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

2008－OS－107 (1)
2008／1／30

仮想機械技術を利用したキーロガー検知システム

小林卓嗣　　山田 浩史　　河野 健二

慶應義塾大学 理工学部 情報工学科
E-mail: {iruka, yamada}@sslab.ics.keio.ac.jp, kono@ics.keio.ac.jp

キーロガーがコンピュータセキュリティ上の脅威となっている．しかし既存のシグネチャを利用した検知手法には多くの欠点がある．本稿ではキーロガーによるキーボード入力の取得とログの出力というふるまいを仮想機械技術を利用して検知する手法を提案する．仮想機械上のオペレーティングシステムに大量のキーボード入力を行ったときのディスクとネットワーク出力の変化からキーロガーの存在の有無を判断する．提案手法を評価するために 56 種類の実際のキーロガーと 8 種類のキーボードユーティリティに対して実験を行った．その結果 55 種類のキーロガーを検知し，false positive は無かった．

A Keylogger Detection System based on Virtual Machine Technology

Takashi Kobayashi　　Hiroshi Yamada　　Kenji Kono

Department of Information and Computer Science, Keio University
E-mail: {iruka, yamada}@sslab.ics.keio.ac.jp, kono@ics.keio.ac.jp

Keyloggers have been one of threats to our computer systems. Because existing keylogger detection techniques based on signature-matching has some drawbacks, current intelligent keyloggers can circumvent the techniques and cannot be detected easily. This paper presents a keylogger detection technique, which exploits typical behavior of keyloggers; after capturing users' keyboard inputs, keyloggers preserve them into disk drives. Our technique sends a large number of keyboard inputs to compromised OSes from the virtual machine monitor, monitoring an amount of disk outputs and network outputs issued by the OSes. We determine whether the keyloggers are in the OSes or not by checking an amount of their outputs. To confirm that our technique works well, we conducted some experiments with 56 keyloggers and 8 keyboard utilities. From the experiments, our technique detected 55 keyloggers with no false positive.

1 はじめに

今日，ネットワークセキュリティ上の新たな脅威として，スパイウェアが問題となってきている [1]．スパイウェアとはシステムに損害を与えることや他のシステムに感染することを目的とするのではなく，ユーザのふるまいの取得や個人情報の漏洩を目的とした悪意あるプログラムの総称である．

スパイウェアは 2 つの特徴を持つ．1 点は積極的に対象システムの情報収集を行うことであり，もう 1 点はスパイウェアの存在をシステムのユーザから隠蔽することである．前者の特徴はデスクトップのスクリーンショット，キーストローク，動作しているアプリケーション，ウェブページの履歴などの情報を収集し外部に送信するなどの機能を持つことである．後者はルートキット技術 [2] を利用してレジストリエントリ，ファイル，動作中のプロセスなどの情報を隠す機能や防御システムを無効にする機能を持つことである．

既存のスパイウェア対策では主にシグネチャという個々のスパイウェアを特徴付けるデータパターンとバイナリイメージとの比較を行うことでスパイウェアの検知と削除を行っている．このようなシグネチャは既知のスパイウェアのサンプルから生成されるため，新しいスパイウェアが発見されるたびにシグネチャのアップデートが必要になる．よって新しいスパイウェアの発生からシグネチャの公開までに遅延がでることになる．また多くのスパイウェア検知システムはユーザモードの権限で動作しているため，スパイウェアが検知システムの動作を停止させたり，スパイウェアを高度に隠蔽することで検知できないようにする可能性がある．

このようなスパイウェアの中で特にキーボード入力のログを収集する機能を持つスパイウェアがキーロガーである．キーロガーはユーザに隠れてそのキーボード入力を記録し，キーロガーの作成者の元に送信するなどの動作を行う．そのため，ユーザが入力したクレジットカード番号やパスワードが外部に洩れ，多大な損害を与える可能性がある．

