G-09

Windows API コールログからのマルウェアの動作再現
-主要な Win32 API への対応と動作再現の評価-

松田 尚也† 福田 洋治† 廣友 雅徳‡ 白石 善明*
Matsuda Naoya Fukuta Youji Hirotomo Masanori Shiraishi Yoshiaki

1. はじめに

情報セキュリティ対策におけるインシデント対応では，根絶と復旧の過程で，ネットワーク構成や保存されているログに基づき，被害を受けた可能性のある端末や情報を洗い出し，影響の範囲を確定させるといった，証拠の収集，処理が行われる[1][2].

証拠の収集，調査の場面では，保存されたログから，インシデント発生時の状況の再現，追跡が試みられることがあるが，インシデントにマルウェアを用いた攻撃が含まれている場合，現場でマルウェアそのものを扱うことは，インシデントの影響範囲が拡大する危険を伴う.

著者らは，マルウェアのような不正なプログラムを直接扱うことなく，その挙動を再現，改めて観測できるようにするために， Windows API [3]コールのログから,マルウェアの動作を再現，観測を行うツール[4][5]（以下，本ツール）を継続して検討している.

本稿では，ランサムウェアのようなマルウェアに対応するため，新たにウィンドウやメッセージ，ファイル・ディレクトリ，システム，デバイス入出力の API に対応した，試作中のツールの機構，動作実験，ペネトレーションテストツール Metasploit[6]のペイロードを用いた動作再現割合の評価について述べる.

2. 関連研究

マルウェアの挙動を取得する研究として，Cao らは，ホストイベントやネットワークがシミュレートされている仮想マシン内の仮想マシンモニタ層で，疑わしい実行可能ファイルの動作や，メインプロセスやサブプロセスなど全てのプロセスの挙動をキャプチャし，Windows API コールやその引数に基づいてマルウェアの挙動を捉え解析を行うシステム[7]を提案している.

また，マルウェアの挙動を把握，再構築する研究として，Shosha らは，マルウェアの疑いがある実行ファイルを解析し，低レベルの機械語の命令から，コードの制御フローグラフを有限オートマトンへモデル化する手法[8]や，マルウェアの静的解析による API 呼び出し時に，欠損してしまう引数を自動で判断し，高レベルの言語のプログラムに再構築し，マルウェアの正確な振る舞いを把握する手法[9]を提案している.

他にも，マルウェアの振る舞い分析の研究として，Jang らは，最長共通部分列(LCS)のアルゴリズムを利用し，API 呼び出しパターンと事前に定義された疑わしい API リストとの共通部分を検出し，悪意のある振る舞いをするメソッドかどうかを分析することで，インシデント対応または対策開発を支援する手法[10]を提案している.

†近畿大学, Kindai University
‡佐賀大学, Saga University
* 神戸大学, Kobe University

3. Windows API コールログを用いたマルウェアの動作再現ツール

3.1 Windows API

Windows API とは，Application Program Interface，すなわち，Windows オペレーティングシステムがアプリケーションソフトウェアに対して公開しているインターフェースである．アプリケーションソフトウェアは，オペレーティングシステムが用意した API を経由して目的の処理を実現する[11].

Windows API は 2 種類存在する．32 ビットマイクロプロセッサ向けの Win32API，と 64 ビットマイクロプロセッサ向けの Win64API である．本研究では Win32API について扱う.

3.2 Windows API のモニタリングツール

プログラムの動的解析を目的とした，プログラムが動作中に発行する Windows API コールをモニタするツールの例を示す.

- **APIMonitor**[12]
  アプリケーションから呼ばれる API コールの引数，戻り値をモニタできるツール．任意のアプリケーションを起動してモニタできるほか，実行中のアプリケーションに対しての API コールも監視できる.
- **Detours**[13]
  Microsoft よりオープンソースで公開されているもので，コマンドプロンプト上で任意のアプリケーションのパスを指定し，実行された API の履歴，並びにその戻り値をテキストファイルに保存することができる.

アプリケーションから呼び出される Windows API をモニタすることによって，そのアプリケーションがどのような動作をしたか詳細に把握できる.

