社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2007－ＣＳＥＣ－38（38）
2007／7／20

マルウェアの動作条件の抽出

星澤 裕二† 岡田 晃市郎† 山村 元昭† 椎木 孝斉‡

†株式会社セキュアブレイン 〒102-0083 東京都千代田区麹町 2-6-7
‡有限責任中間法人 JPCERT コーディネーションセンター 〒101-0054 東京都千代田区神田錦町 3-17
E-mail: †{yuji_hoshizawa, kouichirou_okada, motoaki_yamamura}@securebrain.co.jp, ‡office@jpcert.or.jp

あらまし マルウェアのホストやネットワーク上での振る舞いを知る方法として，他のシステムに影響を与えない隔離された環境でマルウェアを実行し，その挙動を観察する動的解析がある．レジストリやファイルへのアクセスを記録したり，ネットワークトラフィックをキャプチャしたりするツールが数多く存在するので，比較的容易に環境を構築することが実現でき，短時間である程度の結果を得ることができる．しかし，日時により処理を分岐したり，特定のファイルが存在する場合のみ動作したりといった特定の条件下で動作するマルウェアを解析する場合には，その条件の洗い出しとマルウェア実行環境の調整が必要となり，短時間で解析することは困難である．本稿では，動的解析手法を用いたマルウェア挙動解析の解析時間の短縮と解析結果の精度を高めるためにマルウェアの動作条件を自動的に抽出する方法を検討する．

キーワード マルウェア，ボット，動作条件，自動化

Exploring the required conditions for malware to run for behavior analysis

Yuji HOSHIZAWA† Kouichirou OKADA† Motoaki YAMAMURA† and Takayoshi SHIIGI‡

†SecureBrain Corporation 2-6-7 Kojimachi, Chiyoda-ku, Tokyo, 102-0083 Japan
‡Japan Computer Emergency Response Team Coordination Center 3-17 Kandanishikicho, Chiyoda-ku, Tokyo, 101-0054 Japan
E-mail: †{yuji_hoshizawa, kouichirou_okada, motoaki_yamamura}@securebrain.co.jp, ‡office@jpcert.or.jp

Abstract As a way of knowing host or behavior of malware on the network, we run malware in an isolated environment that doesn't affect other systems and monitor the behavior, which is called dynamic analysis. It is relatively easy to realize and get a certain level of results quickly, because there are many tools that record the access to the registry or files and capture the network traffic. However, it is far from easy to analyze malware that work under specific conditions, such as malware that change process depending on time and date or the day of the week, or work only when there are particular files. In this paper, we consider the way of exploring required conditions of malware automatically, to reduce the analyzing time and to improve the accuracy of the result of analysis using the dynamic analysis.

Keyword Malware, Bot, Required conditions, Automation

1. はじめに

マルウェアのホストやネットワーク上での振る舞いを知る方法として，他のシステムに影響を与えない隔離された環境でマルウェアを実行し，その挙動を観察する動的解析がある．レジストリやファイルへのアクセスを記録したり，ネットワークトラフィックをキャプチャしたりするツールが数多く存在するので，比較的容易に環境を構築することが実現でき，短時間である程度の結果を得ることができる．しかし，日時により処理を分岐したり，特定のファイルが存在する場合のみ動作したりといった特定の条件下で動作するマルウェアを解析する場合には，その条件の洗い出しとマルウェア実行環境の調整が必要となり，短時間で解析することは困難である．本稿では，動的解析手法を用いたマルウェア挙動解析の解析時間の短縮と解析結果の精度を高めるためにマルウェアの動作条件を自動的に抽出する方法を検討する．正常なプログラムと同様にマルウェアも動作に必要な環境が整っていなければエラーが発生し，それ以上処理を続けることはできなくなる．例えば，インターネットに接続する機能をもったものは，インターネットへの接続確認を行い，接続できなければ処理を中止するといったことである．また，環境だけでなく，特定のファイルの存在や特定の日時であるかどうかといった条件をチェックするこ