本稿ではキーロガーが対象のオペレーティングシ

ステム (OS) 上で動作しているかどうかの検証を自動的に行うキーロガー検知システムを提案する．本稿の提案システムでは未知のキーロガーも検知するためにシグネチャではなくキーロガーのふるまいを検知に利用する．キーロガーによるキーボード入力のディスクへの記録やネットワークへの送信などを特徴的なキーロガーのふるまいをとして定義し，これらのふるまいを検知することによってキーロガーの存在の有無を判断する．通常はこのようなふるまいは極めてわずかなものであるため，キーロガーの検知に用いることは非常に困難である．よって提案システムでは仮想機械技術を利用してキーロガーに大量のキーボード入力を与えることによって，意図的にこのふるまいがより明確になるようにすることでキーロガーの検知を可能にする．またキーロガーのふるまいの把握に必要な情報として OS が提供する情報を利用せずに仮想機械が提供する情報を利用することで既存のルートキット技術によるキーロガーの隠蔽を困難にする．

提案システムの有効性を示すために 56 種類のキーロガーとキーボード入力を取得するという点でキーロガーとふるまいの似ている 8 種類のキーボードユーティリティをインストールした OS に対して提案システムを動作させたときの検知精度を計る実験を行った．その結果 55 種類のキーロガーを検知し，false positive は無かった．

2 キーロガー

キーロガーは多くの場合，トロイの木馬や，商用あるいはフリーのソフトウェアの機能のひとつとして存在している．そのためユーザはキーロガーの機能が隠されているとは知らずにプログラムのインストールを行ってしまうことがある．一旦インストールしてしまうとキーロガーはユーザに気づかれること無くキーボード入力の収集を行えるように動作するため，ユーザがその存在に気づくことは困難である．特にキーロガー自体は非常に小さなプログラムとして実現可能なため通常はメモリ使用量，ディスク使用量， CPU 使用率などの変化から痕跡を探ることは非常に困難である．またさまざまなキーロガーの隠蔽手法によって検知システムを回避することも可能である．

2.1 キーボード入力の取得方法

キーロガーによるキーボード入力の取得方法は実行環境に依存するためさまざまな方法がある．Windows 環境でのキーボード入力の取得方法として Windows API を利用する方法がある．1) GetAsyncKeyState は呼出し時に指定したのキーの状態を取得する関数である．この関数を利用するキーロガーは反復的に GetAsyncKeyState を呼び出すことによって自身のプロセス以外への入力を含めた全てのキーボード入力を取得することができる．2) SetWindowsHookEx はグローバルフックを利用して全てのプロセスへのメッセージを取得できる関数である．この関数を利用したキーロガーは全てのプロセスへのキーボード入力に関するメッセージをフックすることで全てのキーボード入力を取得することができる．これらの方法以外にもフィルタドライバを利用してデバイスドライバにキーロガーの機能をもたせる方法がある．フィルタドライバを利用することで全てのキーボード入力を取得できるとともに，カーネルモードで動作するため OS レベルの権限で実行することができる．

2.2 キーロガーの隠蔽

キーロガーがユーザや検知システムに発見されないようにするために，自身の痕跡の隠蔽を行うさまざまな方法がある．多くのキーロガーはユーザからは見えないバックグラウンドで動作する．またルートキット技術を利用してプロセス情報の隠蔽，実行ファイルやログファイルの不可視設定，ログの暗号化，実行ファイルの obfuscation などを行う場合もある．Windows 環境の場合，Windows タスクマネージャのアプリケーションリストやプロセスリストからの除外や強制的なキーロガーに関係するファイルの隠しファイル設定などがある．このような場合では一般のユーザがキーロガーの存在に気づくことは非常に困難である．またキーロガーが管理者権限でインストールされていたり OS レベルの権限で動作している場合では，OS にインストールされている検知システムの停止や OS の機能を改変することによる特定ファイルの隠蔽による検知システムの回避が可能である．特に OS 自体をのっとるようなキーロガーの場合，キーロガーの存在を完全に隠蔽できるように OS がユーザや検知システムに偽の情

