実行毎の挙動の差異に基づくマルウェア検知手法の提案

笠間 貴弘†‡　　吉岡 克成†　　井上 大介‡　　松本 勉†

†横浜国立大学
240-8501　神奈川県横浜市保土ヶ谷区常盤台 79-7
kasama@mlab.jks.ynu.ac.jp, {yoshioka, tsutomu}@ynu.ac.jp

‡独立行政法人 情報通信研究機構
184-8795　東京都小金井市貫井北町 4-2-1
{kasama, dai}@nict.go.jp

あらまし 近年のマルウェアの中には，解析や検知への耐性強化を目的とし，実行時の挙動を変動させるものが存在する．例えば，自身のコピーをファイルシステム内に複製する際のファイル名を実行毎に異なる名前にしたり，C&Cサーバ等の攻撃者サーバへアクセスする際に，ブラックリストによるブロックを回避するため，内蔵した多数のドメイン名の中からランダムに接続先を決定したりするマルウェアが存在する．このような動作は正規のプログラムには必要のない動作であると考えられるため，マルウェア検知の際の指標の一つとして利用できる可能性がある．そこで本論文では，実行毎の挙動の差異に基づいた新たなマルウェア検知手法を提案する．提案手法では，検査対象の実行ファイルを複数回動的解析した際のAPIログを比較することで挙動の差異を判断しマルウェアの検知を行う．

Malware Detection Method based on Difference between Behaviors in Multiple Executions

Takahiro Kasama†‡　　Katsunari Yoshioka†　　Daisuke Inoue‡　Tsutomu Matsumoto†

†Yokohama National University
79-7 Tokiwadai, Hodogaya, Yokohama, Kanagawa, 240-8501, JAPAN
kasama@mlab.jks.ynu.ac.jp, {yoshioka, tsutomu}@ynu.ac.jp
‡National Institute of Information and Communications Technology
4-2-1 Nukui-Kitamachi, Koganei, Tokyo, 184-8795, JAPAN
{kasama, dai}@nict.go.jp

Abstract Modern malware often change their runtime behaviors in each execution to avoid analyses and detections. For example, when malware copies itself on a file system, its file name can be randomly selected for avoiding detections. Another example of randomized behavior is when malware connects its command and control server, it randomly chooses its domain name from a domain name list to avoid being blocked by static blacklist. We consider such random behaviors unnecessary for benign software. Therefore these characteristics may be leveraged as a mean to distinguish malware from benign software. In this paper, we proposed a novel malware detection method based on the difference between behaviors in

multiple executions. In proposed method, we analyze a to-be-tested sample in the same analysis environment multiple times to get the lists of specific API call parameters, and compare them to detect the difference between behaviors.

1 はじめに

マルウェアの被害が深刻な社会問題となる中，マルウェアへの対策導出を目的とした多くの研究開発が行われている．その一方で，マルウェア作成者は解析や検知への耐性を持たせたマルウェアを作成するようになった．例えば，解析者による解析を妨害するために，解析時に典型的に用いられるデバッガや仮想化システムを検知し挙動の変更や停止を行う機能を持つマルウェアや，他のマシンに攻撃する際のエクスプロイトコードを難読化することで IDS 等による検知の回避を試みるマルウェアが存在する．そのようなマルウェアの中に，実行毎に挙動に変化を生じさせるマルウェアが存在する．例えば自身のコピーをファイルシステム内に複製する際の名前やレジストリキーを実行毎に異なる名前にしたり，C&C サーバ等のアクセスするサーバを変更したりする．このような動作をすることで，アンチウイルスソフトや IDS 等による単純なパターンマッチングによる検知を困難にする狙いがある．2009 年から爆発的な感染を引き起こした Conficker（別名 Downadup）も疑似乱数を用いてアクセス先のドメイン名を生成する機能によって，ブラックリストによる対策を困難にしていた[14]．

