統合仮想化システムモニタを用いたマルウェアのプロファイリング

†安藤類央　　‡高橋 一志　　†田端利宏　　‡須崎 有康

†情報通信研究機構
〒 184-8795 東京都小金井市貫井北町 4-2-1

‡東京大学 大学院 情報理工学系研究科
〒 113-8656　東京都文京区本郷 7-3-1

†岡山大学大学院自然科学研究科
〒 700-8530 岡山県岡山市北区津島中 3-1-1

‡産業技術総合研究所
〒 101-0021 東京都千代田区外神田 1-18-13

あらまし　本論文では、統合仮想化システムを用いたマルウェアのプロファイリング手法を提案する。提案システムでは、Windows OS 上でのマルウェアの挙動の各種リソースアクセスを、メモリ、ソケット、レジストリ、ファイルと統合的かつ高粒度に取得することが可能であり、ＡＰＩのインターセプトを行い、高粒度なログを定量的に取得することができる。また、仮想化マシンモニタを用いてシステムを観測することが可能なため、観測防御対象システムのシステムリソース利用やパフォーマンスに影響を与えることなく、マルウェアのプロファイルを行うことができる。評価実験では、Windows OS 上でのマルウェアのプロファイリング例を示す。

A profiling method of malware's behavior using integrated virtualized system monitor

†Ando Ruo　　‡Takahashi Kazushi　　†Tabata Toshihiro　　‡Kuniyasu Suzaki

†National Institute of Information and Communications Technology,
4-2-1 Nukui-Kitamachi, Koganei, Tokyo 184-8795 Japan
‡Graduate School of Information Science and Technology, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
†Graduate School of Natural Science and Technology, Okayama University
3-1-1 Tsushima-naka, Kita-ku, Okayama 700-8530 JAPAN
‡11F Akihabara Dai Bldg., 1-18-13 Sotokanda Chiyoda-ku,
Tokyo 101-0021 Japan

Abstract　In this paper a profiling method of malware's behavior using integrated virtualized system monitor. Our monitor is API hook based which enables fine-grained inspection of resource accesses such as file, memory, socket and registry on Windows OS. In proposed system virtualization technology is applied to monitor guest VM without impacting its performance and utilization. In experiment we show some examples of profiling of malware.

1　はじめに

最近のデバッグ技術、仮想化技術の発展は、ＯＳの状態のより詳細な観測と、高粒度なロギングが可能にした。デバッグ環境では、Microsoft Windows が提供するデバッギングＡＰＩ、ネットワークアプリケーション、そしてカーネルモジュールの開発のための環境も急速に整備されつつある。仮想化環境では、仮想マシン観測の

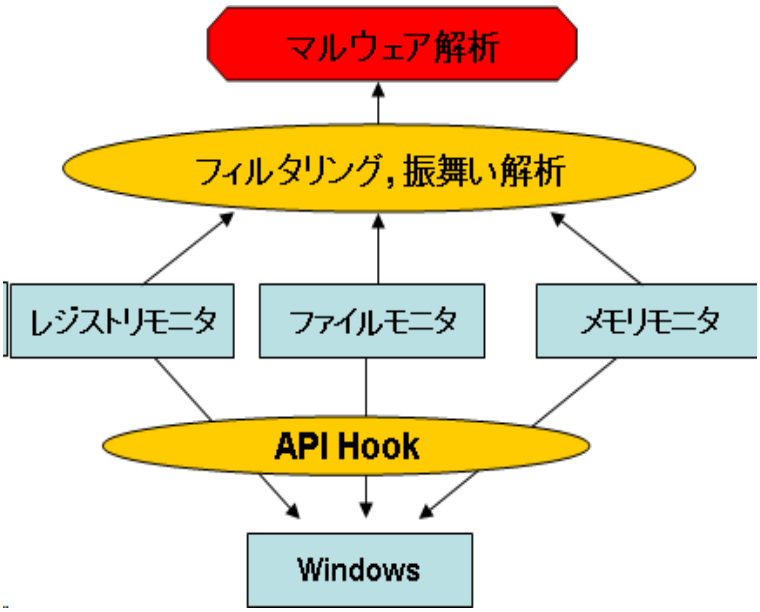

図 1: 統合システムモニタ。ＤＬＬ修正、フィルタドライバ、SysEnter、分岐命令フックなどを用いてＯＳの各種リソースアクセス（レジストリ、ファイル、メモリ、ソケット）をインターセプトし、システムの状態を観測する。