報を提供することが可能である．このように高度な隠蔽機能を持ったキーロガーがインストールされてしまった場合，それらを検知することは非常に困難である．

3 仮想機械技術を用いた検知手法

本稿では仮想機械技術を用いてキーロガーのふるまいを検知する手法を提案する．仮想機械 (VM) は仮想的なマシン環境でありその上で OS を動作させることができる．仮想機械モニタ (VMM) は VM を提供するソフトウェアレイヤである．VMM は VM 上の OS を他の環境から隔離するとともに，その実行状態を管理できるなどのセキュリティ上有用な特性を持っている [3]．

提案手法では VM 上の OS 内に存在するキーロガーを検知するために VMM を利用して意図的に通常のユーザでは不可能なほど大量のキーボード入力を与える．そして大量のキーボード入力に対する資源使用量の変化からキーロガーの有無を判断する．通常のユーザによるキーボード入力を行った場合ではキーロガーのログの出力による資源使用量の変化は極めてわずかである．そのため通常のキーボード入力に対する資源使用量の変化をそのままキーロガーの検知に利用することは非常に困難である．しかし提案手法では VMM を利用して大量のキーボード入力をキーロガーに与えることで意図的に大量にログを出力させることを可能にする．提案手法によるキーボード入力に対する資源使用量の変化をキーロガーのふるまいとしてみなすことで，資源使用量の変化がキーボード入力を与えたことに起因していることがわかればキーロガーが存在している可能性が高いといえる．逆にキーボード入力を与えたことに対してこれらの資源使用量に変化がなければキーロガーが存在する可能性は少ないといえる．提案手法では資源使用量の変化を把握するためにハードディスクへの出力とネットワークへの出力を監視する．

提案手法では図 1 のように検証対象の OS を VM 上で動作させ，検知システムは VMM の持つ仮想デバイスを利用して大量のキーボード入力とディスクやネットワーク出力の監視を行う．仮想デバイスは OS からは物理デバイスとして認識され，VMM は OS によるデバイスへの全ての入出力を把握することができる．検知システムが OS 上から入出力情報を得る場合，キーロガーが OS の制御をのっとり偽の入出力情報を提供することで検知を回避することができる．しかし提案手法の検知システムは OS の外部で動作し，VMM が提供する情報のみを利用する．よってキーロガーが入出力情報の隠蔽や検知システムを停止するには VMM をのっとらなければならない．このようなことは仮想機械技術によって隔離された環境では非常に困難である．

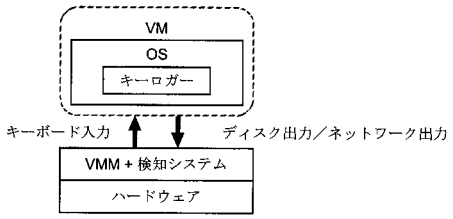

図 1: 仮想機械技術を利用した検知手法

4 システムのデザイン

4.1 システムの概観

提案システムでは，ある環境で既にキーロガーが動作しているかどうかを検証するのでは無く，あるプログラムがキーロガーの機能を有しているかどうかを検証する．そのために検証したいプログラムをあらかじめ用意した環境にインストールした上で，提案システムによってキーロガーのふるまいを検知することでその環境内にキーロガーが存在しているかを判断する．そのためにあらかじめ用意したキーロガーのインストールされていない環境でキーボード入力を行った場合の出力結果とその環境に検証したいプログラムをインストールした環境で同様の入力を行った場合の出力結果とを比較する．あらかじめ用意した環境を基準環境と呼び，検証したいプログラムをインストールした環境を評価環境と呼ぶ (図 2)．基準環境はキーロガーが存在していないこと以外は評価環境とほぼ同一である．例えばハードウェアの状態，OS，検証したいプログラム以外で既にインストールされたソフトウェア，動作中のプログラムなどである．この基準環境を評価環境にとって本来あるべき正常な状態とみなし，提案システムの実行によって得られたそれぞれの出力の比較によってキーロガーの有無を判断する．

