動的解析における検体動作時間に関する検討

青木 一史† 川古谷 裕平† 岩村 誠† 伊藤 光恭†

†NTT 情報流通プラットフォーム研究所
180-8585 東京都武蔵野市緑町 3-9-11
{aoki.kazufumi,kawakoya.yuhei,iwamura.makoto,itoh.mitsutaka}@lab.ntt.co.jp

あらまし マルウェアは日々多数の亜種が出現するため，効率よく解析し挙動を明らかにするには，動作させることで解析結果を取得できる動的解析が有効な手段となる．しかしながら，動的解析は解析中に動作した箇所の解析結果しか得られないため，解析時間を十分長く設定しなければ，得られる解析結果は不十分なものとなってしまう．本研究では、動的解析結果がどの程度得られたかの指標として，解析時に実行されたコード量に着目し，CCC DATAset 2010 マルウェア検体と D3M 2010 マルウェア検体を閉環境/開環境で動的解析したときに実行されたコード量と解析時間の推移を分析する．また，分析結果をもとに，動的解析を行う際に設定するマルウェアの動作時間に関する検討を行う．

Investigation about Malware Execution Time in Dynamic Analysis

Kazufumi Aoki† Yuhei Kawakoya † Makoto Iwamura† Mitsutaka Itoh†

†NTT Information Sharing Platform Laboratories
3-9-11 Midori-Cho, Musashino-Shi, Tokyo 180-8585 JAPAN
{aoki.kazufumi,kawakoya.yuhei,iwamura.makoto,itoh.mitsutaka}@lab.ntt.co.jp

Abstract Since there are many malware in the wild, dynamic analysis, which executes the malware to obtain its behavior, is one of the most useful approach to analyze their behavior efficiently. In dynamic analysis, however, we can only obtain the behavior that was performed during the analysis time frame, it is necessary to execute the malware with sufficient time frame to obtain its activities well. In this paper, we investigate the relationship between executed malware code volume and analysis time frame on the CCC DATAset 2010 and the D3M 2010 based on open/closed dynamic analysis system and we consider the analysis time frame to set in malware dynamic analysis.

1 はじめに

DDoS 攻撃やスパムメールの送信といった攻撃の多くは，マルウェアに感染したコンピュータによってもたらされている．したがって，マルウェア対策を行うことで，インターネットにおけるこれらの脅威を低減させることができる．効果的なマルウェア対策を講じる上では，マルウェアに感染した端末がどのような振る舞いをするのかを把握することが効果的である．そのため，マルウェアの挙動を解析する技術が重要となる．

マルウェアの解析方法は，“静的解析（ホワイ

トボックス解析）” と “動的解析（ブラックボックス解析）” に大別される．静的解析は，解析者が逆アセンブラやデバッガなどを利用し，マルウェアの機能を解明する方法である [1]．マルウェアの全コードを解析すれば，解析対象のマルウェアがどのような脅威をもたらすのかを詳細に把握することができる．しかしながら，静的解析には高度な技術力が必要であり，また，複雑なマルウェアであれば，解析の難易度が高くなり，解析にかかる時間も長くなってしまう．一方の動的解析は，マルウェアを実際に動作させ，その挙動を監視することで解析結果を取得する解析方法である．静的解析のように，解析者がマルウェアを個別に解析せずとも，自動的に解析を行うことができるため，解析自体にかかる労力を大きく減らすことができる．昨今は，マルウェアの新種や亜種が日々出現しており，静的解析ではマルウェアの網羅的な解析は困難である．そこで，比較的容易に解析結果を取得できる動的解析が注目されている [2, 3, 4]．

しかしながら，動的解析の場合は，解析中に実行されたコードに基づく解析結果しか得られない．したがって，動的解析を行う際には，各検体の解析時間を適切に設定しなければ，マルウェアのコードが十分に実行されず，得られる解析結果が不十分なものとなる可能性がある．

マルウェアは，外部への通信や内部リソースへのアクセスなど，様々な挙動を示す．これらの挙動は，それらのためのプログラムコードを実行することで行われる．よって，動的解析でどの程度マルウェアの挙動を把握できたのかは，解析中に実行されたコード量に依存すると考えられる．つまり，解析時間と実行されたコード量の関係性を明らかにすることは，動的解析を行う際に設定すべき解析時間を決める上での有用な手がかりとなる．