ためのＡＰＩやスナップショットの解析ソフトウェア、ホストＯＳとゲストＯＳのインターフェイスなどが急速に整備されつつある。本論文では、これらのソフトウェアを駆使して構築した統合仮想化システムモニタを用いたマルウェアのプロファイリングを提案する。

2　提案システム

2.1　Windows OS 上の統合システムモニタ

本論文で提案システム統合システムモニタは、DLL Inejection やフィルタドライバ、システムコール書き換えなどによるＡＰＩフックにより、Microsoft から提供されているモニタツールよりも高粒度なモニタを行うことを可能にするものである。図１は、統合モニタの概略を示したものであり、Windows ＯＳで提供されているデバッグとフィルタリングのための機構を利用して、各種リソース（メモリ、ソケット、ファイル、レジストリ）のアクセスをインターセプトし、ログに記載する。また、分岐命令の実行やハードウェア割り込みなどのフックは、仮想マシン側で修正を行うことで対応する。

2.2　仮想マシンモニタの修正

仮想マシンモニタにはいくつかの種類があるが、ＸＥＮのようにハイパーバイザーを自前で構築するもの、ＫＶＮのようにホストＯＳ（ＬＩＮＵＸ）内部に構築するもの、そして hyper-Visor のように I/O のフックによる準仮想化方式を採用するものなどがある。本論文では、ＫＶＭとＸＥＮの修正について述べる。

2.3　仮想マシンモニタタイプ１の修正

仮想マシンモニタタイプ１は、ホスト OS をハイパーバイザーとして用いるもので、このタイプには KVM がある。この方式では、ホスト OS のカーネル空間内にあるゲスト OS の仮想メモリを、ユーザ空間あるいはカーネル空間に移動する。この方式では、QEMU のインター

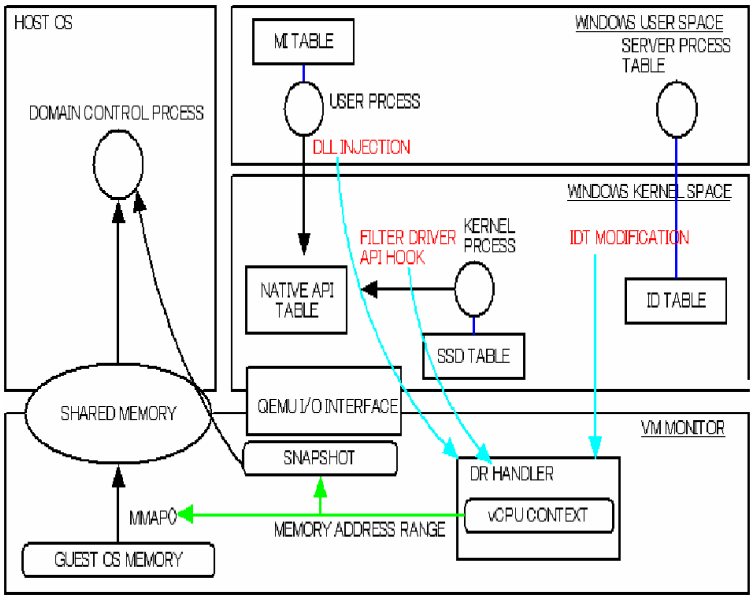

図 2: 仮想マシンモニタタイプ１での観測系の構築。各種テーブルを修正しイベントの情報を仮想マシンモニタに通知する。

フェイスを使う。ＫＶＭはLinuxカーネルに取り込まれているので、管理ＯＳで最新機能が使えるが、専用のＡＰＩが容易されておらず、まだ安定していない。そのため、提案システムの実装にはデバッグハンドラを用いた値の転送や、共有メモリを用いた文字列の転送をなどを適用した。