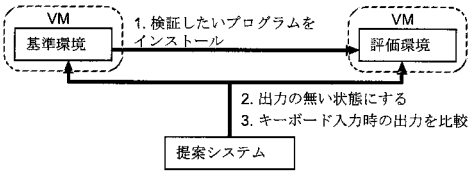

図 2: 提案システムの検知手順

4.2 キーボード入力の増幅手法

VM 上の OS に大量のキーボード入力を与えるために VMM を利用して VM に大量のキーボード割り込みを起こさせる．VMM は VM に対する全ての割り込みの通知を管理しているため VMM に仮想的なキーボード割り込みを起こさせることで VM 上の OS に対して任意のキーボード入力を与えることができる．これによって手動による入力に比べて数十倍以上の速さでの入力が期待できる．また VM 上の OS からは通常のキーボード入力として処理されるためキーロガーのキーボード入力取得方法に依存せずにキーボード入力を与えることができる．

GetAsyncKeyState を利用したキーロガーは数ミリ秒の間隔で取得したいキーごとにこの関数の呼出しを行っている．このキーロガーがある時間内に取得できるキーボード入力数の最大値は与えられたキーボード割り込みの数ではなく，GetAsyncKeyState の呼び出し間隔によって制限される．提案システムではこの制限を越えるために VMM を利用して VM に通常より多くのタイマ割り込みを起こさせる．タイマ割り込みは OS の内部時刻を更新するために定期的に送られる割り込みである．タイマ割り込みの間隔を短縮して内部時刻の更新を早めることで GetAsyncKeyState の呼出し間隔を意図的に短縮する．これによってより多くのキーボード入力を取得させることを可能にする．

4.3 環境の構築

提案システムにとって理想的な検査対象の環境とはキーボード入力を行わないときは出力が無く，キーボード入力を行ったときにはキーロガーが存在するときのみ出力があるという環境である．しかしディスクやネットワーク出力はアプリケーションを実行していなくてもバックグラウンドで動作しているデーモンプログラムや OS によるディスクへの遅延書き込みなどによってキーボード入力の有無に関係なく非同期に出力が行われることがある．またキーロガーが存在していなくても環境によってキーボード入力による副次的な作用がなんらかの形で出力に影響を与える可能性がある．このような出力とキーロガーによる出力の量に明らかな差が無い場合，計測した出力がキーボード入力に起因するものかを判断することは困難である．そのため提案システムでは基本的にディスクやネットワークへの出力が起こらない状態の環境を基準環境として持つ．起動直後やアプリケーションの実行中では出力があるため，そのような状態を避けて意図的に出力のほとんど計測されない状態を再現し，VM のスナップショットとして保持する．例えば特に必要の無い常駐プログラムは停止し，アイドル状態を続けることで OS の遅延書き込み処理を終了させる．同様に評価環境に対しても出力がほとんど計測されない状態を再現した上で提案システムによる検証を行う．

4.4 統計的手法による出力の解析

提案システムではキーボード入力を与えたときの基準環境と評価環境で計測したディスクやネットワークのスループットを比較し，スループットの平均に有意差がある場合キーロガーが存在していると判断する．スループットの違いが有意なものであることを判定するために t 検定を利用する．t 検定は 2 組の標本について平均に有意差があるかどうかの検定を行う統計学的検定法である．それぞれの環境で同じキーボード入力を与えたときのスループットの計測結果を n 対用意し，対応する対の差を $d_i(i = 1, 2, ..., n)$ とする．d_i の平均を $\overline{d}$ とする．このときの t 値を次の式で求める．

$$t = \frac{|\overline{d}|}{\sqrt{\sum_{i=1}^{n}(d_i - \overline{d})^2}/\sqrt{n(n-1)}}$$

これを有意水準 1%での検定結果からスループットの有意差を判定する．

5 実装

提案システムでは x86 システムを仮想化できるオープンソースソフトウェアである VirtualBox[4] を VMM として利用している．VirtualBox はキー

