PTE（Page Table Entry）のスーパーバイザービットをセットすることで，すべてのメモリアクセスの際にページフォルトを発生させ，実行しようとしている領域が動的に生成された領域かどうかを判断する．その後，PTE のキャッシュとなる TLB（Translation Look-aside Buffer）上のスーパーバイザービットのみをクリアすることで，対象プログラムの処理を続行させる．

しかしながら，一般的に仮想マシンにおける TLB は実機のそれを忠実に再現しているとは限らない．実際，広くマルウェア解析に利用される仮想マシンである VMware では，命令用 TLB が特権モードを跨ぐ際にフラッシュされるとの報告もある．こういった状況では，ページフォルト後にスーパーバイザービットをクリアした状態を作り出せないため，カーネルモードとユーザモードを行き来する無限ループに陥ってしまう．一方，Saffron のような実際にマルウェアを動作させることでアンパッキングを行う手法では，マルウェアに感染した解析機を元のクリーンな状態に戻す工程を要し，その解決に仮想マシンは非常に有用である．そこで文献 5）では，TLB の実装に依存せずにメモリへの書き込みおよび実行を監視する手法を提案している．Saffron では書き込み監視時，実行監視時ともにスーパーバイザービットを設定していたが，文献 5）の手法では，書き込み監視時にはライタブルビット，実行監視時には XDbit をクリアすることで，Saffron と同様の機能を実現しつつも，仮想マシン上でのアンパッキングを可能にしている．

●逆アセンブル手法

通常，マルウェアのデバッグシンボル等は入手できない．このためマルウェアを精度よく逆アセンブルすることも，研究課題の 1 つとなる．従来の逆アセンブル手法は主に Linear sweep 法と Recursive traversal 法[6]，もしくはこれらを組み合わせた手法に分けられる．Linear sweep 法は，プログラムを先頭から逆アセンブルし，命令として解釈できない部分はデータと解釈，次バイトから逆アセンブルを行い，この作業を終端まで繰り返す．この手法では，命令として解釈可能であればデータ部も命令部として逆アセンブルしてしまう．また可変命令長の場合は，一度命令の先頭を見誤ると，連鎖的に別命令として逆アセンブルし，真の命令列とは異なる命令列が多数出力される．一方，Recursive traversal 法は，エントリポイントや頻出命令列とマッチする個所を命令列の先頭とし，命令として解釈できない部分までを逆アセンブルする．また，逆アセンブルの過程で分岐命令が現れたときは，その分岐先を新たな命令列の先頭とし，再帰的に逆アセンブルする．これにより，Linear sweep 法のようにデータ部分を誤って命令として解釈することはなくなる．しかしながら，分岐先が動的に決まる場合は，分岐先が命令列として解釈されなくなる．こうした分岐先が動的に決まるケースは，C++ における仮想関数テーブルをはじめとして，その他のコンパイラによる出力でも

散見される．

文献 7）ではこうした従来技術の欠点を踏まえ，分岐命令等に頼らず確率的に命令かデータかを判断する逆アセンブル手法を提案している．この手法では隠れマルコフモデル（HMM）を用いることで，コンパイラ出力コードの特徴をモデル化する．簡単な例としては，機械語命令を 1 命令出力する状態とデータ 1 バイトを出力する状態，そして 4 つの状態遷移（各状態間の遷移と自己遷移）を持つモデルが考えられる．また学習データとして，バイト列と命令かデータかを判断できる逆アセンブル結果を事前に用意しておく．この学習データはマルウェアの開発によく利用される開発環境（Visual C++ や Delphi など）により，事前にさまざまなソフトウェアをコンパイルし作成してもよいし，また実際のマルウェアをいくつか解析し作成してもよい．そして各状態における出力確率および状態遷移確率を，この学習データにより学習させてモデルパラメータを決定し，Viterbi アルゴリズムを用いることで，未知のバイト列に対する最ももっともらしい逆アセンブル結果を算出することが可能になる．

●自動マルウェア分類システム

我々は，前述の TLB の実装に依存しないアンパッキング手法，確率的逆アセンブル手法および LCS に基づくマルウェアの類似度算出手法を組み合わせ，自動マルウェア分類システムを構築した [8]．

まず本システムに対してマルウェアが与えられると，アンパッキング部において，パッキングされているマルウェアからオリジナルのプログラムコードを含むメモリダンプが出力される．アンパッキング部により出力されたメモリダンプは逆アセンブル部において逆アセンブルされ，機械語命令列が得られる．類似度算出部は，各マルウェアの機械語命令列を元に類似度を算出し，マルウェアの全組合せに関する類似度行列を作成する．これにより，入力されたマルウェアを系統別に分類することが可能となる．