2.4 仮想マシンモニタタイプ２の修正その１

仮想マシンモニタタイプ２は、独自にブートローダとハイパーバイザーを用意するもので、このタイプには、XENがある。この場合、準仮想化と完全仮想化に分かれるが、本論文では、Windowsの完全仮想化を扱っているため、スナップショットの取得には、XENのインターフェイスではなく、QEMUのインターフェイスを用いることになる。図３は、ＸＥＮを用いた提案システムの実装方法の概略を示したものである。ＫＶＭと同様に、ＶＣＰＵのレジスタ経由でＡＰＩの引数やＡＳＣＩＩコードなどを受け渡す方法がある。

2.5 仮想マシンモニタタイプ２の修正その２

ＸＥＮの特徴として、ＡＰＩが整備されていることがある。そのため、ＡＰＩを駆使することでメモリのスナップショットからWindowsの状態の解析を行ったり、ログ情報を転送することができる。また、ハードウェア割り込みをWindows OSに先んじて捕捉することが可能である。図４は、これらのＸＥＮ系ＡＰＩを用いたWindows観測系の概略を示したものである。この方式は、仮想マシンモニタタイプ２の修正その１と比べて、観測にWindows OSの修正が少なくて済むという利点がある。

3 Windows OSの修正

3.1 フィルタドライバによるＡＰＩフック

Microsoft Windowsのフィルタドライバは、ＸＰから積極的に導入、活用が始まったソフトウェアモジュールであり、ＩＯマネージャとカーネルドライバの間に位置して、ファンクションド

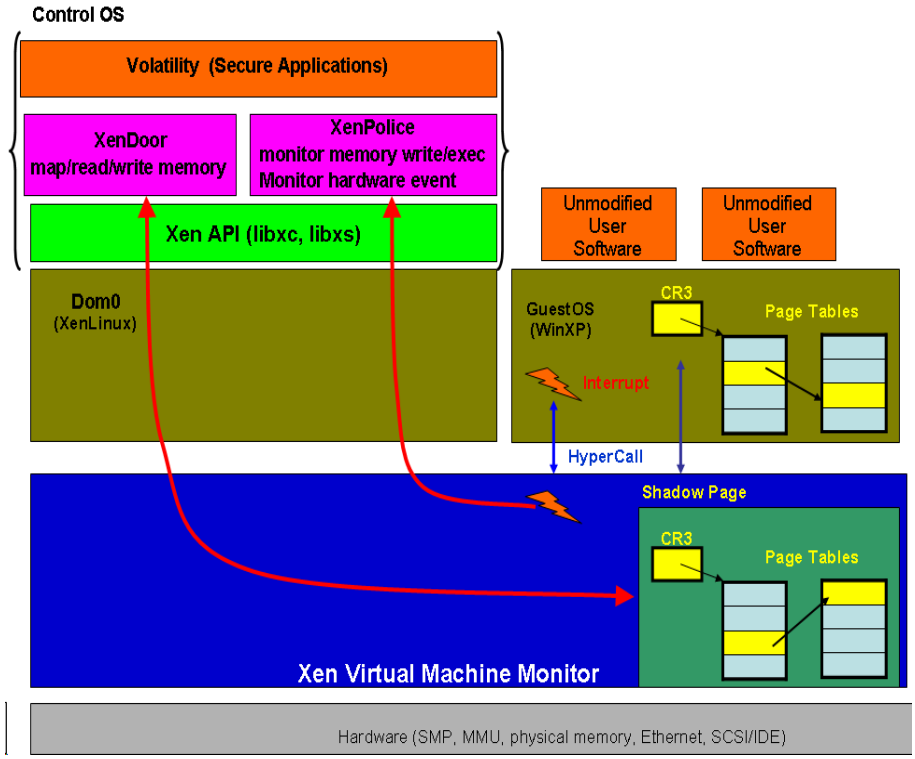

図 3: ＸＥＮ ＡＰＩを利用したホストＯＳの観測。同手法では Windows OS の修正が最低限で済み、ハードウェア割り込みなどもホストＯＳに先んじてフックできる。