本稿では，動的解析によりマルウェアの挙動をどの程度把握できたのかの指標として，解析中に実行されたコード量に着目し，解析時間と実行されたコード量の関連性について，CCC DATAset 2010 マルウェア検体 [5] と D3M 2010 マルウェア検体 [5] を対象に分析した．解析環境については，インターネット環境から隔離された閉環境と，インターネットに接続可能な開環境の 2 パターンを用いて実験を行った．

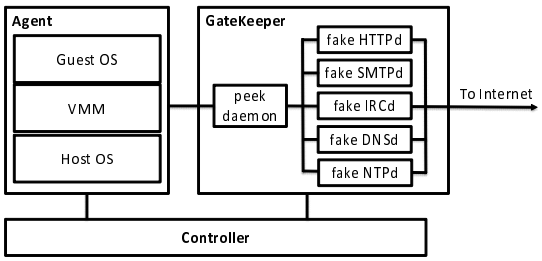

図 1: Botnet Watcher の概要

2 解析手法

動的解析中に実行されたコード量を観測するために，Botnet Watcher[6] と Stealth Debugger[7] を用いた．以下で，本稿で用いたこれらの技術について説明する．

2.1 Botnet Watcher

Botnet Watcher は，マルウェアからの通信に対する応答を生成する．図 1 は Botnet Watcher の概略図である．Botnet Watcher は，マルウェアを動作させる Agent，Agent からの通信に対する応答を行う GateKeeper，Agent 及び GateKeeper の制御を行う Controller で構成される．

Agent は，ホスト OS と VMM[1]，ゲスト OS から構成される．解析を行う際は，Agent 内のゲスト OS にマルウェアをインストールし，挙動を監視する．

Agent からの通信は GateKeeper で終端する．GateKeeper は，Agent から受け取った通信のペイロードを解析し，HTTP / SMTP / IRC / DNS / NTP のいずれのプロトコルであったかを判定する．

開環境での解析を行う場合には，IRC / HTTP（GET リクエストのみ）と判定された通信は，マルウェアからのペイロードを実際のインターネットに転送する．インターネットから取得した応答は，そのままマルウェアに転送する．DNS と判定された通信は，ペイロードから問い合わ

[1]VMM：Virtual Machine Monitor

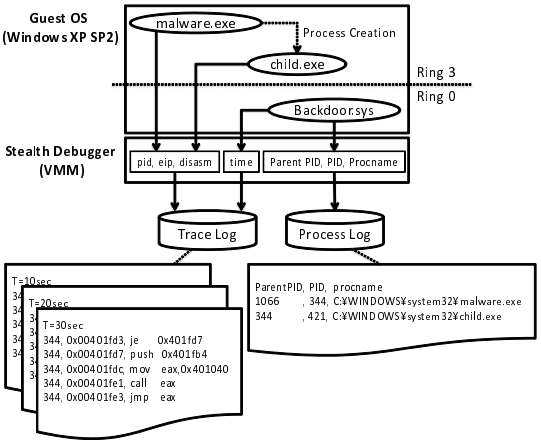

図 2: Stealth Debugger を用いた実行トレース収集システムの概要

せ FQDN を抽出し，GateKeeper が外部の DNS サーバに問い合わせ，その結果を Agent に送信する．NTP と判定された通信は，GateKeeper で実時刻に相当する応答を生成し，Agent に送信する．

閉環境での解析を行う場合には，TCP の通信についてはセッションの確立までは行うが，その後のマルウェアからのリクエストに対する応答は行わない．また，DNS に対する応答は GateKeeper 内部で擬似応答を生成し，マルウェアに送信する．なお，NTP に対する応答は，開環境での場合と同様に，実時刻に相当する応答を生成し，Agent に送信する．

2.2 Stealth Debugger

解析時に実行された命令（実行トレース）の収集には，Stealth Debugger を用いた．Stealth Debugger は，仮想マシンを利用し，ゲスト OS の外側からデバッグ機能を提供するツールである．

マルウェアには種々のアンチデバッグ機能が存在するため，通常のデバッガでは実行トレースを十分に取得できない可能性が高い．また，アンチデバッグ機能が存在しないマルウェアであっても，ユーザモードのデバッガで実行トレースの取得を試みると，コンテキストスイッチが多発し，実行トレースの取得に時間がかかってしまう [8]．マルウェアは他のサーバと通信を行うことがあるため，実行トレースの取得に時間がかかってしまうとセッションがタイムアウトしてしまい，インターネット上のサーバとのやり取りを踏まえた実行トレースが取得できなくなってしまう．