本論文では，このような実行毎の挙動の変動を，正規のプログラムには必要がなくマルウェアのみが有する機能と仮定し，ある実行ファイルがマルウェアか否かを判断するための指標として用いた新たなマルウェア検知手法を提案する．提案手法では，検査対象の実行ファイルを複数回実行して得られた API ログを比較することで実行毎の挙動の差異を判断しマルウェアの検知を行う．マルウェア検体としてインターネット上で収集した 1157 個と，正規プログラムとして Windows 7 のマシンから抽出した実行ファイルと，窓の杜[16]で公開されているフリーソフトの中からダウンロードした計 552 個を用いた評価実験の結果，提案手法によって約 65%のマルウェアが検知可能であり，正規プログラムをマルウェアとして誤検知する確率も 1%程度と低い値であったことから，今回用いた検体セットに対しては一定の有効性が示せた．

本論文の構成は以下の通りである．まず 2 章で関連研究について述べ，3 章で提案手法のシナリオと具体的な検知手法について述べる．その後 4 章で評価実験についてまとめ，5 章で考察を行い，6 章でまとめる．

2 関連研究

2.1 マルウェア動的解析

マルウェア動的解析とは，人手によるリバースエンジニアリング等の静的解析とは異なり，解析環境内でマルウェアを実際に動作させ，その際のマルウェアの挙動をモニタリングすることで解析を行う手法である．静的解析と比較すると比較的短時間での解析が可能であり，解析の自動化が容易であることから大量のマルウェアを処理するのに適している．動的解析は解析環境内からインターネットへのアクセスを許可するか否かによって，インターネット接続型[2, 6, 12, 13]と隔離型[3, 4, 15]の二種類に大別できる．

2.2 マルウェア特有の挙動に基づく検知手法

論文[1]では，エミュレータや仮想マシン上で実行された際に，その事実を検知し挙動を変化させるマルウェアを検知する方式を提案している．提案手法では，まず解析システムとは異なる環境（リファレンスシステム）を用意し，リファレンスシステム上でマルウェアを実行した時のシステムコール呼び出しの引数と返り値を取得する．次に解析システム上でマルウェアを実行し，その際のシステムコール呼び出しに対して，リファレンスシステム上で取得したログを基に，リファレンスシステム上と同じ返り値をマルウェアに対して返答する．その上で，両環境でマルウェアが呼び出すシステムコールのシーケンスを比較することで，挙動の変化を検知している．

論文[10]では，感染時に外部ホストと通信する目的でポート待ち受け状態になるマルウェアに着目し，そのようなマルウェアに感染したマシンを遠隔からネットワークスキャンによって検知する手法を提案している．提案手法はポート待ち受け状態のマルウェアが返答する特徴的な応答をシグネチャとして利用した検知手法であ

る.

論文[8]では，侵入挙動と攻撃機能を一つの検体の中に併せ持つタイプのボットを検知対象としている．これらのボットは初期感染時には侵入挙動（ファイル生成，自動実行リストへの登録）を示し，その後は攻撃挙動を示すため，ボットの実行環境を初期感染時に戻すことによって再度侵入挙動を示すと考えられる．そこで論文[8]ではこの侵入挙動の繰り返しをボット固有の挙動とみなした検知手法を提案している．

論文[9]では，マルウェアが検知回避を目的に，感染マシン上で動作しているアンチウイルスソフトやファイアウォールなどのセキュリティソフトのプロセスを強制終了させる機能に着目したマルウェア活動抑制手法を提案している．提案手法では，セキュリティソフトウェアに見せかけたおとりプロセスを動作させ，そのおとりプロセスを強制終了しようとするプロセスを検知し，当該プロセスをマルウェアプロセスと判定する．そしてマルウェアプロセスを逆に強制終了することで，マルウェアの活動抑制を行う．

3 提案手法

本章では実行毎の挙動の差異に基づくマルウェア検知手法を提案する．

まず，3.1 節で本手法を適用する際のシナリオ例を示し，3.2 節で提案手法の流れと検知に用いた API とその引数を示す．

3.1 検知シナリオ

提案手法は，検査対象の実行ファイルを複数回実行した際の挙動の差異を判断することでマルウェアを検知する手法となっている．つまり一つの実行ファイルを検査するためには，複数回の動的解析と観測した挙動の比較という処理が必要となる．特に動的解析の解析環境を動かすためには相応の負荷がかかるため，提案手法は一般のアンチウイルスソフトのようなユーザのマシンに常駐する形での使用には向かない．