本研究で試作中のツールは，上で挙げたようなツールを用いて，マルウェアが特徴的な挙動を見せたときの Windows API コールログが既に取得されており，これを利用することを前提としている.

3.3 ツールの構成と動作

Windows API コールログを用いてマルウェアの動作再現を行うツールの構成を図 1 に示し，本ツールの各機能を以下に示す.

- **ログ抽出部**…API コールログファイルから時系列順に１行ずつログを読み込み API 名，引数，戻り値などを抽出する

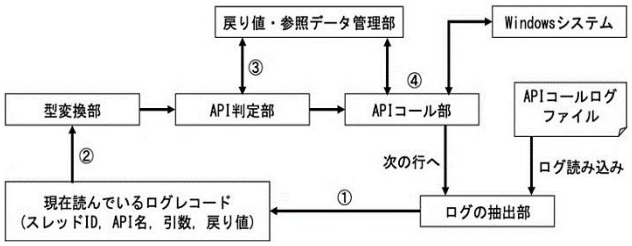

図 1 WindowsAPI コールログを用いた
マルウェアの動作再現ツールの構成

- **API 判定部**…ログ抽出部で抽出された API 名を用いて，どの API が呼び出されているか判別し，引数の型も判別させる
- **型変換部**…API 判定部で判別した API の引数の型に適するように，ログ抽出部で抽出した引数を変換する．必要に応じて，戻り値・参照データ管理部から，ハンドルなどのデータを受け取ることや，登録を行う
- **API コール部**…判別した API に引数を与えて，Windows システムから API を呼び出す．戻り値や参照データを取得した場合，戻り値・参照データ管理部に渡す
- **戻り値・参照データ部**…API の実行によって得られたハンドルなどの戻り値や参照データが返ってきた際に，保管しておき，その後の API の実行の際に必要なデータを渡す

本ツールの動作の流れを以下に示す．また，ツールの前提条件として，マルウェアの API コールログが取れていること，マルウェアで使用されている API が Win32API であることが挙げられる．

① API コールログファイルをツールに読み込ませ，ログ抽出部で，現在読んでいるログレコードの API 名，引数，戻り値を抽出する
② API 判定部で抽出した API 名を用いて，どの API であるか，引数の型は何かを判別する
③ 型変換部で，ログから抽出した引数や，引数に対応する戻り値・参照データ管理部が保持するデータを受け取り，当該 API の引数の型に変換する
④ API コール部では当該 API に引数を与えて呼び出し，戻り値や参照データを得ると，それらを戻り値・参照データ管理部に渡す

API 判定部で判別できなかった未実装である API の場合は，次のステップへ移らず次のログレコードへと移動する．

プロセス・スレッド管理の API へ対応するため，図 2 に示すようなマルチスレッド処理を再現するための機構がログ抽出部に実装されている．

この機構は，ログに含まれるスレッド ID を用いて，メインのスレッド，もしくは新しく作られたサブスレッドのどちらで呼び出された API なのかを判別し，対応するスレッドから API 呼び出しを行うことで，擬似的にマルチスレッド処理を再現している．

本ツールは，インシデント対応における調査活動を支援するためのフォレンジック支援のために，API コールログに従って，当時呼び出された API を時系列順に忠実に呼び出し，当該プログラムの動作を再現するものであり，端末に対する入出力，端末の状態によって挙動を変えることには，現在対応していない．

