Windows 7 x64 環境におけるマルウェア解析向けデータ取得法

中野 進† 大月 勇人† 明田 修平† 瀧本 栄二†
齋藤 彰一‡ 毛利 公一†

† 立命館大学
525-8577 滋賀県草津市野路東 1-1-1
{snakano, yotuki, saketa, takimoto, mouri}@asl.cs.ritsumei.ac.jp

‡ 名古屋工業大学
466-8555 愛知県名古屋市昭和区御器所町
shoichi@nitech.ac.jp

あらまし 増加を続けるマルウェアへ対策を行うためには，それぞれの挙動を明らかにしなければならない．Alkanet は，解析動作を妨害するアンチデバッグ機能を持つマルウェアに対しても有効な動的解析機構である．現行の Alkanet は，マルウェア解析環境として 32bit 版 Windows XP のみを対象としているが，より新しいバージョンの Windows へとシェアが移行してきている．そのため，普及率の高い 64bit 版 Windows 7 を新たな解析環境とし，32bit 環境と 64bit 環境における差異と Windows のバージョン間の差異に対処することで Alkanet の情報取得部を対応させる．

Data Acquisition for Malware Analysis on Windows 7 x64

Susumu Nakano† Yuto Otsuki† Shuhei Aketa† Eiji Takimoto†
Shoichi Saito‡ Koichi Mouri†

†Ritsumeikan University
1-1-1 Nojihigashi, Kusatsu, Shiga 525-8577 Japan
{snakano, yotuki, saketa, takimoto, mouri}@asl.cs.ritsumei.ac.jp

‡Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya, Aichi, 466-8555 Japan
shoichi@nitech.ac.jp

Abstract Malware analysis is required to deal with threats of malware. Alkanet is a dynamic malware analysis system, which is useful for analyzing malware armed anti-debugging functions. Alkanet targets the malware which works on Windows XP x86, but the new version of Windows became the top of market share. A target of malware also has become the version that is most frequently used, therefore we are developing new Alkanet for Windows 7 x64. In this paper, we describe the data acquisition for malware analysis on Windows 7 x64.

1 はじめに

マルウェアの脅威は，大きな社会問題となっている．その対策には，マルウェアの挙動を明らかにする必要がある．我々は先行研究で，マルウェア動的解析システム Alkanet[1] を提案してきた．Alkanet は，Windows が発行するシステムコールをトレースすることでマルウェア

の挙動を解析する．ただし，現在 Alkanet は，マルウェアの動作環境として 32bit 版 Windows XP ServicePack 3 のみを対象としている．しかし，Windows XP は 2014 年 4 月にサポートが終了した．それにともない，2015 年 7 月時点で Windows XP は OS の世界シェアが 12%まで減少し，Windows 7 が 60%を占めている [2]．そのため，マルウェアの攻撃対象も Windows 7 へ移行している．Windows 7 には 32bit 版と 64bit 版の 2 つが存在しており，64bit 版が主流となっている．そこで，本論文では，Windows 7 x64 におけるマルウェア解析のためのシステムコールフックによるデータ取得法について述べる．

Alkanet は，解析に必要な情報として，発行されたシステムコールの種類や引数，発行元のプロセスやスレッドの情報を取得する．しかし，32bit 版 Windows XP と 64bit 版 Windows 7 間で行われた様々な変更により，既存手法では必要な情報の取得が行えなくなった．そのため，64bit 版 Windows 7 への対応には，32bit 環境と 64bit 環境間の相違および Windows のバージョン間の変更を明らかにし，これらの対策を行う必要がある．

以下，本論文では，2 章で Alkanet の概要について述べ，3 章で 32bit 版 Windows XP と 64bit 版 Windows 7 間での変更について述べる．また，4 章で 64bit 版 Windows 7 に対応したシステムコールトレースの実装について述べ，5 章で作成したプロトタイプを用いた機能評価を述べ，6 章でまとめる．

2 Alkanet

Alkanet は仮想計算機モニタ (VMM) ベースのマルウェア動的解析システムである．Alkanet は，マルウェアが多く出回っている Windows のシステムコールをトレースすることができる．観測単位をシステムコールとすることで，マルウェアの挙動を機能単位で抽出し，挙動の理解を容易にする．取得したシステムコールトレースのログからは，ログ解析ツールを用いて特徴的な挙動を抽出したレポートを得ることができる．また，VMM ベースとすることで，多くのマルウェアに搭載されているアンチデバッグ機能を回避できるという特徴を有している．