そのため，提案手法を適用するシナリオとしては，ユーザマシンでの常駐型の使用形態ではなく，外部に検査を委託するクラウド型の使用形態を想定する．例えば，インターネット上でユーザからの検体投稿を受け付け，自動的に動的解析を行い，結果を提供する公開型マルウェア動的解析システム[7, 12, 13, 15]が現在多数公開されているが，このようなシステムにおいて投稿された検体がマルウェアか否かを判定するための一つの指標として活用できる．また，論文[5]において既存のユーザマシンで動作するアンチウイルスソフトをクラウド上のネットワークサービスとして提供するシステムが提案されている．論文[5]のシステムでは，個々のマシン上では軽量なクライアントのみが動作し，クライアントはマシン上のファイルの中から未検査のファイルを解析サービスへと送信し検査する．解析サービスでは受け取ったファイルに対して複数のアンチウイルスソフトや検知エンジンによって並列に解析が行われ，マルウェアと判定された場合には，クライアントによってファイルの実行を阻止したりすることが可能となる．ネットワークサービスとして実現するメリットとしては，複数の解析エンジンを用いることによる検知率の向上，マシン上で軽量なクライアントのみを動作させることで携帯端末などの計算資源の少ない環境でも使用可能，アンチウイルスソフト自体の脆弱性を悪用されることがない，などが挙げられている．このようなシステムにおいても，検査ファイルがマルウェアであるか否かの判定に提案手法が活用できる．また，類似するシステムとして論文[11]で提案されているシステムが存在する．論文[11]では，最終的な目標としてマルウェアと判定した場合の駆除ツールの自動生成を挙げているが，効果的な駆除ツールの作成のためには，マルウェアの実行毎の挙動の変化を把握しておくことは有用である．

3.2 提案手法の流れ

提案手法の検知の流れは以下の通りとなる

1. 検査対象のファイルについて，2 回動的解析を行い，それぞれの実行時の API 呼び出しのログを取得する
2. API ログから特定の API の引数のみを抽出し，リストアップする．なおこの際，API の呼び出し順は考慮せず，重複は除外する
3. 2 つのリストの項目を比較し，各リストに含まれる項目が完全一致，もしくは一方のリストの全項目がもう一方のリストに含まれている（包含されている）場合，正規プログラムと判定し，それ以外はマルウェアと判定する

表 1 に今回比較対象とした API と比較に用いた引数を示す．なおレジストリエントリ名については，自動実行

登録を目的として以下のRunキーに追加されたエントリ名のみを比較に用いた.

- HKEY_LOCAL_MACHINE¥Software¥Microsoft¥Windows¥CurrentVersion¥Run
- HKEY_CURRENT_USER¥Software¥Microsoft¥Windows¥CurrentVersion¥Run
- HKEY_LOCAL_MACHINE¥Software¥Microsoft¥Windows¥CurrentVersion¥RunOnce
- HKEY_CURRENT_USER¥Software¥Microsoft¥Windows¥CurrentVersion¥RunOnce

また, ファイル・ディレクトリ名については, Temp フォルダ (C:¥Document and Settings¥{user name}¥Local Settings¥Temp¥)に作成されたファイル・ディレクトリは比較対象から除外することとした.

表 1 比較に用いた API と引数リスト

API	比較に用いた引数
RegSetValueEx	レジストリエントリ名
RegSetValue	レジストリエントリ名
CreateFile	ファイル名
LZOpenFile	ファイル名
_lcreat	ファイル名
CreateDirectoryEx	ディレクトリ名
CreateDirectory	ディレクトリ名
CopyFileEx	コピー先ファイル名
CopyFile	コピー先ファイル名
LZCopy	コピー先ファイル名
MoveFileEx	移動先ファイル名
MoveFile	移動先ファイル名
DnsQuery	名前解決するドメイン名
gethostbyname	名前解決するドメイン名
InternetOpenUrl	アクセス先URL
HttpOpenRequest	リクエストパス
InternetConnect	アクセス先ドメイン名(or IPアドレス)
URLDownloadToFile	ダウンロードファイルのURL

4 評価実験

本章では, 提案手法の評価実験を行う.