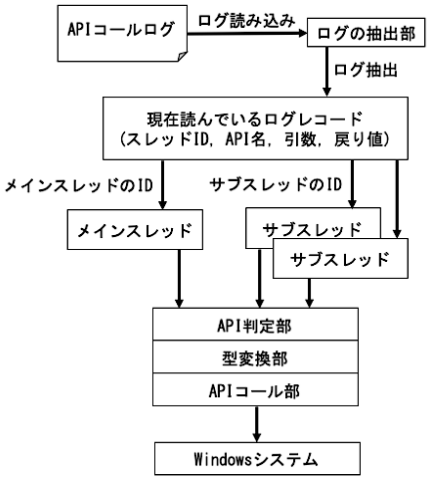

図 2 マルチスレッド処理を再現するための機構

今までに対応している API はファイル・ディレクトリ，通信，プロセス・スレッド管理，レジストリ操作カテゴリの基本的な Win32API（CopyFile，CreaeFile，CreateDirectory，CreateDirectoryEx，DeleteFile，MoveFile，MoveFileEx，WriteFile，InternetReadFile，InternetOpen，InternetOpenUrl，HttpOpenRequest，HttpSendRequest，HttpQueryInfo，InternetConnect，InternetCrackUrl，IntetnetSetOption，InternetCloseHandle，CloseHandle，CreateEvent，OpenEvent，SetEv-ent，ResetEvent，CreateThread，SuspendThread，ExitThread，ResumeThread，WaitForSingleObject，RegCreateKeyEx，RegSetValueEx，RegOpenKeyEx，RegQueryValueEx，RegCloseKey）である．

3.6 新たに追加した API

新たに 13 種類の API を追加実装した．追加実装した API を以下に示す．

- **CreateWindowEx**…拡張ウィンドウスタイル付きのウィンドウを作成する
- **IsWindow**…指定されたウィンドウハンドルを持つウィンドウが存在するかどうかを調べる
- **ShowWindow**…指定されたウィンドウの表示状態を設定する
- **SendMessage**…ウィンドウへ複数のメッセージを送信する
- **PostMessage**…メッセージキューにメッセージをポストする
- **PeekMessage**…メッセージキューにポスト済みメッセージがあるかどうか確認し，構造体に格納する
- **MessageBoxA**…メッセージボックスの作成，表示，操作を行う
- **GetFileSizeEx**…指定されたファイルのサイズをバイト単位で取得する
- **RemoveDirectory**…空のディレクトリを削除する
- **FindFirstFileEx**…指定された名前と属性が一致するファイルのディレクトリを検索する
- **ExitWindowsEx**…ログオフ，シャットダウン，再起動を行う
- **GetUserName**…システムにログオンしているユーザの名前を取得する

- **DeviceIoControl**…指定されたデバイスドライバへ制御コードを直接送信し，対応するデバイスに対応する動作をさせる

追加対象の API は，株式会社 FFRI のマルウェア解析レポート[14]などのマルウェアの使用する API から，まだ実装できていないものを抜き出し，ランサムウェアのようなウィンドウを表示するマルウェアに対応できるよう，ウィンドウの作成，表示を行う API を選択した．

ウィンドウの表示の際，RegisterClass を用いて，雛型の登録を行わなければならないが，登録の際のデータが，現在のデータだけでは再現できないので，CreateWindowEx 内に仮実装している．

表 1 にマルウェアによってよく使用される API リストの対応状況を示す．マルウェアがよく使用する API リストのカバー率は昨年 56%だったものが，69%に増加した．メモリ，メッセージ，ウィンドウ，DLL，プロセス，スレッド，同期の API 553 種類[4]のうち，実装した割合は 5.6%から 7.7%に増加した．

4. 実験・考察

4.1 動作実験

追加実装した API コールの再現ができるかどうかを確かめるために，次のような実験を行った．なお，実験の端末は Windows10 64bit Home の PC を使用し，API コールログの取得は APIMonitor[15]を用いて取得した．

1) メッセージボックスが表示されるとディレクトリが削除され，新たに Malware という名のディレクトリを作成，そのディレクトリ内に malware というテキストファイルを作成するプログラムを C++言語で作成する．
2) 作成したプログラムの API コールログを APIMonitor で取得し，ファイルに API コールログを保存する．
3) 本ツールにファイルに保存した API コールログを与え，プログラムの動作再現をさせる．
4) 本ツールの API コールログを APIMonitor で取得し，ファイルに保存された API コールログと一致するかどうか確認する．

表 1 マルウェアがよく使用する API リストの対応状況

API 名	対応状況
CopyFile, CreateFile, CreateDirectoryEx, DeleteFile, MoveFileEx, WriteFile, InternetOpen, InternetOpenUrl, HttpSendRequest, InternetConnect, InternetCrackUrl, CreateThread, WaitForSingleObject, RegCreateKey, ExRegSetValueEx, RegOpenKey, RegQueryValueEx	対応済み
GetFileSizeEx, RemoveDirectory, FindFirstFileEx, ExitWindowsEx, GetUserName	今回対応
GetSystemDirectory, FindWindowEx, FindNextFile, GetWindowsDirectory, FindResourceEx, GetDiskFreeSpaceEx, GetDriveType, RegQueryInfoKeyEx, WSASocket, GetCurrentProcess, GetVersionEx, RegEnumKeyEx	未対応