分類結果と考察

まず，これまでに我々がハニーポットで収集した 3232 種類（SHA1 ハッシュ値が重複するものは取り除いてある）のマルウェア検体を分類した．

群平均法を用いた階層的クラスタリングの結果，図 -1 のデンドログラムが得られた．図 -1 の円周上には，分類対象となった 3232 種類の検体が並んでおり，検体（クラスタ）をつなぐ弧の半径が，検体（クラスタ）間の類似度となっている．つまり，円周近くで繋がっているほど類似しており，中心でつながっているほど，類似度が低いことを意味している．

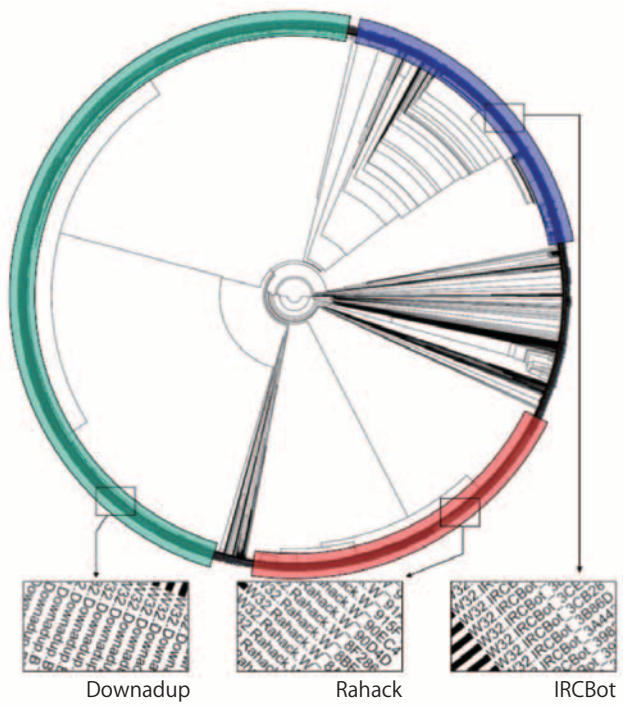

図 -1　3232 検体の分類結果

ここで類似度 90% を閾値とすると，141 のクラスタに分割された．このうち含まれる検体数が多い上位 2 つのクラスタが，全検体数のそれぞれ約 49%（1576 検体，図 -1 緑色部分），約 21%（675 検体，図 -1 赤色部分）を占めることが分かった．さらに，アンチウイルスソフトによる検出名や静的解析の結果，各ファミリは Downadup（図 -1 緑色部分），Rahack（図 -1 赤色部分）であることが分かった．

また Downadup のクラスタを，類似度 91% を閾値としてさらに分解してみると，3 つのクラスタとなった．これらは，アンチウイルスソフトの検出名等を参考にすると，Downadup.A/B/B++ の 3 種類であることが分かった．Rahack に関しては，類似度 94% を閾値とすると Rahack.W と Rahack.H の 2 種類のクラスタに分けられた．Rahack.W の多様性はなく 1 種類のプログラムコードである一方，Rahack.H に関しては，類似度 98% を閾値として分類してみると，4 種類に分けられた．特にそのうち 1 種類に関して，アンチウイルスソフトでは半数以上が Backdoor.Trojan として検出されていることが分かった．

また，図 -1 青色部分のクラスタは階段状になっている．これは，他の 2 つのファミリと比べてプログラムコードの多様性に富んでいることを示している．このクラスタに含まれるマルウェアをいくつか静的解析した結果，IRC 関連の処理が含まれるボットであることが分かった．ただし，アンチウイルスソフトの検出名を見てみると，IRCBot のほかにも Virut と識別されているものが多く含