まず 4.1 節で実験に用いた検体セットについて, 4.2 節で実験に用いた動的解析システムについて説明する. そして 4.3 節で実験結果を示す.

4.1 評価用検体

今回は評価用のマルウェア検体として, インターネット上で収集したマルウェア検体 1157 個を使用した. 図 1 にマルウェア検体のアンチウイルスソフトによる名前付けの結果を示す.

また, 評価用の正規プログラムとしては, 事務作業用 PC(Windows 7)内から抽出した実行ファイル 1181 個と窓の杜[16]で公開されているフリーのプログラムの中からダウンロードした 400 個の計 1582 検体を用いた. なお, 窓の杜で公開されているプログラムの中にはインストーラ型として配布されているのものも存在するがそれらについてはインストーラ自体を評価に用いた. さらに, 解析の際には実行ファイル単体を動的解析システムに投入・実行しており, プログラムと一緒に配布されていたファイル類は投入していない.

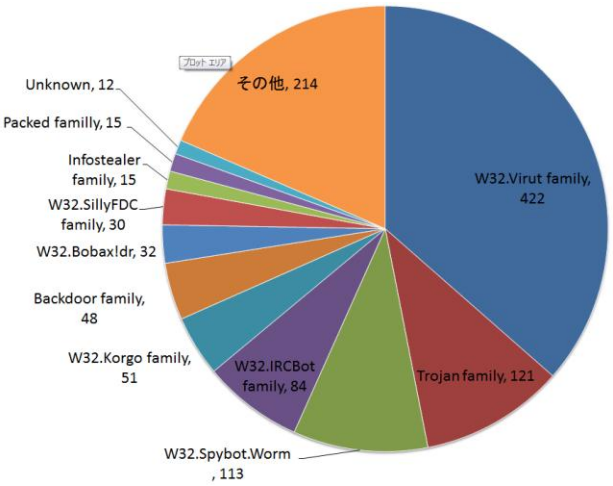

図 1 マルウェア名の割合(Symantec)

4.2 動的解析システム

今回の実験では, 動的解析システムとして NICT で研究開発を行っているミクロ解析システム[3]を利用し, 検体の実行環境の OS は Windows XP SP2, 検体実行時間は 60 秒で解析を行った.

ミクロ解析システムの特徴としては, 実行環境として実マシンを用いており, マルウェアによる仮想化システム検知やエミュレータ検知の影響を受けない. また, ミクロ解析システムは隔離型の動的解析システムであり, インターネットへ接続する代わりに解析環境内の疑似インターネットが応答を返す. 提案手法では実行ファイルの挙動へ影響を与える外的変化(アクセス先のサーバ応答の変化等)は極力減らすべきであるため, 隔離型の動的解析システムが適している. 挙動の観測に関しては, ミクロ解析システムでは API フックによって解析対象の実行ファイルの API 呼び出しを観測可能であり, この API ログを用いて挙動の差異を判断する.

4.3 実験結果

表 2 にマルウェア検体を用いた実験結果を示す. 表 2 及び表 3 の『全体』とは表 1 で示した API の引数全てを抽出し比較した結果である. 一方, レジストリ系 API,

ファイル系 API, ネットワーク系 API の結果は, 比較に用いた API をレジストリ系 API（RegSetValueEx, RegSetValue）, ファイル系 API（CreateFile, LZOpenFile, _lcreat, CreateDirectoryEx, CreateDirectory, CopyFileEx, CopyFile, LZCopy, MoveFileEx, MoveFile）, ネットワーク系 API（DnsQuery, gethostbyname, HttpOpenRequest, InternetConnect, URLDownloadToFile）に分け, それぞれの API の引数のみを抽出し比較した場合の結果である.

全ての API の引数を使用した結果を見ると. 全体の 65.2%のマルウェア検体が実行毎にレジストリ系, ファイル系, ネットワーク系のいずれかの挙動を変動させていることがわかった. また, 各 API の種類での結果を見ると, ファイル系 API での検知率が高いことがわかる. これは多くの検体が自身をコピーする際の名前を実行毎に変更していたからである. また, ネットワーク系 API の検知率も 5 割弱であった. これはアクセス先のサーバを変更するマルウェアが存在していたのに加えて, ランダムな宛先に対してネットワークスキャンを行うマルウェアが存在し, そのようなマルウェアも検知されているからである. 一方レジストリ系 API で検知された, つまり Run キーへ追加する際のエントリ名を変更したマルウェアは全体の 1/4 程度に留まった.