とも考えられる．

通常，これらの環境や条件を確認するためには，Windows API が呼び出される．API を使用して情報を取得し，取得した情報を元に分岐処理を行っている．このことからマルウェアコードの条件分岐処理の前方に存在する動作条件に関係する API 呼び出し部分に注目すれば動作条件を抽出することが可能と考えられる．

2. 動作条件の種類

マルウェアの動作条件を分類し，それぞれ利用される API を列挙してみる．

2.1. システム時刻

感染ホストのシステム時刻をチェックして処理を分岐するケースが考えられる．システム時刻のチェックには，GetLocalTime，GetSystemTime，GetSystemTimeAsFileTime，time，GetTimeZoneInformation が利用される．

上記以外に，NTP サーバから時刻を取得する方法も考えられる．この場合，単一の API 呼び出しではなく複数の API の組み合わせになる．

2.2. 処理時間

あらかじめ想定した処理時間と実際にかかった処理時間を比較し，その結果により処理を分岐するケースが考えられる．

これはデバッグモードで起動されているかどうかを判断するために利用されることがある．具体的には，予想される処理時間より実際の処理が長い場合，デバッグモードで実行されている判断する．処理時間をチェックするためには，GetTickCount，QueryPerformanceCounter が利用される．

2.3. CPU 使用率

CPU 使用率をチェックし，CPU がアイドル状態の時に動作を行うといったケースが考えられる．CPU 使用率のチェックには，GetSystemTimes，NtQuerySystemInformation が利用される．

2.4. インストールされているプログラム

インストールされているプログラムをチェックし，その有無により処理を分岐するケースが考えられる．インストールされているプログラムの存在を確認する方法には次のようなものがある．

（1）ファイルの存在チェック

動作に必要なファイルの存在を確認するためには，FindFirstFile，SearchPath が使用される．

（2）DLL 存在チェック

動作に必要な DLL の存在を確認するためには，LoadLibrary が使用される．

（3）サービス存在チェック

サービスの存在を確認するためには，OpenSCManager，OpenService，QueryServiceStatus，CreateNamedPipe，CreatePipe が使用される．

2.5. ネットワーク関連アプリケーションの存在

ネットワーク関連アプリケーションの存在をチェックして，処理を分岐するケースが考えられる．

ネットワーク関連アプリケーションの存在を確認する方法には，メール機能を扱うためのアプリケーションの存在をチェックする方法と，アドレスブックが存在するかどうか確認する方法がある．前者には，MAPIInitialize，MAPILogonEx を使用し，後者には OpenAddressBook を使用する．

2.6. OS 環境

Windows のバージョンやロケールをチェックし，その内容により処理を分岐するケースが考えられる．

Windows のバージョンの確認には GetVersionEx，ロケールの確認には GetLocaleInfo, GetUserDefaultLCID, IsValidLocale を使用する．

2.7. ネットワーク環境

感染ホストのネットワーク接続状態などをチェックして処理を分岐するケースが考えられる．ネットワーク環境を確認する方法には次のようなものがある．

（1）Winsock 使用可否

Winsock が使用可能かどうかを確認するためには，WSAStartup を使用する．

（2）インターネット接続状態

インターネットへの接続状態を確認するためには，InternetGetConnectedState，InternetGetConnectedStateEx，InternetAttemptConnect を使用する．

（3）共有資源へのアクセス状態

共有資源へのアクセス状態を確認するためには，WNetOpenEnum, WNetEnumResource，NetShareEnum, NetUseAdd を使用する．

2.8. サーバ接続状態

DNS や HTTP，FTP サーバなどに接続が可能かどうかをチェックし，処理を分岐するケースが考えられる．アクセスするサーバの種類によって確認方法が異なるため，次のように分類する．

（1）DNS サーバへの接続

DNS サーバに接続可能かどうかを確認するためには，DnsQuery，gethostbyname を使用する．

（2）HTTP/FTP サーバへの接続

HTTP と FTP サーバに接続可能かどうかを確認するためには，InternetConnect，InternetOpenUrl，URLDownloadToFile を使用する．