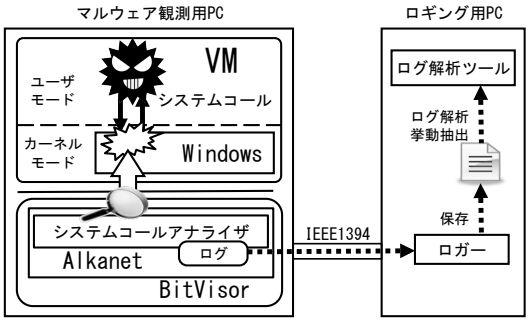

図 1: Alkanet の全体構成

Alkanet の全体構成を図 1 に示す．Alkanet は， VMM である BitVisor[3] の拡張機能として実装している．BitVisor は，ホスト OS を必要とせず，ハードウェア上で直接動作するハイパーバイザ型の VMM である．Intel 製 CPU の仮想化支援機能である Intel VT を利用しており，Windows を修正なしで実行できる．

マルウェアの実行環境であるゲスト OS には，32bit 版 Windows XP Service Pack 3 を用いている．この環境におけるシステムコールは，通常 sysenter 命令によってカーネルモードへ遷移し，sysexit 命令によってユーザモードへ復帰する．システムコールのフックは，これらの命令にハードウェアブレイクポイントを設定することで実現している．フック後に，レジスタやメモリ内容を解析することでシステムコールの種類や引数を取得し，ログに保存する．Alkanet のログは，ロギング用 PC から IEEE 1394 を用いて取得し，その後に解析することができる．

3 32bit 版 Windows XP と 64bit 版 Windows 7 間の変更点

32bit 版 Windows XP SP3 と，64bit 版 Windows 7 SP1 の間では，様々な変更が行われた．これらの変更の中には，Windows 内部の構造体や関数のマップ位置をランダムにするといった解析動作に大きく影響する変更も含まれており，従来の Alkanet で用いていた情報取得の手法を用いることができない．新たなマルウェア

動作環境において Alkanet が正常な情報取得を行うには，以下の変更への対処が必要である．

Windows のバージョン間の変更

ASLR (Address Space Layout Randomization)

32bit 環境と 64bit 環境における相違

KPCR(Kernel Processer Control Region) 構造体へのアクセス方法，呼出規約，システムコール呼び出しの流れ，ハンドルとハンドルテーブル

3.1 ASLR

ASLR は，Windows 内部の構造体や関数のマップ位置をランダマイズする機構である．Windows Vista より前のバージョンでは，Windows 内部の構造体や関数は固定の位置にマッピングされていた．しかし，マップ位置が一定の場合，固定のアドレスを用いた攻撃を受ける恐れがある．そのため，Windows Vista 以降は，セキュリティ強化のために ASLR を用いてマップ位置をランダム化している．

ASLR による Kernel および HAL (Hardware Abstraction Layer) のマップ位置のランダマイズは図 2 のように行われる [4]．Kernel と HAL は，隣接してマッピングされるため，それぞれの位置関係は前後の 2 通りである．また，Kernel と HAL とがマップされる開始位置は，ページサイズ 0x1000 を 1 スロットとして 32 スロットのうちいずれかとなる．よって，それぞれのマップ位置は 64 通りとなる．Kernel および HAL 内部でシンボルのランダマイズは行われないため，それぞれの内部でのオフセットは変化しない．

3.2 KPCR 構造体へのアクセス方法

Windows には，個々のプロセッサの状態を保存するデータ構造である PCR(Processer Control Region) と PRCB(PRocesser Control Block) がある．これらはそれぞれ KPCR 構造体と KPRCB 構造体で定義される．KPCR 構造体は KPRCB 構造体へのポインタを，KPRCB 構造体は