表 2 マルウェア検体を用いた検知率評価

	検知	未検知	検知率
全体	754	403	65.2%
レジストリ系API	314	843	27.1%
ファイル系API	725	432	62.7%
ネットワーク系API	541	616	46.8%

表 3 正規プログラムを用いた誤検知率評価

	検知	未検知	誤検知率
全体	9	543	1.6%
レジストリ系API	2	550	0.4%
ファイル系API	8	544	1.4%
ネットワーク系API	3	549	0.5%

表 3 に正規プログラムを用いた場合の実験結果を示す. ただし, 正規プログラムについては, 1581 個の実行ファイルを解析したが, 単純に実行ファイル単体を解析環境内に投入しただけでは実行できなかったものが多く, 表には実行できた正規プログラム計 552 個についての実験結果を示す. 表 3 を見ると全ての API の引数を比較した場合, 全 552 個中 9 個のプログラムがマルウェアとして誤検知されている. また, レジストリ系 API, ファイル系 API, ネットワーク系 API の結果を見ると, ファイル系 API で検知された結果が多い. 検知結果を確認したところ, RSA のキーコンテナ作成や, Temp フォルダ以外への一時ファイルらしきファイルの作成などでファイル名にランダムらしき文字列が用いられており, それが検知されていた.

5 考察

5.1 他の検知手法との併用

今回の実験では, 約 65%のマルウェアに挙動の変動がみられた一方で, 残りの 35%では挙動の変動がみられなかった. これはつまり, 約 35%のマルウェアは毎回同じファイル名でファイルを作成したり, 同じドメイン名（IP アドレス）にアクセスしたりしているということである. これらのマルウェアに対してはパターンマッチングによる検知が効果的だと考えられる. しかし, 検体一つ一つによって作成するファイル名が異なっているなどした場合, 検知用のシグネチャが膨大になってしまう可能性もある. 評価実験からもわかる通り, 提案手法単体の適用ではマルウェア全体の検知は難しいため, 提案手法と組み合わせた際に効果的な検知手法については今後の検討課題である.

5.2 一時ファイルについて

今回の実験ではアプリケーションやシステムで利用される一時ファイルが作成される Temp フォルダ（C:¥Document and Settings¥{user name}¥Local Settings¥Temp¥）内に作成されるファイルは比較対象から除外した. これは, 多くのアプリケーションが一時ファイルをランダムな名前で作成するため, Temp フォルダも対象にした場合, 誤検知が多くなってしまうからである. 実際に今回用いた検体セットに対して Temp フォルダ内へのファイル作成も比較対象とした場合, 正規プログラムの 2 割以上が誤検知されてしまった. 一方, Temp フォルダを除外することによって, Temp フォルダ内にファイル作成を行うマルウェアの挙動が検知対象外となってしまうが, 通常 Temp フォルダ内のファイルはアプリケーション実行後に削除されるものであり, Windows 標準のディスククリーンアップツールなどを使

用した場合もTempフォルダ内のファイルは削除されてしまう．そのため，マルウェアが自身のコピーなどの重要なファイルをTempフォルダ内に作成することは稀であると考えられ，検知結果にさほど影響はないと思われる．

5.3 マルウェアによる解析環境検知

今回の実験では，隔離型の動的解析システムを用いて評価実験を行った．提案手法では，検査対象プログラムを複数回解析する際に，プログラムの挙動に影響を与えるような環境の差異を極力減らすべきであるため，その観点からは疑似インターネットによって毎回同じ応答を返答するのが望ましい．ただし隔離型の場合，マルウェアの中には C&C サーバとの通信において，独自プロトコルを用いるようなマルウェアも存在するため，疑似インターネットが適切な応答を返答しない場合，マルウェアに解析環境が検知され、挙動を変更もしくは停止されてしまう可能性がある．それを回避するためには，実際のインターネット上のサーバからの応答を蓄積しておき，それを用いて適切な応答を返答する方法[4]が考えられるが，適用は今後の課題である．