ボード，ハードディスク，ネットワークインターフェース，プログラマブルインタラプトタイマ (PIT) などを仮想デバイスとして持っている．

提案システムがキーボード入力を行う場合はキーダウンとキーアップにそれぞれ対応する割り込みを行うことで 1 文字分の入力を行う．これはキーダウンだけでキーを押し続けた状態を再現したとき，OS 内のキーリピート間隔の設定によるキーボード入力数を制限するような影響を避けるためである．与える入力は英数字だけでなく，数秒ごとにアプリケーションフォーカスの切替えを行うコマンドも入力する．これはキーロガーにはキーボード入力だけでなく入力されたアプリケーションも記録するものがあるため，アプリケーションフォーカスが変わったときにそれまで行われたキーボード入力のログとアプリケーション名を出力する仕組みになっている場合，英数字の入力だけではフォーカスが切り替わらずに出力が計測されないことになるのを防ぐためである．OS の内部時刻を制御するためにタイマ割り込みの間隔を変更する．OS は起動時に仮想 PIT にタイマ割り込み間隔の設定を行う．提案システムがこれより短い割り込み間隔を仮想 PIT に再設定することで内部時刻の更新を早める．ディスク出力は仮想ディスクへの出力ブロック数を記録し，同様にネットワーク出力は送信バイト数を記録する．

このような実装手法では VMM に実装しなければならない核心部分は仮想デバイスへの修正だけである．そのため VirtualBox だけでなく Xen[5] や VMware[6] といった他の VMM にも原理上適用可能である．

6 実験

提案システムによってキーロガーが検知できることを確認するための実験を行った．実験環境としてインターネット上から表 2 の 56 種類のキーロガーを収集し，Windows XP がクリーンインストールされた VM を基準環境とし，標本環境に各キーロガーをインストールしたものを評価環境とした．それぞれの環境に対して提案システムを用いてキーボード入力を与えた時のディスクとネットワークの出力を計測した．同様にキーボード入力を取得するという点でキーロガーと似たふるまいを行う表 1 の 8 種類のキーボードユーティリティに対しても実験を行った．これらのキーボードユーティリティはシステムに常駐することでキーボードレイアウトの変更やキーボード入力に関連するさまざまな操作性を提供するプログラムである．実験に用いた PC は Core 2 duo 2.0 GHz の CPU，2 GB のメモリを備えている．VM 上のゲスト OS に Windows XP を利用し，ホスト OS に Linux 2.6.19.7 を利用した．ゲスト OS には 128 MB のメモリを割り当て，残りをホスト OS に割り当てた．

本実験ではキーボード入力の手法を 3 種類のパターンに分けた．1) キーボード入力のみ．2) キーボード入力とフォーカスの切替え．3) キーボード入力，フォーカスの切替え，内部時刻の変更．実験を 1) のパターンから順に行い，キーロガーが存在すると判断されなかったものに対しては次のパターンで実験を行うものとした．それぞれのパターンでは 60 秒間に約 3 万文字のキーボード入力を与え，ディスクとネットワークのスループットを計測する．そして基準環境と評価環境にキーボード入力を行ったときの 5 秒ごとのスループットを対応する対として t 検定によるキーロガーの有無の自動判定を行った．

実験の結果，56 種類のキーロガーのうち 55 種類がディスク出力の結果から提案システムによってキーロガーと判定された．図 3 に Family Keylogger v2.83 について，提案システムによってキーロガーであると判定されたときに計測されたディスクスループットの様子を示す．キーロガーがインストールされている評価環境ではキーボード入力に応じて基準環境と比べてスループットが明らかに増加していることがわかる．検知されたキーロガーではいずれかの入力パターンにおいて全て同様のスループットの増加が計測された．検知されなかった Solid Key Logger Ver. 2.0 はログをファイルに出力しない設定であったため，図 4 のようにキーボード入力に対応したディスク出力は計測されなかった．また全てのキーロガーに対する実験においてネットワークの出力は計測されなかった．実験に用いたキーロガーは記録したキーボード入力を直ちにネットワークに送信するのではなく，設定された日時にログをまとめて送信する仕組みになっているためである．キーボードユーティリティではいずれの入力パターンでも明らかなディスクやネットワークのスループットの増加は計測されず，キーロガーであると誤検知されなかったため false positive は無かった．