（3）DNS/HTTP/FTP サーバ以外への接続

DNS や HTTP，FTP サーバ以外のサーバに接続可能かどうかを確認するためには，socket，connect を使用する．

2.9. DCOM への接続

DCOM に接続可能かどうかを確認し，処理を分岐するケースが考えられる．DCOM への接続確認には，RpcBindingFromStringBinding が利用される．

2.10. レジストリの設定

レジストリにアクセスし，その内容により処理を分岐するケースが考えられる．レジストリの内容を確認するために，RegOpenKeyEx，RegQueryValueEx，が利用される．

2.11. デバッグモードでの動作

デバッグモードで実行されているかどうかを確認し，処理を分岐するケースが考えられる．これはマルウェア解析を困難にするためのテクニック「アンチデバッギング」のひとつの手法である．デバッグモードで実行されているかどうかを確認するためには，IsDebuggerPresent，CheckRemoteDebuggerPresent，NtQueryInformationProcess，SetDebugPrivilege，GetExceptionCode を利用する．

3. 検出方法の検証

20 個程度のボットの逆アセンブラを行い検出方法の検証を行った．以下に上記検出方法を適用したときに抽出可能と思われる動作条件による処理分岐を行っているマルウェアコードの抜粋を記述する．

3.1. 処理時間

API 名	GetTickCount()		
MEW:00402FCB	mov	esi, GetTickCount	
MEW:00402FD1	call	esi ; GetTickCount	
MEW:00402FD3	mov	[ebp-14h], eax	
MEW:00402FD6	call	esi ; GetTickCount	
MEW:00402FD8	sub	eax, [ebp-14h]	
MEW:00402FDB	cmp	eax, 0AFC8h	
MEW:00402FE0	jnb	short loc_403009	

3.2. インストールされているプログラムファイル

API 名	LoadLibrary()		
UPX0:00401887	call	LoadLibraryA	
UPX0:0040188C	mov	esi, eax	
UPX0:0040188E	or	esi, esi	
UPX0:00401890	jz	loc_401926	
UPX0:00401896	push	offset aInternetgetcon ; "InternetGetConnectedStateEx"	
UPX0:0040189B	push	esi	; hModule
UPX0:0040189C	call	GetProcAddress	

3.3. OS 環境

API 名	GetLocaleInfo()		
CODE:004064F3	call	GetLocaleInfoA	
CODE:004064F8	xor	esi, esi	
CODE:004064FA	cmp	[ebp+LibFileName], 0	

CODE:00406501	jz	loc_4065EA	
CODE:00406507	cmp	[ebp+LCData], 0	
CODE:0040650B	jnz	short loc_406517	
CODE:0040650D	cmp	[ebp+Data], 0	
CODE:00406511	jz	loc_4065EA	
CODE:00406517	lea	eax, [ebp+LibFileName]	
CODE:0040651D	push	eax	
CODE:0040651E	call	lstrlen	
CODE:00406523	mov	ebx, eax	
CODE:00406525	lea	eax, [ebp+LibFileName]	
CODE:0040652B	add	ebx, eax	

3.4. ネットワーク環境

API 名	WSAStartup()		
UPX0:004016B7	call	WSAStartup	
UPX0:004016BC	or	eax, eax	
UPX0:004016BE	jz	short loc_4016C7	
UPX0:004016C0	push	0	; uExitCode
UPX0:004016C2	call	ExitProcess	

3.5. ネットワーク接続状態

API 名	InternetGetConnectedStateEx()		
UPX0:00401B80	call	InternetGetConnectedStateEx	
UPX0:00401B86	test	eax, eax	
UPX0:00401B88	jnz	short loc_401BA3	
UPX0:00401B8A	push	80h	
UPX0:00401B8F	push	offset aNotConnected ; "not connected"	
UPX0:00401B94	lea	ecx, [ebp+var_98]	
UPX0:00401B9A	push	ecx	
UPX0:00401B9B	call	sub_40BAF0	
UPX0:00401BA0	add	esp, 0Ch	