6 まとめ

本論文では，マルウェアが解析や検知への耐性を目的として実行毎の挙動を変化させるという性質に着目し，そのような動作をマルウェア特有の挙動とみなした新たなマルウェア検知手法を提案した．提案手法では，検査対象の実行ファイルを複数回動的解析し，得られた API ログを基に挙動の差異を判定し，マルウェアの検知を行う．評価実験の結果，今回の評価用検体セットにおいては，正規プログラムの誤検知率を 1%程度に抑えながら約 65%のマルウェアを検知可能であり，提案手法の有効性が示せた．

参考文献

[1] D. Balzarotti, M. Cova, C. Karlberger, C. Kruegel, E. Kirda, and G. Vigna, “Efficient Detection of Split Personalities in Malware,” In Proceedings of 17th Annual Network and Distributed System Security Symposium (NDSS 2010), 2010.

[2] U. Bayer, C. Kruegel, and E. Kirda, “TTAnalyze: A Tool for Analyzing Malware,” In Proceedings of 15th Annual Conference of the European Institute for Computer Antivirus Research (EICAR), 2006.

[3] D. Inoue, K. Yoshioka, M. Eto, Y. Hoshizawa, and K. Nakao, “Automated Malware Analysis System and its Sandbox for Revealing Malware’s Internal and External Activities,” IEICE Trans. Vol. E92D, No 5, pp.945-954, 2009.

[4] T. Kasama, K. Yoshioka, T. Matsumoto, M. Yamagata, M. Eto, D. Inoue, K. Nakao, "Malware Sandbox Analysis with Automatic Collection of Server Responses using Dummy Client," Proc. of the 5th Joint Workshop on Information Security (JWIS2010), 5A-2, 2010.

[5] J. Oberheide, E. Cooke, F. Jahanian, “CloudAV: N-Version Antivirus in the Network Cloud,” In Proceedings of 17th USENIX Security Symposium, 2008.

[6] C. Willems, T. Holz, and F. Freiling, “Toward Automated Dynamic Malware Analysis Using CWSandbox,” Security & Privacy Magazine, IEEE, Vol.5, Issue 2, pp.32-39, 2007.

[7] K. Yoshioka, Y. Hosobuchi, T. Orii, and T. Matsumoto, "Your Sandbox is Blinded: Impact of Decoy Injection to Public Malware Analysis Systems," IPSJ Journal, Vol.19, pp.153-168, 2011.

[8] 酒井 崇裕, 竹森 敬祐, 安藤 類央, 西垣 正勝, “侵入挙動の反復性を用いたボット検知方式,” 情報処理学会論文誌 Vol51, No 9, pp.1591-1599, 2010.

[9] 松木 隆宏, 新井 悠, 寺田 真敏, 土居 範久, “セキュリティ無効化攻撃を利用したマルウェアの検知と活動抑止手法の提案,” 情報処理学会論文誌 Vol.50, No 9, pp.2127-2136, 2009.

[10] 吉岡 克成, 村上 洸介, 松本 勉, “マルウェア感染ホスト検出のためのネットワークスキャン手法と検出用シグネチャの自動生成,” 情報処理学会論文誌 Vol.51, No 9, pp.1633-1644, 2010.

[11] 川口 信隆, 余田 貴幸, 川口 龍之進, 寺田 真敏, 笠木 敏彦, 星澤 裕二, 衛藤 将史, 井上 大介, 中尾 康二, “マルウェア対策ユーザサポートシステムを用いた CCC DATASet 2010 の解析,” マルウェア対策研究人材育成ワークショップ 2010 (MWS2010), 2010.

[12] Anubis: Analyzing Unknown Binaries, http://anubis.iseclab.org/

[13] CWSandbox, http://www.cwsandbox.org/

[14] Encyclopedia entry: Win32/Conficker - Learn more about malware, http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=Win32%2fConficker

[15] Norman Sandbox, http://www.norman.com/security_center/security_tools/

[16] 窓の杜, http://www.forest.impress.co.jp/