仮想計算機モニタを用いたマルウェアの挙動解析

大月勇人† 瀧本栄二†
樫山武浩† 毛利公一†

1. 背景

近年, マルウェアの脅威が問題となっている. マルウェア対策には, マルウェアを解析し, どのような挙動をするかを調査する必要がある. しかし, マルウェアは, 新種や亜種が次々に出現するため, 1 体のマルウェアの解析に時間を費やすことができない. このような場合, 実際にマルウェアを実行し, その挙動を追跡することにより, 比較的短時間で解析できる動的解析が有効である. しかし, 最近のマルウェアの多くは, アンチデバッグと呼ばれる機能を持つ[1]. これは, マルウェア自身が動的解析されていることを検知し, 実行の停止や解析の妨害などを行うものである. アンチデバッグにはさまざまな手法があり, 全てを回避して解析することは一般に困難である. そこで, OS よりも下位のレイヤで動作する仮想計算機モニタ (VMM) を利用したマルウェア解析機構 Alkanet を開発している[2].

2. Alkanet

2.1 概要

Alkanet は VMM を用いたマルウェア動的解析システムである. VMM を用いることで, 多くのマルウェアに搭載されているアンチデバッグを回避できる. また, Alkanet は, Windows のシステムコールをトレースする. これにより, マルウェアの挙動を機能単位で抽出し, 挙動の理解容易性を実現している. そして, システムコールトレースのログをさらに分析し, マルウェアの特徴的な挙動を抽出したレポートを出力する.

2.2 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 を修正なしで実行することができる. なお, マルウェアの実行環境であるゲスト OS には, 32bit 版 Windows XP SP3 を用いている. この環境で発行されるシステムコールをフックし, その種類や引数を取得, ログに保存する.

† 立命館大学

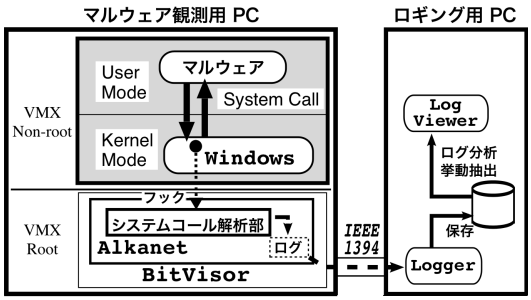

図 1 Alkanet の全体構成

Alkanet のログは, ロギング用 PC から IEEE 1394 を用いて取得する. IEEE 1394 は, 接続先デバイスの物理メモリを Direct Memory Access で読み書きできる. そのため, マルウェアに検知・妨害されずにログの取得が可能である. また, ログ取得後に, そのログの分析を行い, マルウェアの挙動を示すレポートを出力する.

3. 挙動解析に必要な情報

システムコールの発行元は, スレッドレベルで区別する必要がある. これは, Windows における実行単位が, スレッドであるためである. また, マルウェアによるコードインジェクションによって, 通常のプロセスの中にも「悪意あるスレッド」が存在する場合がある. したがって, システムコール発行元をスレッドレベルで区別するために, プロセス ID(PID), スレッド ID(TID), イメージ名を取得する. なお, Windows では PID と TID の組を Cid と呼ぶ.

さらに, マルウェアの挙動を調査するために, システムコールの引数と戻り値を取得する. しかし, 引数や戻り値には, ポインタや OS 固有のデータ構造が用いられることが多く, その値だけでは不十分な場合がある. したがって, これらのデータ構造を解釈し, 必要な情報を補う必要がある.

以上から, 具体的に以下の情報が必要である.

- システムコール発行元の Cid とイメージ名
- システムコール番号
- システムコールの引数と戻り値
- 固有のデータ構造に対する補足情報

4. システムコールのフック

4.1 システムコールフックの方法

32bit 版 Windows XP SP3 におけるシステムコールは, 通常 sysenter によってカーネルモードへ入り, sysexit によってユーザモードへ復帰する. 3 章で述べた情報を取得するために, sysenter と sysexit の両方でフックを行う. 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する.

sysenter 時の情報と sysexit 時の情報は, 両方揃って 1 つのシステムコールのログである. したがって, これらを対応付ける必要がある. しかし, この 2 つのログは必ずしも連続しない. そこで, それぞれのフックは個別に行い, ログ解析時にシステムコールの番号や発行元の情報などから対応付けを行う.