Stealth Debugger の場合，マルウェアが動作しているレイヤと異なるレイヤで監視するため，マルウェアが有する種々のアンチデバッグ機能を回避しながら実行トレースを取得することができる．また，ゲスト OS が実行した命令を VMM で監視するため，実行トレースの収集に伴うコンテキストスイッチの発生を抑えることができ，実 OS の動作時間とほぼ同じ速度でマルウェアを動作させることができる．

Stealth Debugger を利用した実行トレース収集システムの概略図を図 2 に示す．マルウェアによっては，自身を別のディレクトリにコピーし，コピーされたマルウェアを実行した後，最初に実行されたマルウェアの動作を停止させるものが存在する．そこで，解析対象のマルウェアと，マルウェアによって生成された子プロセスを実行トレースの取得対象とした．なお，起動及び終了したプロセスの PID，プロセス名，親子関係については，ゲスト OS にインストールしたカーネルモジュールにより Stealth Debugger に通知し，ログを出力できるようにしている．

コードが実行された時刻の取得には，ゲスト OS 内にインストールしたカーネルモジュール用いた．ゲスト OS の動作は，ホスト OS と比べてわずかに速度が低下するため，正確なマルウェアの動作時間を計測するには，ゲスト OS 内の時刻を用いる必要がある．そのため，ゲスト OS 内で動作するカーネルモジュールから時刻を Stealth Debugger に通知し，実行トレースのログがいつ実行されたものであるかを識別できるようにした．

3 マルウェア検体を用いた実験

3.1 実験環境

解析環境で CCC DATAset 2010 マルウェア検体を動作させたところ，SHA1 のハッシュ値が 43A5*のものは正常に動作しなかった．そこ

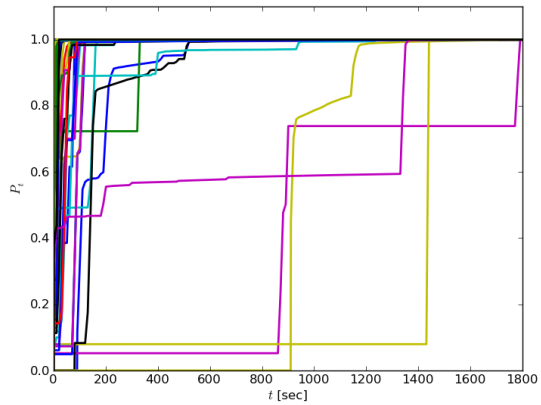

図 3: CCC DATAset 2010 マルウェア検体における動的解析時間と実行コード割合の関係（閉環境）

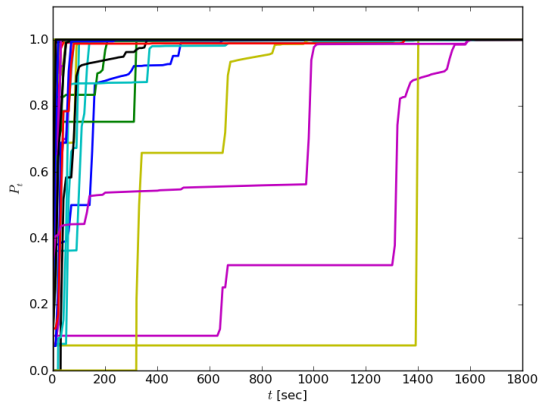

図 4: CCC DATAset 2010 マルウェア検体における動的解析時間と実行コード割合の関係（開環境）

で，このハッシュ値を持つ検体を除いた 49 検体と，D3M 2010 マルウェア検体である 3 検体を対象に，前述の手法を用いて実行トレースの収集を行った．Botnet Watcher の Agent では，VMM として Stealth Debugger を用い，ゲスト OS として Windows XP SP2 をインストールした．各検体の解析では，ゲスト OS 内でマルウェアを 1800 秒間（30 分間）動作させ，通信が発生した場合には，Botnet Watcher により閉環境または開環境で応答を行うようにした．なお，各環境における動的解析については，2010 年 8 月 23 日から 2010 年 8 月 24 日の期間に行った．

実行された命令のカウントでは，命令実行時の EIP と命令の逆アセンブリをペアとしてカウントした．ただし，EIP と逆アセンブリのペアが同一のものが繰り返し出現した場合には，最初に出現した時点でのみカウントした．また，実行トレースログの肥大化を防ぐために，Windows がデフォルトで備えている cmd.exe や ipconfig.exe のような正規プログラムをマルウェアが子プロセスとして生成した場合，これらの正規プログラムに対するトレースは取得しなかった．