ライバ（既存のデバイスドライバ）として機能する。フィルタドライバを用いることで、Native API フックを行うことができる。

3.2 DLL Injection

DLL Injection とは、任意の関数をライブラリ化して特定の API が発行されたときに、実行されるものであり、デバッグ時にＡＰＩの引数を参照したい時に用いられる。関数形は以下である。

```
void ReplaceIATTableInP2Psoftware
("kernel32.dll",
funcORG,
funcINSERT",
moduleHandler);
```

3.3 フィルタマネージャによるＡＰＩフック

フィルタマネージャは、Windows XP SP2 から導入されたファンクションドライバのための機構で、ファイルＩＯのフックの安定化などに用いられる。図３は、フィルタマネージャの機能概要を示したものである。フィルタマネージャは、カーネルドライバファンクションドライバとフィルタドライバの間でフィルタリングを行い、ドバイバの実装と稼動を簡素化する。

4 評価実験

評価実験では、統合システムモニタを用いて提供されたマルウェア検体のログデータの定量化を行った。表１は、プロファイル時に、インターセプトしたＡＰＩのリストである。図５は、実験結果の一例を示した。評価実験によって、

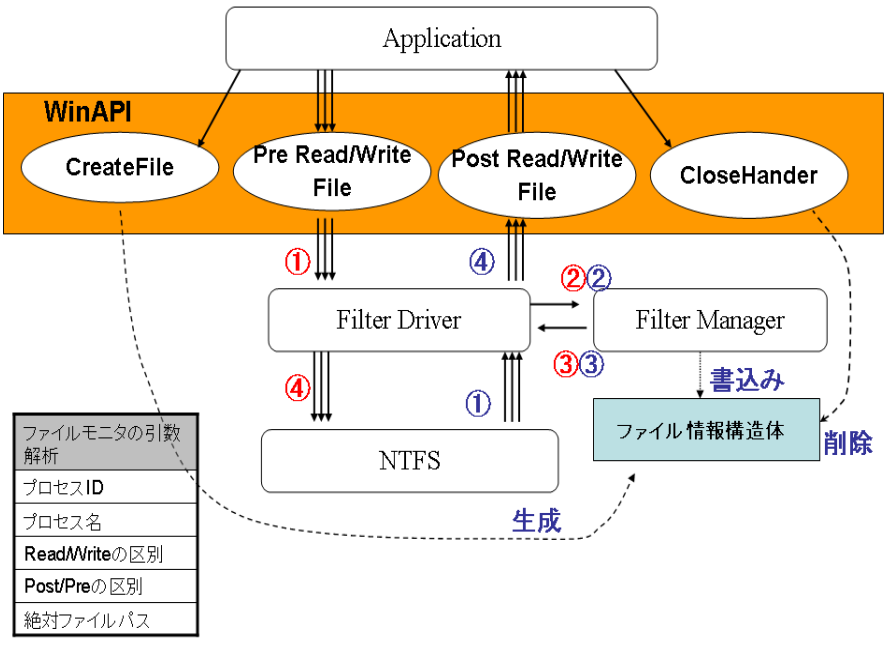

図 4: ファイルＩＯのイベントを検出、修正するためのフィルタマネージャの機構の概略。

感染動作でも定量化によっては検体間に大幅な違いがあることが明らかになった。ここで提案した定量化の手法は検体の分類などに有効であると考えられる。

5　まとめと今後の課題

本論文では、統合仮想化システムを用いたマルウェアのプロファイリング手法を提案した。提案システムでは、Windows OS 上でのマルウェアの挙動の各種リソースアクセスをメモリ、ソケット、レジストリ、ファイルと統合的かつ高粒度に取得することが可能であり、ＡＰＩのインターセプトを行い、高粒度なログを定量的に取得することができる。また、仮想化マシンモニタを用いてシステムを観測することが可能なため、観測防御対象システムのシステムリソース利用やパフォーマンスに影響を与えることなく、マルウェアのプロファイルを行うことができる。評価実験では、Windows OS 上でのマルウェアのプロファイリング例を示した。

参考文献

[1] Kernel Based Virtual Machine, http://kvm.qumranet.com/kvmwiki

[2] XEN virtual machine monitor, http://www.cl.cam.ac.uk/Research/

[3] Tal Garfinkel and Mendel Rosenblum, ”A Virtual Machine Introspection Based Architecture for Intrusion Detection ”, In the Internet Society ’s 2003 Symposium on Network and Distributed System Security (NDSS), pages 191.206, February 2003.