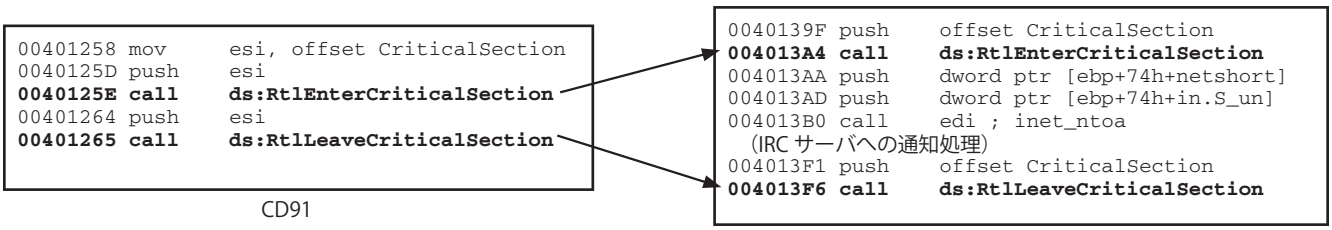

図 -2　7190 検体と CD91 検体の比較

まれていた．Virut のプログラムコードは IRC 関連の処理と比べると小さい．一方，今回の実験では，複数のメモリダンプが得られた際には，最も命令数が多い逆アセンブル結果を用いて分類した．このため，アンチウイルスの検出名とは異なる分類結果が得られたと考えられる．

ここで，CCC Dataset 2009 のマルウェア検体 2 種類の比較結果についても，興味深い結果が得られたので紹介する．これらの検体の SHA1 ハッシュ値先頭 2 バイトは，7190，CD91 であり，データセット提供側から何らかの関連があると提示されている．これらの検体の類似度は約 29% であり，その一致個所には TCP 139/445 番を用いた通信ロジックが含まれていた．また 7190 の検体には，他ホストへのスキャン活動後に IRC サーバへのスキャン結果通知用のメッセージを作成するロジックが含まれていた．このスキャン結果に関するロジックは，EnterCriticalSection と LeaveCriticalSection に挟まれており，実際に IRC サーバとの通信を行うスレッドとの排他制御が行われている．

一方で，CD91 の検体にも 7190 の検体とまったく同じ他ホストへのスキャン活動を行うロジックが含まれていたが，IRC サーバへの通知にかかわるロジックは存在しなかった（図 -2）．しかし，排他制御すべき処理が存在しないにもかかわらず，EnterCriticalSection と LeaveCriticalSection を連続して呼び出す処理は存在していた．

これはあくまで推測であるが，7190 の検体から IRC 関連の機能を取り除き，感染活動に特化させたマルウェアが CD91 検体であり，クリティカルセクション関連 API の呼び出しは，その際に削除し損じた処理であると考えられる．実際，CD91 検体にだけは autorun.inf を悪用した比較的新しい感染活動に関するロジックが含まれており，こうした状況からも，7190 検体の後に CD91 検体が開発されたと考えられる．

今後の展望

マルウェアをプログラムコードに基づき分類しようとすると，アンパッキングや逆アセンブル，さらにはコールツリーの再構築やベーシックブロック単位の分割など，さまざまなリバースエンジニアリングを要する．しかしながら，マルウェアのリバースエンジニアリングの多くは，一部の専門家による手作業によりなされているのも事実であろう．日々増加するマルウェアへの対策のためにも，こうしたリバースエンジニアリングの自動化に関する研究のさらなる促進を願い，本稿の締めくくりとしたい．

参考文献

1) Bailey, M., et al. : Automated Classification and Analysis of Internet Malware, In International Symposium on Recent Advances in Intrusion Detection (2007).
2) Karim, Md. E., et al. : Malware Phylogeny Generation Using Permutations of Code, European Research Journal of Computer Virology, Vol.1, No.1-2, pp.13-23 (Nov. 2005).
3) Gheorghescu, M. : An Automated Virus Classification System, In Virus Bulletin Conference, pp.294-299 (Oct. 2005).
4) Quist, D., et al. : Covert Debugging : Circumventing Software Armoring, Blackhat USA Las Vegas, NV. (2007).
5) 岩村　誠：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).
6) Schwarz, B., et al. : Disassembly of Executable Code Revisited, In Proceedings of the Ninth Working Conference on Reverse Engineering, IEEE Computer Society, pp.45-54 (2002).
7) 岩村　誠：隠れマルコフモデルに基づく新規逆アセンブル手法，電子情報通信学会総合大会，基礎・境界，p.181 (2008).
8) 岩村　誠：機械語命令列の類似性に基づく自動マルウェア分類システム，MWS2009, pp.799-804 (2009).

（平成 21 年 12 月 30 日受付）

岩村　誠（正会員）
iwamura.makoto@lab.ntt.co.jp
2002 年早稲田大学大学院理工学研究科修士課程修了．同年日本電信電話(株)入社．現在，NTT 情報流通プラットフォーム研究所勤務．

伊藤光恭（正会員）
itoh.mitsutaka@lab.ntt.co.jp
1984 年早稲田大学大学院理工学研究科修士課程修了．同年日本電信電話公社入社．現在，NTT 情報流通プラットフォーム研究所勤務．