3.2 実験結果

設定した解析時間を t_{max}，解析開始からの経過時間を t，t までに実行されたコード量を V_t としたとき，t における実行コード割合 P_t を式 1 のように定義する．

$$P_t = \frac{V_t}{V_{t_{max}}} \tag{1}$$

また，t_{max} でコードが十分実行され，$V_t = V_{t_{max}} \times p$ となったときに解析が完了したものとみなすとき，t で解析が完了した検体の割合 M_{fin} を式 2 のように定義する．

$$M_{fin} = \frac{M_t}{M_{all}} \tag{2}$$

なお，M_{all} は解析対象の全マルウェア検体数，M_t は t で解析が完了したマルウェア検体数，p は解析が完了したとみなす際の閾値を設定するための定数（0.0〜1.0）である．

CCC DATAset 2010 マルウェア検体と D3M 2010 マルウェア検体を対象に，閉環境及び開環境で動的解析を行い，解析時間に対する P_t と M_{fin} の変化について分析を行った．

3.2.1 CCC DATAset 2010 マルウェア検体での実験結果

CCC DATAset 2010 マルウェア検体を閉環境及び開環境で動的解析を行った際の t と P_t の

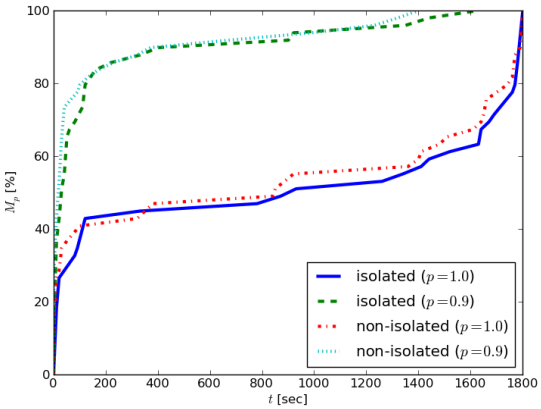

図 5: M_{fin} の時間変化（$p = 0.9, p = 1.0$）

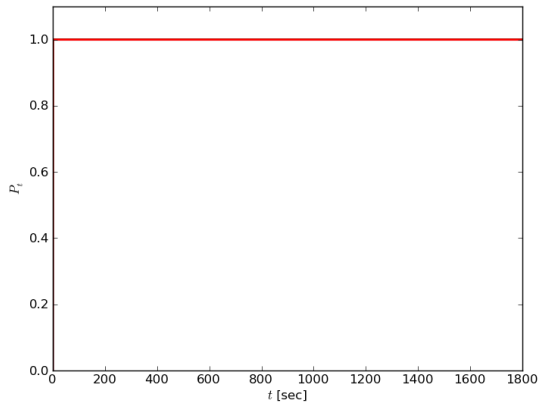

図 6: D3M 2010 マルウェア検体における動的解析時間と実行コード割合の関係（閉環境）

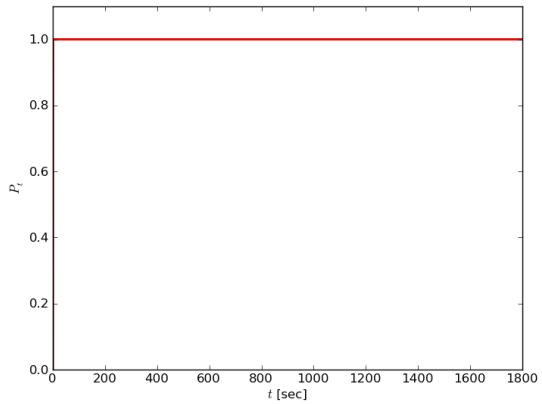

図 7: D3M 2010 マルウェア検体における動的解析時間と実行コード割合の関係（開環境）

関係を，それぞれ図 3 及び図 4 に示す．閉環境及び開環境のいずれの結果でも，解析直後に多くのコードを実行し，以降では実行コード量の変化が少なくなる検体が多数確認された．一方で，解析開始から数分間が経過してから，実行コード量が増加するような検体も確認された．このような検体の場合，しばらくは実行コード量に変化が見られないが，ある時点で急激に実行コード量が増加するような様子が確認された．