本ツールで動作再現を行なったときに取得した API コールログを図 3 に，実験用プログラムを動作させたときに取得した API コールログを図 4 に示す．API コールログが一致しており API コールが正しく時系列順に再現できていることがわかる．

RemoveDirectoryA や CreateDirectoryA，CreateFileA の API コールが正しく動作したかを確認するために，ツールを実行する前のディレクトリの中身を図 5 に，ツール実行後のディレクトリの中身を図 6 に示す．また，図 7 に Malware ディレクトリの中身を示す．

図 5，図 6 のディレクトリの中身を見ると，UBI というディレクトリが削除され，新たに Malware というディレクトリが作成されていることがわかる．図 7 より，Malware ディレクトリの中には malware というテキストファイルが作成されたことがわかる．

これらの結果から，ディレクトリを削除する RemoveDirectoryA，ディレクトリを作成する CreateDirectoryA，ファイルを作成する CreateFileA が正しく呼び出せていることがわかる．本ツールで，メッセージボックスを表示しディレクトリを削除・作成，ファイルの作成が可能であることを確認した．

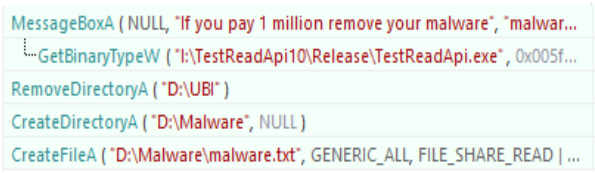

図 3 本ツールで動作再現したときに取得した API コールログ

図 4 実験用プログラムを動作させたときに取得した API コールログ

図 5 本ツール使用前のディレクトリの中身

図 6 本ツール使用後のディレクトリの中身

図 7 Malware ディレクトリの中身

4.3 評価実験

本ツールの課題となっていた実際のマルウェアの API コールログからの動作再現の実験を行い，再現率を求める評価実験を実施する．今回の評価実験ではマルウェアと同等の働きをする，Metasploit のペイロードを用いて次のような手順で行った．なお，MessageBox と exec のペイロードは Windows10Pro 64bit の PC で，format_all_drives と adduser のペイロードは Windows7 SP1 Pro 32bit の PC で動作させ，Metasploit は Kali Linux(2019.4)に標準インストールされているものを使用した．

1) Metasploit のペイロードを msfvenom で実行ファイル化する．
2) 実行ファイル化したペイロードを動作させ，API コールのログを APIMonitor で取得する．
3) 同じ環境で本ツールに読み込ませ動作させ API コールのログを取得する．
4) ペイロードから取得した API コールログと本ツールから取得した API コールログを用いて，動作再現の割合を求める．

動作再現の割合は以下の式の通りに求める．分子は本ツールで再現できた API（コール）の個数とし，分母はペイロードを動作させて取得した API（コール）の個数とする．

$$動作再現率(\%) = \frac{本ツールで再現できたAPIコールの個数}{ペイロードを動作させて取得したAPIコールの個数} \times 100$$

評価で使用した metasploit のペイロードの動作は次の通りである．

- **messagebox**…メッセージボックスを表示．ボタンやテキストは任意に設定可能
- **exec**…パスで指定した実行ファイルを実行
- **adduser**…ユーザの追加．ユーザ名とパスワードは任意に設定可能
- **format_all_drive**…フォーマットできるドライブをすべてフォーマットし，ボリュームラベルを変更

API コールの再現を確認した例として，図 8，図 9 に本ツールで CreateWindowEx の呼び出しを再現したときに取得した API コールログと，ペイロードを動作させたときに取得した API コールログを示す．CreateWindowEx の呼び出しが再現できていることがわかる．

図 10，図 11 に本ツールで DeviceIoControl の呼び出しを再現したときに取得した API コールログと，ペイロードを動作させたときに取得した API コールログを示す．DeviceIoControl の呼び出しが再現できていることがわかる．