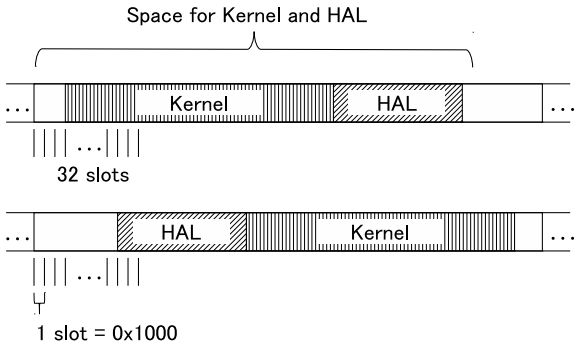

図 2: ASLR によるマップ位置のランダム化

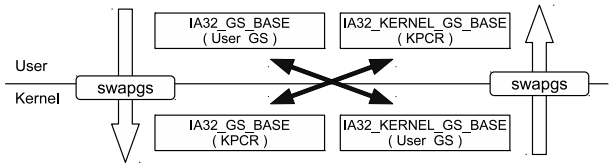

図 3: swapgs 命令による MSR の値の交換

動作中のスレッドの状態を持つスレッド構造体へのポインタをそれぞれメンバに持つ．

32bit 環境において，CPU0 の KPCR 構造体は，固定の位置 (0xffdff000) にマッピングされていた．CPU1 以降の KPCR 構造体については，FS レジスタに格納されたセレクタ値を基に先頭アドレスを算出することができた．一方，64bit 環境においては，CPU0 の KPCR 構造体が固定の位置にマッピングされることはなくなった．また，全ての CPU の KPCR 構造体において，仕様によりセグメントレジスタを用いたマップ位置の算出が不可能となった．一方で，64bit 環境では，MSR の `IA32_GS_BASE` に KPCR 構造体の先頭をアドレスを格納している．このレジスタへの書込みは，ブート時に行われる．

ただし，KPCR 構造体の先頭アドレスが常にこのレジスタに格納されているわけではない．64bit 環境において新たに追加された swapgs 命令により，ユーザモードとカーネルモードの処理を切り替える際に，図 3 のように MSR に格納された値の交換が行われる．そのため，KPCR 構造体の先頭アドレスを取得する際には，どちらのレジスタの値を参照するのが適切か判断する必要がある．

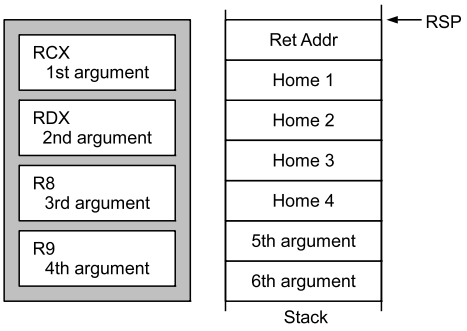

図 4: x64 呼出規約

3.3 呼出規約

呼出規約は，関数呼び出し時のデータの受け渡しなどに関する規約である．32bit 環境の Windows では，関数の引数を全てスタックに格納する stdcall 呼出規約を用いていた．一方，64bit 環境では，x64 呼出規約 [5] と呼ばれる新たな呼出規約を用いている．x64 呼出し規約では図 4 のように，関数の引数をレジスタとスタックに分けて格納する．第 1 引数から第 4 引数は，揮発性の RCX，RDX，R8，R9 レジスタに格納される．第 5 引数以降が格納されるスタックには，HOME 領域と呼ばれる第 1 引数から第 4 引数までの領域も確保され，必要があればこれらにも引数の値が格納される．また，関数の戻り値は RAX レジスタに格納される．

3.4 システムコール呼出しの流れ

32bit 環境と 64bit 環境とでは，システムコールの呼び出し手順が異なる．システムコールを呼び出す際，レジスタに呼出し対象を示す 14bit で構成された値を格納する．上位 2bit はシステムコールテーブルを選択するインデックスであり，下位 12bit はテーブル内インデックスとなっている．32bit 環境では値の格納先レジスタに EAX レジスタを用いる．これに対して，64bit 環境では RAX レジスタを用いる．

レジスタへ値をセットした後，32bit 環境では SystemCallStub と呼ばれるスタブを呼び出し，このスタブの中で EIP などの退避を行った後で sysenter 命令を実行する．sysenter 命令が実行されると，処理がユーザモードからカーネルモードへ移り，指定したシステムコールが実行される．対象のシステムコール実行後は，sysexit 命令を用いてカーネルモードからユーザモードへと処理を戻す．一方，64bit 環境では，sysenter 命令に代わり syscall 命令を用いる．syscall 命令は，SystemCallStub で行っていた RIP の退避などを内部で行う．そのため 64bit 環境ではスタブの呼出しを行わず直接 syscall 命令を実行する．対象のシステムコール実行後は，sysexit 命令ではなく sysret 命令を用いてカーネルモードからユーザモードへと処理を戻す．