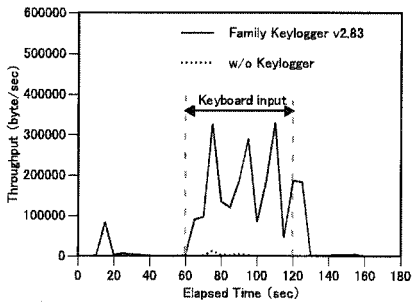

図 3: ディスクスループットの比較

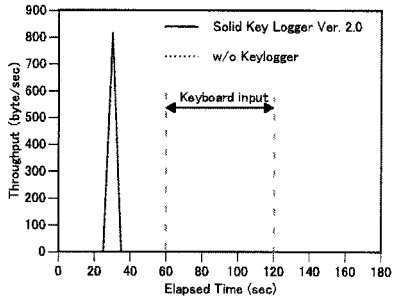

図 4: ディスクスループットの比較

7 議論

7.1 False positive

本稿の実験では false positive は無かった．実験で用いたキーボードユーティリティにはディスクやネットワークに出力するというふるまいが無かったためである．そこで検証したいプログラムのふるまい以外を原因とした false positive が起こる可能性として検査対象の環境のバックグラウンドで動作しているプログラムの影響が考えられる．本稿の検知手法ではキーボード入力に対応した出力の変化を検知に利用するため，キーロガーが存在していなくても偶然キーボード入力に対応した明らかな出力の増加があればキーロガーが存在すると判断してしまう．しかし基準環境や評価環境では検証対象のプログラム以外は全て任意の状態にすることができるため原理的に常に提案システムにとって最適な環境を用意することができる．よって検査対象の環境を要因とした false positive は起こりにくいといえる．

7.2 提案システムの回避手法

提案システムを回避するようないくつかの手法が考えられる．1) キーロガーの出力を検知させない．2) 提案システムによるキーボード入力を取得しない．

キーロガーの出力を検知させない単純な手法がログをファイルに出力しないことである．キーロガーの目的が最終的にネットワーク上のホストにデータを届けることである場合，ディスクにログを保存する必要は無いからである．実際，本稿の実験ではログをファイルに出力しない Solid Key Logger Ver. 2.0 を検知することができなかった．ログをファイルに出力する場合でも出力がキーロガーのものであるかどうかを提案システムが判別できなくさせることは可能である．例えばキーボード入力の取得時にファイルに出力するのでは無く，アイドル時に出力することで無害な常駐プロセスのようなふるまいをする方法が考えられる．

提案システムによるキーボード入力を取得しない手法として，特定のアプリケーションに対するキーボード入力のみを取得することや，大量のキーボード入力を提案システムによる入力とみなすなどして通常のキーボード入力と提案システムによるキーボード入力とを区別することなどが考えられる．

特定のアプリケーションに対する入力のみを取得する場合，まず提案システムは何がその特定のアプリケーションであるかを見付けなければ検知することができない．しかしキーロガーにとっては自身のキーボード入力の取得機能を制限することになるため大きな損失となる．また提案システムが網羅的にさまざまなアプリケーションに対して入力を行うことで検知できる可能性がある．

キーボード入力の数から提案システムによる入力を区別する手法の場合，キーロガーは一定時間あたりのキーボード入力数から判断することになる．しかし OS の内部時刻は VMM を利用することで任意に変更することができるため，キーボード入力と同様にタイマ割り込みを送ることでキーロガーに一定時間あたりのキーボード入力数が通常のものであるように認識させることが可能である．