また，$V_t = V_{t_{max}} \times p$（$p = 0.9$ または 1.0）となった検体数の推移を図 5 に示す．$p = 1.0$ とした場合，つまり $V_t = V_{t_{max}}$ となった時点で解析が完了したと判定する場合には，60 秒間の解析では，閉環境だと約 30%，開環境だと約 37%のマルウェアについて解析が行えたとみなせる．また，$p = 0.9$ とした場合には，閉環境の場合は約 120 秒，開環境の場合は約 100 秒を解析時間と設定すると，データセットの約 80%の検体の解析を十分に行うことができたとみなせる．

3.2.2 D3M 2010 マルウェア検体での実験結果

D3M 2010 マルウェア検体について，閉環境及び開環境で動的解析を行ったときの t と P_t の関係を，それぞれ図 6 及び図 7 に示す．D3M 2010 マルウェア検体のの全てにおいて，解析実施直後にコードを実行した後，新たなコードを実行する様子は確認されなかった．これは，解析環境が閉環境か開環境かの如何に関わらず同様であった．

4 考察

実行コード量の分析の結果，CCC DATAset 2010 のマルウェアの約 8 割については，100 秒から 120 秒を解析時間と設定すれば，1800 秒間で実行されるコードの約 90%を実行した挙動を確認できるという結果が得られた．ただし，マルウェアの挙動が変化する要因として，動作している日時やユーザインタラクションなども考えられるため，動的解析中に実行されるコード

量を増加させる，解析時間以外についても検討を進める必要がある．

また，今回の解析では，解析時間の上限を1800秒と設定している．この時間で全てのマルウェアが十分に動作するのかは，静的解析を行って明らかにする必要がある．仮に，1800秒では不十分であったとしても，全てのマルウェアについて非常に長い時間をかけて解析を行うのは，昨今のマルウェア数に見られる増加傾向を踏まえると難しいと考えられる．したがって，動的解析では短時間でできるだけ多くの挙動を分析できるようにし，分析ができていない箇所については，動的解析で取得した実行トレースを活用して，静的解析にかかる時間を削減するという手段が効果的ではないかと考えられる．

5 まとめ

本稿では，CCC DATAset 2010 マルウェア検体と D3M 2010 マルウェア検体を用いて，動的解析時の解析時間と実行コード量の関連性を分析した．

実験では，CCC DATAset 2010 マルウェア検体の約 8 割については，100 秒から 120 秒程度を解析時間と設定すれば，1800 秒間で動作するコードの約 90%を実行したときの解析結果が得られるという結果が得られた．しかしながら，動的解析で全てのマルウェアを十分に動作させるには，それ以上に長い解析時間を設定しなければならない．また，マルウェアの挙動に変化をもたらす契機には，特定の日時であることや，ユーザによる操作なども考えられるため，動的解析により多くの挙動を把握するには更なる検討が必要である．

本稿では，データセットとして提供された 50 検体程度のマルウェアのみを解析対象とした．そのため，今回得られた結果が世の中の全マルウェアを網羅した結果であるとは言いがたい．今後は，能動的攻撃や受動的攻撃により拡散するマルウェア幅広く収集し，それらを対象として実験を行い，動的解析における解析時間の推定を行う．

参考文献

[1] Konstantin Rozinov. Reverse code engineering: An in-depth analysis of the bagle virus. *6th Annual IEEE SMC Information Assurance Workshop*, 2005.

[2] Norman. Norman sandbox. `http://www.norman.com/technology/norman_sandbox/`, 2009.

[3] Anubis. Anubis - analyzing unknown binaries. `http://anubis.iseclab.org/`, 2009.

[4] Carsten Willems, Thorsten Holz, and Felix Freiling. Toward automated dynamic malware analysis using cwsandbox. *IEEE Security and Privacy*, Vol. 5, No. 2, pp. 32–39, 2007.

[5] 畑田充弘, 他. 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, 2010.

[6] 青木一史, 川古谷裕平, 岩村誠, 伊藤光恭. 半透性仮想インターネットによるマルウェアの動的解析. マルウェア対策研究人材育成ワークショップ 2009, 2009.

[7] 川古谷裕平, 岩村誠, 伊藤光恭. ステルスデバッガを利用したマルウェア解析手法の提案. マルウェア対策研究人材育成ワークショップ 2008, 2008.

[8] Rafal Wojtczuk. umss: efficient single stepping on Win32. `http://www.avertlabs.com/research/blog/index.php/2006/11/29/umss-efficient-single-stepping-on-win32/`, 2006.