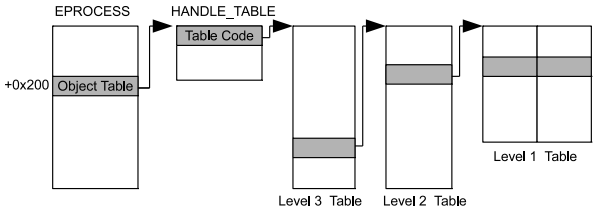

図 5: ハンドルテーブルの構造

3.5 ハンドルとハンドルテーブル

ハンドルとハンドルテーブルは，オブジェクトへのアクセスに用いられる．オブジェクトには，オブジェクトヘッダが付加され，これによってオブジェクトの参照回数や名前情報，セキュリティ情報などが管理されている．32bit 環境と 64bit 環境では，ハンドルおよびハンドルテーブルの構造が異なる．

3.5.1 ハンドルテーブル

ハンドルテーブルの構造は，図 5 のようになっている．オブジェクトの総数に応じて最大 3 階層のテーブルで構成され，最下位のテーブルにはオブジェクトへのポインタが格納されている．また，最下位テーブルには監査用のエントリが 1 つ含まれる．最上位テーブルは，古いバージョンとの互換性のためエントリ数が制限されており，最大 32 エントリとなっている．ページサイズとポインタのサイズより，32bit 環境では 2 段目のテーブルの最大エントリ数が 1024 エントリ，最下位のテーブルが 512 エントリである．64bit 環境では，2 段目のテーブルが 512 エントリ，最下位のテーブルが 256 エントリである．

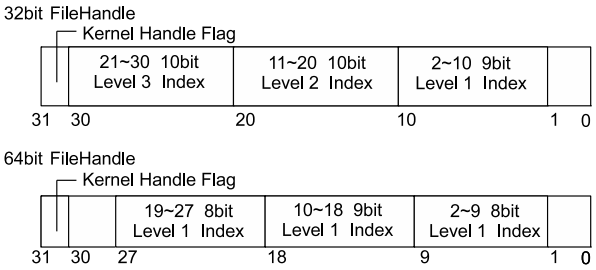

図 6: 32bit 環境と 64bit 環境のハンドル

3.5.2 ハンドル

ハンドルには，図 6 のように各ハンドルテーブルのインデックスが含まれる．最上位ビットはカーネルのハンドルテーブルであることを示すフラグであり，32bit 環境では，2〜10bit，11〜20bit，21〜30bit がそれぞれ最上位から最下位のハンドルテーブルのインデックスとなっている．64bit 環境では各ハンドルテーブルの最大エントリ数が減ったため，2〜9bit，10〜18bit，19〜27bit がそれぞれ最上位から最下位のハンドルテーブルのインデックスとなった．

4 新環境に対応したシステムコールトレースの実装

Alkanet では，マルウェアの挙動を解析するためにシステムコールトレースを行っている．システムコールを実行する前後でフックを行い，解析に必要な情報を取得する．

4.1 システムコールのフック

Alkanet は，カーネルモードへ処理が移る際と，ユーザモードに処理が戻る際の両方でフックを行う．64bit 版 Windows 7 では，syscall 命令によりカーネルモードへ入り，システムコール実行後は sysret 命令によりユーザモードへと戻る．図 7 のように，syscall 命令の直後に呼び出される KiSystemCall64 の先頭と，KiSystemServiceExit 内の sysret 命令の直前にハードウェアブレイクポイントを設置する．ハードウェアブレイクポイントを設置するには，ASLR の回避が不可欠である．KiSystemCall64 の先頭

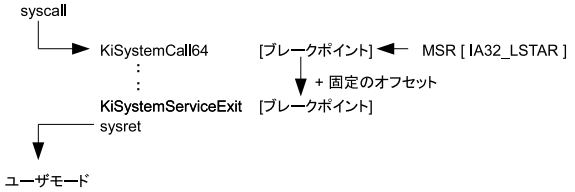

図 7: システムコールのフック

アドレスは，MSR の IA32_LSTAR に格納されているためこれを用いる．KiSystemServiceExit の先頭アドレスは，MSR に格納されていないため，オフセットを用いたアドレス算出を行う．ASLR によるランダマイズでは，カーネルや HAL 単位でマップ位置がランダム化されるが，カーネル内部のランダマイズは行われないため，内部の構造体や関数の相対位置は固定である．したがって，KiSystemCall64 の先頭アドレスと KiSystemServiceExit 内の sysret 命令の直前の差をオフセットとすることで，sysret 側のフックポイントのアドレスを算出することが可能である．