7.3 運用方法

キーロガーは主にクライアント PC で動作するように作られている．よってその PC 環境はさまざまであるため，実環境では誤検知の要因になるキーボード入力とは無関係なディスクやネットワーク出力が頻繁に起こる．提案システムではあるプログラムがキーロガーの機能を持っているのかを検証するために運用されることを前提にしているため，キーロガーの動作する環境を固定することで誤検知の要因となりうるものを排除した基準環境上でキーロガーの検知を行うことができる．

原理上は実環境に対して既にキーロガーが動作しているかどうかを検証することも可能である．なぜならばキーボード入力とは無関係な出力がキーロガーの出力に比べて十分に小さければ，提案システムによるキーボード入力とキーロガーの出力とを関連付けることが可能であるからである．そこで実環境では検査対象の環境をどれだけ理想的な環境に近づけられるかが検知能力を高める重要な要因となる．実環境を基準環境のようにキーボード入力とは無関係な出力がほとんどない状態にすることができれば，キーボード入力が無いときの出力とキーボード入力を与えたときの出力を比較することでキーロガーを検知することが可能である．どの程度実現可能であるかはクライアント PC の環境に依存するため簡単に評価することは困難である．

8 関連研究

既存のシグネチャを利用したスパイウェアの検知手法ではさまざまな欠点があるため，スパイウェアらしいふるまいを利用した検知手法が提案されている．E.Kirda ら [7][8] はインターネットエクスプローラーのコンポーネントである Browser Helper Object (BHO) を利用して作られたスパイウェアの検知手法や解析手法を提案している．BHO による特定のブラウザ関数や Windows API の実行をスパイウェアのふるまいとみなした検知や，動的な taint analysis によって機密情報の流れを解析する．これらの手法は BHO を利用したスパイウェアにのみ注目しているため，それ以外のスパイウェアは対象にしていない．本稿では原理的にキーロガーの実装に依存しないため，BHO を利用したキーロガーに対しても有効である．Siren[9] は対象システムに既知のネットワークリクエストを出力するような入力を与え，スパイウェアのふるまいによって実際の出力と既知の出力が異なることを利用してスパイウェアを検知する．本稿ではどのような内容のネットワーク出力がなされたかを詳細に把握するのでは無く，出力量の変化にのみ注目する手法を用いている．

仮想機械技術のセキュリティ上の利点を利用した悪意あるプログラムの検知手法や解析手法が提案されている．VMwatcher[10] は検知システムを VM の外側で動作させることで，OS 上で動作するような既存の検知システムを回避するようなプログラムの検知を可能にする手法を提案している．そのために検知システムが必要とする OS レベルの情報を OS のコンテキストに関する既知の詳細な情報を利用することで VMM レベルの情報から再構築している．本稿の提案手法では OS レベルの情報を必要としないため OS の種類に依存しない．R.Crandall ら [11] は悪意あるプログラムの時間に依存するふるまいを解析する手法を提案している．これは仮想機械技術を利用して VM 上のシステムの時刻を任意に制御することで，システム内に存在する悪意あるプログラムがいつどのように動作するかの情報収集を行うものである．本稿では時間に依存したふるまいではなく，キーボード入力に対するふるまいを解析するという点で異なっている．

9 まとめ

本稿ではキーロガーのふるまいを仮想機械技術を用いて検知する手法を提案した．仮想機械技術によってキーロガーに対して大量の入力を与えることで，キーボード入力の取得とログの出力という本来の僅かなキーロガーのふるまいを意図的に顕在化することでキーロガーのふるまいを検知する．56 種類のキーロガーと 8 種類のキーボードユーティリティに対して検知精度を評価する実験を行い，55 種類のキーロガーを検知し false positive は無かった．今後は提案手法を回避するような手法に対する考察を深めることでその対策手法を考案する予定である．

参考文献

[1] Saroiu, S., Gribble, S. and Levy, H.: Measurement and Analysis of Spyware in a University Environment., *Proceedings of the ACM/USENIX Symposium on Networked Systems Design and Implementation*, pp. 141–153 (2004).

[2] Kruegel, C., Robertson, W. and Vigna, G.: Detecting Kernel-Level Rootkits Through Binary Analysis, *Proceedings of the 20th Annual Computer Security Applications Conference*, pp. 91–100 (2004).