図 12，図 13 に本ツールで DllMain の呼び出しを再現したときに取得した API コールログと，ペイロードを動作させたときに取得した API コールログを示す．dll 呼び出しのための DllMain が再現できていると分かる．それに加え，RtlInitUnicodeString や RtlAllocateHeap，NtDeviceIoControl-

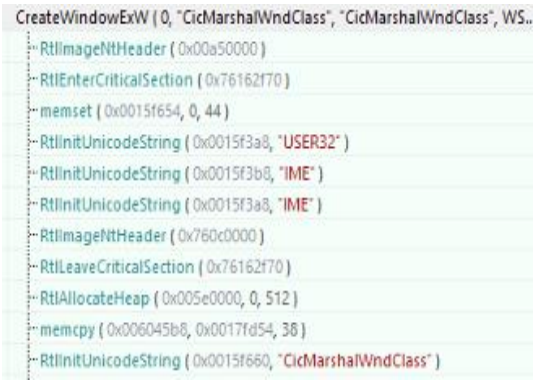

図 8 本ツールで CreateWindowExW の呼び出しを再現したときに取得した API コールログ

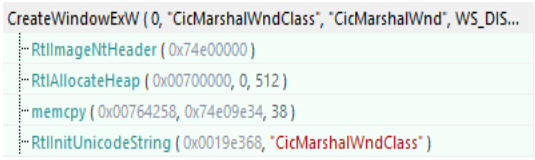

図 9 ペイロードを動作させたときに取得した CreateWindowExW の API コールログ

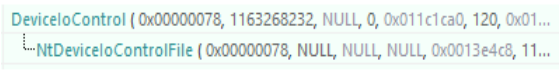

図 10 本ツールで DeviceIoControl の呼び出しを再現したときに取得した API コールログ

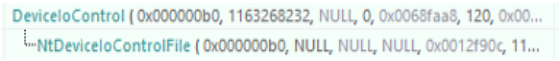

図 11 ペイロードを動作させたときに取得した DeviceIoControl の API コールログ

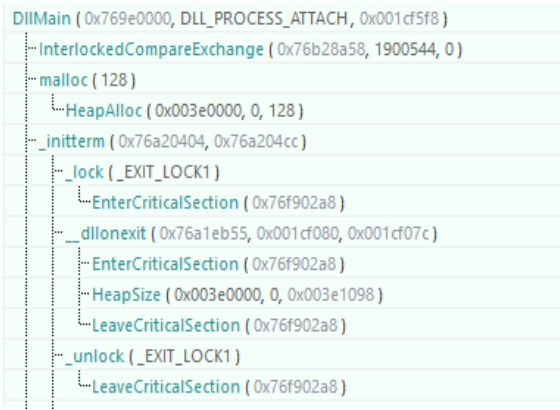

図 12 本ツールで DllMain の呼び出しを再現したときに取得された API コールログ

File，RtlImagentHeader といった，関連して API コールされている部分も再現されていることがわかる．

再現できている API コールの割合をまとめたものを表 2 に示す．それぞれのペイロードの動作再現率は，messagebox が約 18%，exec が約 18%，format_all_drives が約 24%，adduser が約 26%となった．この結果より，今回用意したペイロードでの，本ツールの動作再現の割合は，2 割前後であることが確認できた．

図 13 ペイロードを動作させたときに取得した DllMain の API コールログ

図 14 ペイロードを動作させたときに取得した未実装の API コールログ

表 2 評価実験の結果

ペイロード名	再現できた API 数	ペイロードの API 数	動作再現割合 (%)
messagebox	6556	35525	18
exec	40	221	18
format_all_drives	6332	25257	24
adduser	78	294	26

Metasploit のペイロードを使った評価では，図 14 に示すような，API の呼び出しが未実装であったことが判明した．NtAllocateVirtualMemory や NtWriteVirtualMemory，NtResumeThread といった API はコードインジェクションで使用される API である．NtAllocateVirtualMemory と，NtWriteVirtualMemory は，主に対象プロセス内のメモリ確保及びコード書き込みに使用され，NtResumeThread は主に書き込んだコードの実行に主に使用される．他にも，指定した文字にバッファーを設定する memset やバッファー間でバイトをコピーする memcpy のようなメモリ関係の API 呼び出しも未実装であった．実際のマルウェアの動作を再現するにはさらなる追加実装が必要と考えられる．