3.6. サーバ接続状態

API 名	connect()		
UPX0:0040202A	call	socket	
UPX0:0040202F	mov	ebx, eax	
UPX0:00402031	mov	edi, [ebp+var_8]	
UPX0:00402034	mov	s[edi*4], ebx	
UPX0:0040203B	push	10h	; namelen
UPX0:0040203D	lea	eax, [ebp+var_174]	
UPX0:00402043	push	eax	; name
UPX0:00402044	push	ebx	; s
UPX0:00402045	call	connect	
UPX0:0040204A	mov	[ebp+ThreadId], eax	
UPX0:00402050	cmp	eax, 0FFFFFFFFh	
UPX0:00402053	jnz	short loc_40206A	
UPX0:00402055	push	ebx	; s
UPX0:00402056	call	closesocket	
UPX0:0040205B	push	7D0h	; dwMilliseconds
UPX0:00402060	call	Sleep	
UPX0:00402065	jmp	loc_402128	
UPX0:0040206A	lea	eax, [ebp+name]	
UPX0:00402070	push	eax	
UPX0:00402071	push	offset	

```
aConnectedToS  ; "connected to %s."
```

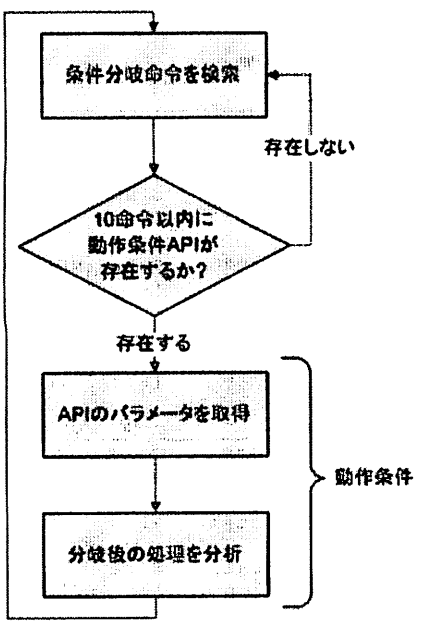
4. 動作条件抽出の自動化

マルウェアを静的解析することにより動作条件を抽出することは可能であるが，静的解析手法ではすべての動作条件を確実に洗い出すことは困難である．静的解析手法による動作条件抽出の自動化で最大の難点は，API のパラメータがプログラム実行時に動的に生成されるような場合に完全な条件を抽出できないことである．この部分については，静的解析手法だけではなく，エミュレーション技術や動的解析手法を取り入れる必要がある．

本調査により，マルウェアの動作条件に関係する API を整理し，静的解析を行う場合に，どの API に注目して解析すれば動作条件を見つけることができるかがわかった．しかし，静的解析をマニュアルで行い，動作条件を見つけるのは時間のかかる作業である．

そのため，本調査では，得られた知見から動作条件の抽出を自動化することを検討する．

下図は，動作条件自動検出のフローである．まず，マルウェアコードの先頭から条件分岐命令を検索する．発見した場合，前方に本調査で整理したマルウェアの動作条件に関係する API が存在するかをチェックする．存在する場合，API のパラメータと分岐後の処理を分析し，それらを動作条件として記録する．これらをマルウェアコード全体に対して行う．



必要と思われる動作条件を列挙するだけであれば，動作条件 API を検索するだけで十分である．それだけでも，実験環境を当該マルウェアが実行できるように再構築するヒントにはなるが，十分な情報とは言えない．やはり，API のパラメータと分岐後の処理の分析が必要である．しかし，前述のようにプログラム実行時に動的に値が決定されるようなものは，静的解析手法のみを用いたケースでは分析は非常に難しい．また，値があらかじめ決められたものだとしても，分岐後の処理の分析は難易度が高い．