[3] Chen, P. M. and Noble, B. D.: When Virtual Is Better Than Real, *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, p. 133 (2001).

[4] innotek AG: *VirtualBox*, http://www.virtualbox.org/.

[5] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A.: Xen and Art of Virtualization, *Proceedings of ACM Symposium on Operating Systems Principles*, pp. 164–177 (2003).

[6] Waldspurger, C. A.: Memory Resource Management in VMware ESX Server, *Proceedings of Symposium on Operating System Design and Implementation*, pp. 181–194 (2002).

[7] Kirda, E., Kruegel, C., Banks, G., Vigna, G. and Kemmerer, R. A.: Behavior-based spyware detection, *Proceedings of the USENIX Security Symposium*, pp. 273–288 (2006).

[8] Egele, M., Kruegel, C., Kirda, E., Yin, H. and Song, D.: Dynamic Spyware Analysis, *Proceedings of the 2007 USENIX Annual Technical Conference*, pp. 233–246 (2007).

[9] Borders, K., Zhao, X. and Prakash, A.: Siren: Catching Evasive Malware (Short Paper), *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pp. 78–85 (2006).

[10] Jiang, X., Wang, X. and Xu, D.: Stealthy malware detection through vmm-based "out-of-the-box" semantic view reconstruction, *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 128–138 (2007).

[11] Crandall, J. R., Wassermann, G., de Oliveira, D. A. S., Su, Z., Wu, S. F. and Chong, F. T.: Temporal search: detecting hidden malware timebombs with virtual machines, *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 25–36 (2006).

表 1: キーボードユーティリティ一覧

Sample name	
xkeymacs	u-key
Q's Nicolatter	NoIme2
にとしょーと	keylay21
X Button Maker	AppleKbWin

表 2: キーロガー一覧

Sample name
007 Spy Software Ver. 3.874
123Keylogger Ver. 1.0
Absolute Key Logger v3.0.302
Active Key Logger Ver. 3.7.3 04.05.2007
ActMon Ver. 5.03
Actual Spy Ver. 3.0
Advanced Invisible Keylogger Ver. 2.0.3
Advanced Keylogger Ver. 1.8
All In One Keylogger Ver. 2.8
All-In-One Spy Ver. 2.0
Ardamax Keylogger 2.8
Ardamax Keylogger Lite 1.2
Busted.NET Ver. 2.4.0
Computer Monitor Keylogger Ver. 2.9
Data Doctor KeyLogger Ver. 2.0.15
Elite Keylogger 3.5
Elite Keylogger 3.6
Family Keylogger Pro Ver. 2.9
Family Keylogger v2.83
Free Keylgger King 1.3
Free Keylogger 1.1
Free KGB Keylogger 4.04
Golden Keylogger
Home Key Logger Free Edition v1.70
Inside Keylogger 3.5
Invisible Keylogger
Keyboard device logger 1.0.4
Keyboard Spectator Lite Ver. 1.3
Keylogit
KGB 4.04
Klog v1.5
LoggerA
LoggerB
PC Spy Keylogger Ver. 2.3
Perfect Keylogger 1.66
Perfect Keylogger Lite 1.0
Power SPY Ver. 6.7.0
Powered Keylogger Ver. 2.00
Quick Keylogger 2.1
Remote Keylogger v4.0
Revealer Free Edition v1.2
SC-KeyLog PRO Demo Ver. 3.2
Sentinel Professional 2005 Ver. 4.0
Solid Key Logger Ver. 2.0
Spy Agent Build 6.01
Spy Lantern Keylogger v6.0
SpyBuddy Ver. 3.7.5
Spy-Keylogger Ver. 1.31
SpyKeySpy 1.1
Stealth Letlogger Ver. 4.5
Supreme Spy Ver. 2.9
Tiny Keylogger
typeteller
Win-Spy Software 8.9 Pro
XPCSpy Pro 3.10
キーロガー