4.2 システムコールの特定

Windows では, システムコール発行時にシステムコールの番号を EAX に格納する. したがって, sysenter のフックでは, EAX からシステムコールの番号が取得できる. 一方で, sysexit のフックでは, どのシステムコールに対応したものかわからない. そこで, 通常システムコールは ntdll.dll に実装されているスタブを用いることを利用する. スタブは, 発行するシステムコールと同名のシンボルである. そのため, ユーザモードスタックに格納された戻りアドレスを調べることで, 発行されたシステムコールが特定できる.

4.3 引数と戻り値の取得

Windows の API では, 戻り値は EAX に, 引数はスタックに格納される. したがって, 戻り値は EAX, 引数はユーザモード時のスタックから取得できる. Windows のシステムコールでは, 発行時に ESP の値を EDX に格納する. よって, sysenter のフックの際, スタックの位置は EDX の値からわかる. また, sysexit は, 実行時に ECX の値を ESP にロードする. したがって, スタックの位置は ECX から取得できる. これにより, スタックから引数の取得が可能となる. このとき, 必要なら固有のデータ構造に対する補足情報の取得も行う.

5. 評　　価

Alkanet を用いてマルウェアの解析を行った. なお, ネットワークには接続していない. 検体には, CCC DATAset 2011[4] で活動が記録されているマルウェアを用いた. ここでは, Polipos.exe と呼称する.

図 2 に Polipos.exe を実行して得たシステムコールのログの一部を示す. 図 2 では, Polipos.exe が, NtCreateThread を発行し, 別のプロセスの explorer.exe に対してスレッドを作成している. これはコードインジェクションの挙動である. sysexit のログから, explorer.exe 内に TID 1e8 のスレッドが作成されたことがわかる. この後, このスレッドは Polipos.exe を複製する挙動やネットワークへの接続を試みる挙動などを示した. このように, マルウェアが正常なプロセスに対して, コードインジェクションを行った場合でも, 通常のスレッドを区別し, 悪意あるスレッドを正確に追跡できる.

```
No. :    6339
Time:    689820849
Type:    sysenter
Ret :    - (-)
SNo.:    35 (NtCreateThread)
Cid :    bc.304
Name:    Polipos.exe
Note:    PID: b0, ProcessName: explorer.exe

No. :    6340
Time:    689820959
Type:    sysexit
Ret :    0 (STATUS_SUCCESS)
SNo.:    35 (NtCreateThread)
Cid :    bc.304
Name:    Polipos.exe
Note:    Cid: b0.1e8, ProcessName: explorer.exe
```

図 2　explorer.exe へのコードインジェクション

6. ま と め

本稿では, 仮想計算機モニタを用いて, マルウェアの動的解析を実現する “Alkanet” について述べた. Alkanet は, システムコールのトレースによりマルウェアを解析する. 実際にマルウェアの解析を行うと, マルウェアから派生する悪意あるスレッドも含めて正確に追跡が可能であった. 今後の課題として, ネットワーク通信の解析機能やログ解析ツールの充実が挙げられる.

参 考 文 献

1) Falliere, N.: Windows Anti-Debug Reference, `http://www.symantec.com/connect/articles/windows-anti-debug-reference` (2007).
2) 大月 他: マルウェア挙動解析のためのシステムコール実行結果取得法, コンピュータセキュリティシンポジウム 2011 論文集, Vol. 2011, No. 3, pp. 95–100 (2011).
3) Shinagawa, T. at el.: BitVisor: a thin hypervisor for enforcing i/o device security, *In Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, Washington, DC, USA, ACM, pp. 121–130 (2009).
4) 畑田 他: マルウェア対策のための研究用データセット 〜 MWS 2011 Datasets 〜, コンピュータセキュリティシンポジウム 2011 論文集, Vol.2011, No. 3, pp. 1–5 (2011).