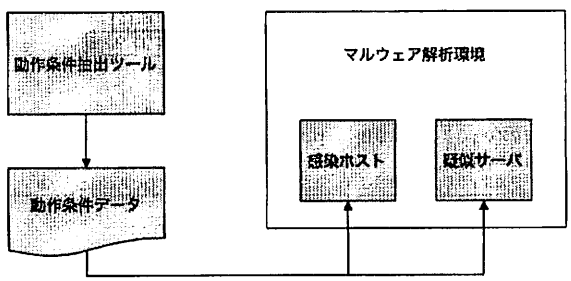
動作条件の抽出を高速に行うためには，分岐後の処理の分析の範囲を狭くするといった工夫も必要である．例えば，数 10 命令以内に存在するプログラムの終了コードのみを検索するといったものにする．

また，API の引数に関して引数にレジスタを使用したポインタ参照などになっている場合には引数の取得方法を検討する必要がある．取得できなかった API の引数情報を動的解析と同様の方法で取得などの方法が考えられる．

5. まとめ

検証によりマルウェアコードの条件分岐処理の前に動作条件を取得可能な Windows API を呼び出している部分に注目すれば，何らかの動作条件を抽出することが可能である．

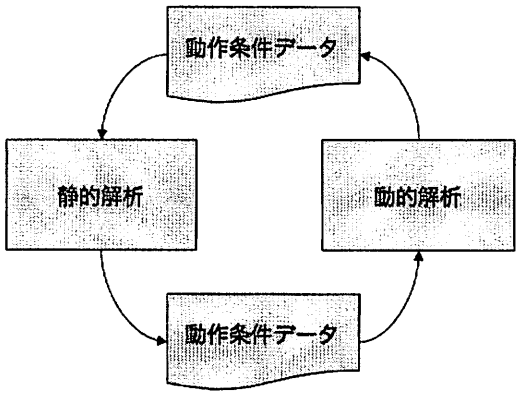
短時間で動作条件を抽出することが目的の場合，前述の動作条件抽出アルゴリズムにより自動的にある程度の動作条件を抽出することが可能である．さらに，抽出された動作条件を人間可読データとして出力すれば，マルウェアを解析するエンジニアは実験環境を適切な状態に変更し，解析することが可能である．また，機械可読データとして出力し，それに対応した自動環境変更システムを構築すれば，マルウェア解析の自動化を実現できる．



しかし，前述の動作条件抽出アルゴリズムは，短時間で動作条件を抽出することを目的としているため，

抽出される動作条件は完全なものではない．実際には，実験環境をカスタマイズするためのヒントとして利用することになるだろう．

動作条件の抽出精度を向上させるには，静的解析手法のみによるアプローチだけでなく，動的解析手法も併用した手法が必要と思われる．二つあるいはそれ以上の手法をうまく組み合わせることで，抽出精度と抽出速度の両面の向上が期待できる．



動作条件抽出精度の向上などを検討した場合に，最終的に，動作条件の抽出は，マルウェアの振る舞いを解析することとほぼ同じことをしなければならない．今後の研究では，さまざまな解析手法を活用し，動作条件を抽出するだけでなく，マルウェアの挙動を自動的に解析するシステムを構築することを検討すべきである．

今後の課題としては，次のようなものが考えられる．

- レジストリなどの内容を確認する場合，API の呼び出しと戻り値だけではなく，API に渡した引数などをチェックする方法を検討する．
- 短時間で動作条件を抽出する方法では，調査対象プログラムが終了する処理を行う部分（経路）のみに着目し，動作条件を抽出する．
- 動的解析手法を利用し，動作条件を抽出することを検討する．具体的には，API 呼び出しログやサーバアクセスログを分析し，条件を抽出する方法などが考えられる．
- ボットのように，受信したコマンドにより処理を分岐するような場合，API のチェックだけでは動作条件を抽出することができないので，別な方法を検討する．
- 動作条件データの形式を検討する．また，動作条件データにより自動的に実験環境の環境を変更できるような仕組みを検討する．
- 本書で提案した動作条件抽出方法と動的解析手法を利用した動作条件抽出方法の連携を検討する．

6. 謝辞

本研究は平成 18 年度に JPCERT コーディネーションセンターから委託を受け実施した「ボット静的解析に関する調査・研究」の成果の一部である．

本研究を進めるにあたって有益な助言と協力を頂いた関係者各位に深く感謝致します．