4.2 システムコール番号の取得

システムコール番号は，syscall 時と sysret 時にそれぞれ取得する．syscall 時は，RAX レジスタに格納された値からシステムコール番号を取得する．sysret 時は，RAX レジスタの値が書き換わっているためシステムコール番号の取得が行えない．そのため，戻り先アドレスを用いた取得を行う．システムコールの呼び出し元のスタブにおいて，syscall 命令の直前で RAX レジスタにシステムコール番号を示す値を格納している．したがって，この命令からシステムコール番号を抽出する．

4.3 プロセスとスレッドの特定

プロセスとスレッドには，ぞれぞれプロセス ID(Pid) とスレッド ID(Tid) が割り当てられている．スレッド構造体のメンバである Cid は，Pid と Tid で構成されており，この Cid を取得することで呼び出し元のプロセスとスレッドを特定することが可能である．また，プロセス構

造体には ImageFileName というフィールドが存在し，ここにプロセスのイメージ名が格納されている．これらは KPCR 構造体から辿って取得することができる．

4.4 システムコールの引数の取得

システムコールを呼び出す際に引数が渡されるが，その用途には入力と出力が存在する．すなわち，システムコール実行前に値が意味を持つものと，実行後に引数の値が確定するものとがある．よって，syscall と sysret 命令の両方で値を取得する必要がある．システムコールの引数は，呼出規約に準じてレジスタとスタックに格納される．ここで用いられるレジスタは揮発性であり，sysret 命令では既に他の値が上書きされているため，syscall と sysret 命令で異なる取得方法を用いる必要がある．

syscall 命令では，引数をレジスタおよびスタックから取得する．3.3 節で述べたように，x64 呼出規約では，第 1 引数から第 4 引数までは RCX，RDX，R8，R9 レジスタに，第 5 引数以降はスタックに値を格納すると定義している．しかし，syscall 命令は戻り先アドレスを保存するために RCX レジスタを使用する．そのため，syscall 命令を実行するスタブにて syscall 命令を実行する前に RCX レジスタの値を R10 レジスタへと退避させ，syscall 命令実行直後に呼び出される KiSystemCall64 の内部で R10 レジスタから RCX レジスタに復元される．syscall 時のフックポイントが設置されている KiSystemCall64 の先頭では，システムコールの第 1 引数は R10 レジスタに格納されているため，第 1 引数から第 4 引数までは R10，RDX，R8，R9 レジスタから，第 5 引数以降はスタックから値を取得する．

sysret 命令では，レジスタを参照する以外の方法でシステムコールの引数を取得する必要がある．入力時に値が不定な引数は，定数ではなくポインタとして渡される．そのため，引数保存用のテーブルを作成し，syscall 側でレジスタに格納される 4 つの引数を保存する．syscall と sysret 命令の対応づけには，Tid を用いる．Pid および Tid において id の重複は発生しないため，Tid のみでスレッドを一意に特定することが可能である．第 5 引数以降はスタックに格納されており，sysret 命令でも syscall 命令と同様に値を取得することが可能である．

4.5 戻り値の取得

システムコールの戻り値は，x64 呼出規約に準じて RAX レジスタへ格納される．sysret 側のフックポイントは sysret 命令の直前であり，この時点では RAX レジスタに戻り値が格納されている．そのため，RAX レジスタを参照することにより，システムコールの戻り値を取得することが可能である．

4.6 観測対象のシステムコール

Alkanet は，マルウェアの挙動を明らかにするために特定のシステムコールに着目し情報取得を行う．Alkanet は，Windows XP では合計 35 個のシステムコールを対象としている [1]．これに対して，Windows 7 では新たに追加された 3 個を加えた計 38 個のシステムコールを対象として情報の取得を行う．追加したシステムコールは，NtOpenKeyEx，NtCreateThreadEx，NtCreateUserProcess である．NtOpenKeyEx は，Windows 7 で追加されたシステムコールであり，NtOpenKey と比べてオープン時の動作を指定する OpenOption という値を引数にとる部分が異なる．Windows 7 x64 では，NtOpenKey と NtOpenKeyEx の両方が使用されていることを確認している．NtCreateThread および NtCreateUserProcess は，Windows Vista 以降に追加されたシステムコールであり，それぞれ NtCreateThread と NtCreateProcessEx に代わって使用されるようになった．NtCreateUserProcess は，プロセスのみを生成する NtCreateProcessEx とは異なり，プロセスと初期スレッドを生成することが可能である．Alkanet では，対象システムコールの引数に応じた追加情報の取得も行う．引数にポインタやハンドルが渡された場合には，そのポインタが指し示す先の値の取得や，対象のオブジェクトに含まれる情報を取得することで，より詳細な挙動の解析を実現する．