5. おわりに

マルウェアの挙動を示す API コールログから，ログの時系列順に API 呼び出しを実行し，マルウェアそのものを扱わずにその動作を再現する，再現ツールについて検討し，ウィンドウやメッセージ，ファイル・ディレクトリ，システム，デバイス入出力の API に対応する実装を行った．

実装した機構の動作を確認する実験を行い，ツールに追加した実装が意図したとおり動作すること，新たに対応したウィンドウやメッセージ，ファイル・ディレクトリ，システム，デバイス入出力の API の呼び出しが可能であることを確認した．

ペネトレーションテストツール Metasploit のペイロードを動作させたときに取得した API コールログと，本ツールによる動作再現を行ったときに取得した API コールログを比較し，動作再現率を調べる評価実験を行い，メッセージ，ファイル・ディレクトリカテゴリの API の呼び出しが可能であること，ペイロードの動作再現を行うには，さらなる API 呼び出しの追加実装が必要であることを確認した．

今後の課題として，コードインジェクションのような攻撃によく使われる API である，NtAllocateVirtualMemory，NtResumethread，NtWriteVirtualMemory や，memset や memcpy のような API を追加実装することが挙げられる．

参考文献

[1] IPA，インシデント対応へのフォレンジック技法の統合に関するガイド，入手先〈https://www.ipa.go.jp/files/000025351.pdf〉（参照 2020-07-22）．

[2] Jason T.Luttgens，Matthew Pepe,Kevin Mandia インシデントレスポンス第 3 版コンピュータフォレンジックの基礎と実践，高橋 聡，露久保 由美子ら（訳）日経 BP 社(2016)．

[3] MSDN：Windows API リスト，入手先<https://msdn.microsoft.com/ja-jp/windows/hh240557>(参照 2020-07-22)．

[4] 近藤秀紀ほか：Windows API のログからのマルウェアの疑似復元の検討，平成 29 年度電気関係学会関西連合大会，G11-16，pp.332-333(2017)．

[5] 末吉真也ほか：Windows API コールログからのマルウェアの動作再現の検討，情報処理学会 第 81 回全国大会講演論文集，6W-01，pp.527-528(2019)．

[6]RAPID7，metasploit，入手先〈https://www.metasploit.com〉（参照 2020-07-22）

[7] Ying Cao，Qiguang Miao，et.al，Osiris: A Malware Behavior Capturing System Implemented at Virtual Machine Monitor Layer，2012 Eighth International Conference on Computational Intelligence and Security，Guangzhou，pp. 534-538 (2012)．

[8] Ahmed F. Shosha，Joshua I. James，et.al，Towards Automated Forensic Event Reconstruction of Malicious Code (Poster Abstract)，RAID 2012，LNCS 7462，pp.388-389 (2012)．

[9] Ahmed F. Shosha，Lee Tobin，Pavel Gladyshev，Digital Forensic Reconstruction of a Program Action，2013 IEEE Security and Privacy Workshops，San Francisco，CA，pp.119-122 (2013)．

[10] Jae-wook Jang，Huy Kang Kim，Function-Oriented Mobile Malware Analysis as First Aid，Mobile Information Systems，2016 入手先<http://downloads.hindawi.com/journals/misy/2016/6707524.pdf >(参照 2020-07-22)

[11] 北山洋幸： Win32/64API システムプログラミング 32/64 ビットの共存コンパクト版，株式会社カットシ

ステム(2011).
[12] Rohitab.com：API Monitor, 入手先 <http://www.roh-itab.com/apimonitor>(参照 2020-07-22).
[13] GitHub.com：microsoft/Detours, 入手先 <https://github.com/microsoft/Detours>(参照 2020-07-22).
[14] FFRI BLOG：緊急レポート：韓国サイバー攻撃マルウェア詳細解析結果, <https://www.ffri.jp/blog/2013/03/2013-03-27.htm>(参照 2020-07-22).
[15] Rohitab.com：API Monitor, 入手先<http://www.roh-itab.com/apimonitor>(参照 2020-07-22).