[4] 安藤類央、Nguyen Anh Quynh、須崎有康、「Windows のメモリ挙動モニタと Libvirt によるゼロデイ攻撃の検出システムの構築」, 暗号と情報セキュリティシンポジウム（SCIS2009）2009 年 1 月 21 日

API name	API type A	API type B
AllocVirtualMemory	Native API	Memory allocation.
CreateSection	Native API	Memory section creation .
LoadLibrary	User API	Invoking library.
MapViewOfSection	Native API	Memory Allocation.
PreRead	filter function	File Access.
PreWrite	filter function	File Access.
ZwCreateEvent	Native API	Memory Allocation.
ZwOpenKeys	Native API	Register Access.
CreateProcess	User API	Process Operation.
ZwOpenKeys	Native API	Registry Access.
WSARecv	User API	Socket Access.
Send	User API	Socket Access.

表 1: 評価実験でフックしたＡＰＩのリスト

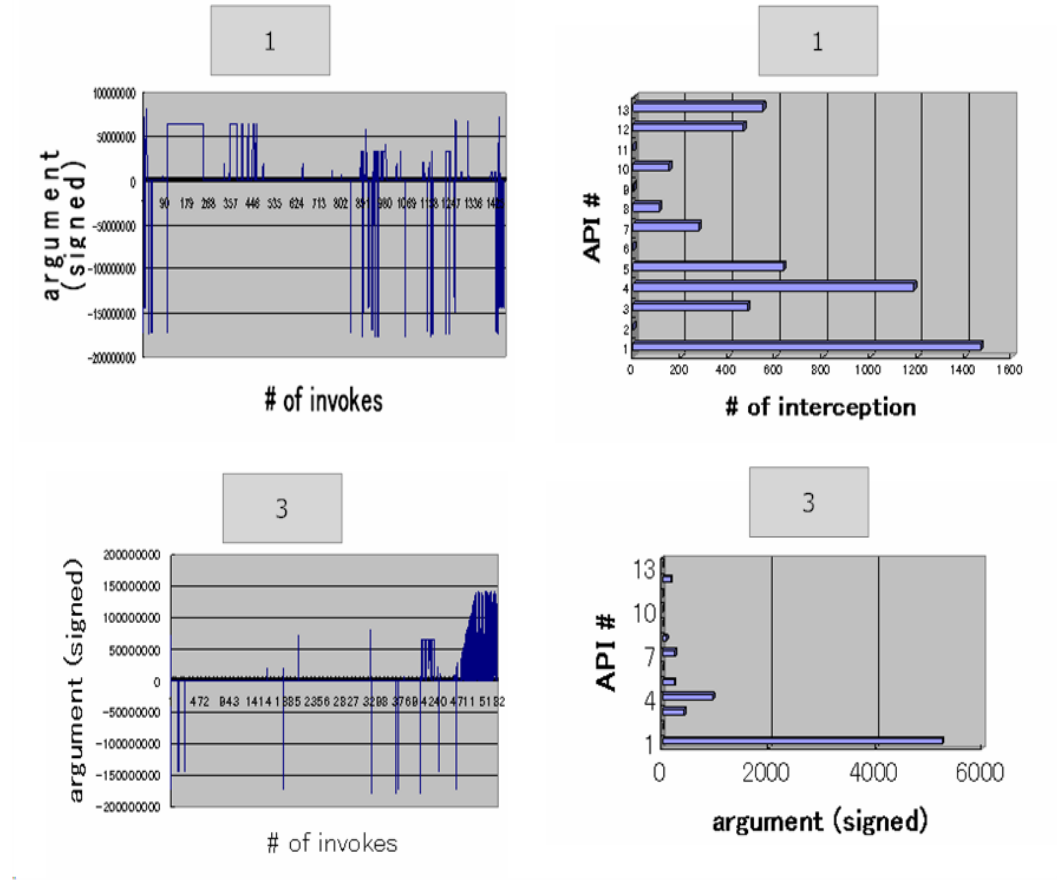

図 5: 統合システムモニタによるマルウェアのプロファイル結果の抜粋