5 機能評価

5.1 評価目的と検体

マルウェアの挙動の解析に必要な情報の取得が行えることを検証するため，Windows 7 の 64bit 環境で発行されるシステムコールをトレースするプロトタイプを作成して評価を行った．本プロトタイプは，観測対象のシステムコールについて，システムコール番号や発行元のスレッド情報などを取得する．また，各システムコールに応じて引数を解析し追加情報の取得を行う．現在は，4.3 節で述べた観測対象のシステムコールのうちレジストリとネットワーク以外のシステムコールに関して追加情報を取得する機能の実装が完了している．

評価に用いる検体は，MWS Datasets 2015[6] に含まれる D3M2014 に活動が記録されているマルウェアを用いた．複数のアンチウイルスにおいて Fareit と検出された当該検体を Windows XP 上で実行し解析したところ，Temp フォルダに bat ファイルをドロップし，cmd.exe を起動して読み込ませる挙動が観測された．また，cmd.exe が，検体本体を削除する挙動が観測された．本評価では，Windows 7 x64 上で検体を実行し，作成したプロトタイプにおいて上記の挙動が観測できることを確認する．

5.2 ログエントリ

図 8, 図 9 は，前述の検体を malware.exe という名前で実行し，取得したログの一部である．なお，ログ番号と CID のみ 10 進数表記で，他は 16 進数表記である．ここでは，図 8 の NtCreateFile，NtCreateUserProcess，図 9 の NtSetInformationFile の 3 つのシステムコールについて，syscall 命令時と sysret 命令時に取得したログを示す．NtCreateFile システムコールはファイルを作成もしくは開く際に使用され，NtCreateUserProcess は，4 章で述べたとおりプロセス作成時に使用される．NtSetInformationFile は，ファイルオブジェクトに関する情報を変更するシステムコールであり，ファイルの削除を行う際にも用いられる．

それぞれのシステムコールに対して，基本情報の取得に加えて，引数を解析した追加情報を付与する．NtCreateFille は，syscall 命令時に第 3 引数の ObjectAttributers の ObjectName の値を取得して OA name として出力する．sysret 命令時は，第 1 引数より FileHandle を取得し，生成されたファイルオブジェクトからパスを取得して Path として出力する．NtCreateUserProcess は，第 9 引数の ProcessParameters に含まれる ImagePathName および CommandLine を取得する．また，sysret 命令時に第 1 引数の ProcessHandle と第 2 引数の ThreadHandle から，生成される PID と TID の値を取得する．NtSetInformationFile は，syscall 命令時に第 1 引数のファイルハンドルからファイルオブジェクトのパスを取得し Path として出力する．また，第 3 引数のポインタの先を参照することにより，書き換え内容のバッファである FileInformation を取得し，parg(3) として出力する．

5.3 評価結果

図 8(1)，(2) のログより，malware.exe が NtCreateFile システムコールによって Temp フォルダに 232004.bat を作成したことがわかる．続く (3)，(4) のログより，malware.exe が NtCreateUserProcess システムコールによって cmd.exe を起動したことがわかる．さらに，取得した CommandLine から 232004.bat を引数として与えられたことも確認できる．また，ここでは作成したプロセスおよびスレッドの PID および TID の値を取得しており，PID が 1476，TID が 1560 であることがわかる．

図 9 の (5) のログおよび (6) のログにおいて，cmd.exe の CID の値は 1476.1560 であり，malware.exe によって起動されたものであるとわかる．cmd.exe は，NtSetInformationFile システムコールによって malware.exe のファイルオブジェクトの情報を書き換えている．書き換えを行う項目は，第 5 引数の FileInformationClass の値で指定されており，書き換え内容は，第 3 引数のポインタの参照先である FileInformation に格納されている．parg(3) の値より FileInformation の値は 1 であり，第 5 引数の値が d であ

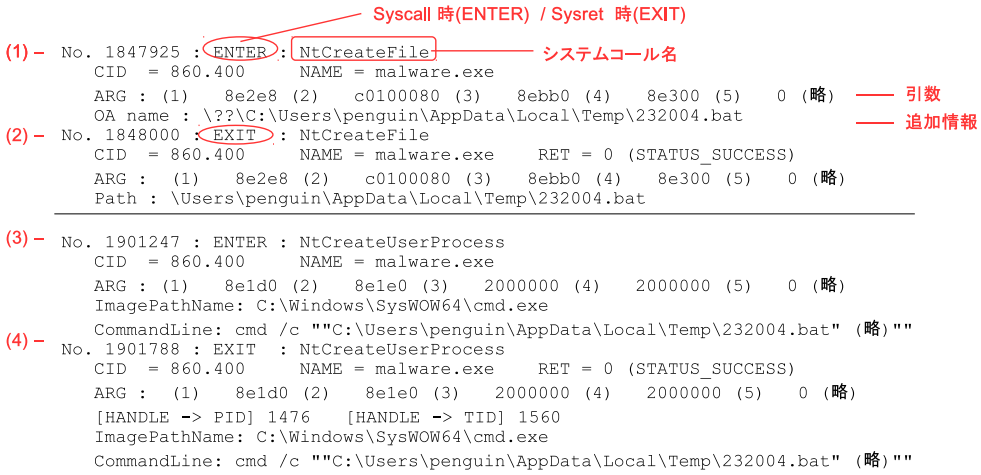

図 8: bat ファイルのドロップと cmd.exe の起動のログ

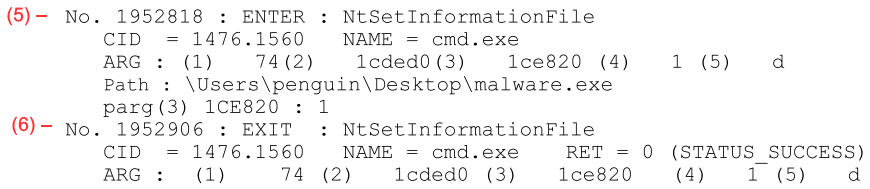

図 9: cmd.exe による検体の削除のログ

ることから，これに該当する FileDispositionInformation の値を 1 に書き換えたことがわかる．FileDispositionInformation の値を 1 に変更することは，そのファイルオブジェクトを削除することと等しい．したがって，cmd.exe によって malware.exe が削除されたことが確認できた．以上より，確認事項である検体の挙動を取得したログより Windows XP 版の Alkanet と同等の解析情報を取得することができた．よって，4 章で述べた手法を用いることで，Windows 7 64bit 環境においてマルウェアの挙動解析に必要な情報の取得が行うことが可能である．

6 おわりに

本論文では，Windows 7 x64 におけるシステムコールフックによるデータ取得法について述べた．32bit 環境と 64bit 環境における差異と Windows のバージョン間の変更を明らかにし，それらを対策することで Windows 7 x64 を対象とするシステムコールトレーサを実現した．今後の予定として，未対応なレジストリとネットワークに関するシステムコールの対応を行う．

参考文献

[1] 大月勇人，瀧本栄二，齋藤彰一，毛利公一：マルウェア観測のための仮想計算機モニタを用いたシステムコールトレース手法，情報処理学会論文誌，Vol. 55, No. 9, pp. 2034-2046 (2014).

[2] Net Applications: NetMarketShare Desktop Operating System Market Share July 2015, Net Applications.com, `https://netmarketshare.com` (accessed 2015-08-20)

[3] Shinagawa, T., Eiraku, H., Tanimoto, K., et al.: BitVisor: a thin hypervisor for enforcing i/o device security, Proc. VEE '09, ACM, pp.121-130 (2009).

[4] Ralf Hund, Carsten Willems and Thorsten Holz: Practi-cal Timing Side Channel Attacks Against Kernel Space ASLR, 2013 IEEE Symposium on Security and Privacy, pp. 191-205 (2013).

[5] Microsoft: x64 Software Conventions, Microsoft, `https://msdn.microsoft.com/en-us/library/7kcdt6fy.aspx` (accessed 2015-08-20)

[6] 神薗 雅紀，秋山 満昭，笠間 貴弘，村上 純一，畑田 充弘，寺田 真敏：マルウェア対策のための研究用データセット〜MWS Datasets 2015〜，研究報告コンピュータセキュリティ(CSEC), Vol.2015-CSEC-70, No.6, pp.1-8(2015).