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

2006－DPS－126（36）
2006－CSEC－32（36）
2006／3／17

# 動的 API 検査方式によるキーロガー検知方式の提案

高見知寛† 鈴木功一† 馬場達也‡ 前田秀介‡ 松本隆明‡ 西垣正勝†
†静岡大学情報学部 ‡NTT データ技術開発本部

**あらまし** 本稿ではキーボード入力を取得するというキーロガーの挙動に着目し，キーボード入力に用いられる API の使用を検出することでキーロガーの検知を行う方式を提案する．本来の DLL の代わりに API の使用を検出する機能を付加した検査用 DLL をプログラムにロードさせた上で試実行させることが本方式の特徴であり，ウイルス検知における動的ヒューリスティック法的なアプローチによるキーロガー検知方式となっている．本稿では本方式の基礎実験を行い，その検知率と誤検知率について評価する．

# A keylogger detection using dynamic API inspection

Tomohiro Takami†, Koichi Suzuki†, Tatsuya Baba‡, Shusuke Maeda‡,
Takaaki Matsumoto‡, Masakatsu Nishigaki†
† Faculty of Informatics, Shizuoka University ‡ R&D Headquarters, NTT Data corp.

**Abstract** This paper proposes a keylogger detection scheme by monitoring APIs employed by keylogger to capture user's keyboard input. API inspection is one of efficient ways for keylogger detection, since the use of keyboard-input-related APIs is a typical behavior found in keyloggers. To achieve this, we create a modified DLL which can detect the use of these APIs. By executing a program with the modified DLL, we can check whether the program includes any of these APIs or not. We can say that this scheme is in the category of dynamic heuristic virus detection (in dynamic heuristic detection, programs are executed in "virtual" machine to check virus behavior; in the proposed scheme, programs are executed with "virtual" DLL to check keylogger behavior). This paper carries out basic experiments to evaluate its detection rate and false detection rate.

## 1. はじめに

近年，スパイウェアと呼ばれる悪質なプログラムがインターネット上を横行し，その被害が増加している[1]．スパイウェアの被害が急速に拡大している原因の一つとして，スパイウェアに対するユーザの認識の低さが挙げられる．スパイウェアがインストールされている可能性のある環境（インターネットカフェ等）でのオンラインバンクやクレジットカードによるオンラインショッピングの利用により，スパイウェアの被害に遭うケースが少なくない[2]．さらにスパイウェアは，ウイルスのように感染活動を行わず，ユーザに気付かれないように行動するという特徴を持つ．そのためユーザは自分の PC がスパイウェアに感染していることにすら気付かず，それがスパイウェアによる被害を拡大する要因となっている．

このような現状に対して，各アンチウイルスベンダはスパイウェア検知機能を搭載したウイルス対策ソフトウェアを開発している．このようなスパイウェア検知機能の多くはウイルス検知の場合と同様にパターンマッチング法を用いたものが主流となっている[1]．しかしパターンマッチング法は既知のスパイウェアを検知するには非常に有効な方式であるが，未知のスパイウェアを検知することができないという欠点が存在する．

そこで本稿では，スパイウェアの中でも近年深刻な被害をもたらしているキーロガーを対象に，パターンマッチング法に依らない未知キーロガー検知方式を提案する．本方式は，キーボード入力を取得するというキーロガーの挙動に着目し，キーボード入力に用いられる API の使用を検出することでキーロガーの検知を行う．具体的には，本来の DLL の代わりに API の使用を検出する機能を付加した検査用 DLL をプログラムにロードさせた上で試実行させることにより，プログラムの中で当該 API が使用されているか否かを検査する．プログラムを試しに実行させるという意味では，本方式は，ウイルス検知における動的ヒューリスティック法的なアプローチによるキーロガー検知方式であるといえる．

## 2. 従来方式

### 2.1 パターンマッチング法の問題

現在のスパイウェア検知方式の主流はパターンマッチング法である[1]．パターンマッチング法は，スパイウェアの

特徴的なプログラムコードをパターンとしてデータベースに予め登録しておき，検査対象となるプログラムと比較することでスパイウェアの検知を行う方式である[3]．この方式では，既知のスパイウェアであれば，データベースに登録されているパターンから確実に検知することが可能であるため，既知のスパイウェアに対しては非常に有効な方式である．しかしデータベースに登録されていない未知のスパイウェアを検知することができないという問題が残る．

またウイルスの場合であれば，広範囲に感染活動を行うため比較的検体が入手しやすく，アンチウイルスベンダもウイルスのパターンを生成することができる．そのため，ウイルス対策とパターンマッチング法は比較的相性が良いともいえる．事実IPAの調査では，ウイルス届出件数はここ数年増加傾向にあるものの，実害のあったケースが毎年減少してきている理由をウイルス対策ソフトウェアの導入の増加による効果であると分析している[4]．しかしスパイウェアはウイルスと異なり，感染活動を行わないため，パターンを生成するための検体を入手することが困難であり[5]，パターンマッチング法の効果は期待できない．特にスパイウェアはその目的から特定のユーザやグループを狙う場合も多く，その場合検体の事前入手は実質不可能である．

以上の問題に対処するために，パターンマッチング法に依らない，未知のスパイウェアを検知する方式が求められている．

### 2.2 従来の未知ウイルス検知方式

従来の未知ウイルス検知の代表的な方式としてビヘイビアブロッキング法とヒューリスティック法が提案されている[6]．

ビヘイビアブロッキング法は，プログラムの振る舞いから「ウイルスらしい挙動」を検出する方式である．これはプロセスが発行するシステムコールなどを検査することにより，コンピュータ上で動作しているプログラムの動きをリアルタイムで監視し，ウイルスらしい挙動が検出された時点でそのプログラムをウイルスとして検知する方式である．

ヒューリスティック法も，ビヘイビアブロッキング法と同様，プログラムの振る舞いから「ウイルスらしい挙動」を検出する方式であるが，ウイルス本体を実環境上で実際に実行させることはない．ヒューリスティック法には静的ヒューリスティック法と動的ヒューリスティック法が存在する．静的ヒューリスティック法は，プログラムにマシン語レベルのコード解析を行い，コードの中から「ウイルスらしい」挙動を検出する．動的ヒューリスティック法は，仮想環境上でウイルスを実行させ，「ウイルスらしい挙動」を検出する．

いずれの未知ウイルス検知方式も，プログラムの振る舞いから「ウイルスらしい挙動」を検出することで未知ウイルスを検知するというアプローチを採っている．そこで，ウイルスと同じマルウェアであるスパイウェアに対しても同様のアプローチが有効であるのではないかと考えられる．つまりプログラムの振る舞いから「スパイウェアらしい挙動」を検出することで，未知スパイウェアを検知することが可能ではないかと考えられる．

## 3. スパイウェアおよびキーロガーの特徴

### 3.1 スパイウェアの定義

スパイウェアとは，ユーザの適切な同意なしにインストールされ，以下の事項をユーザの制御を超えて実装したものである[7]．

- ユーザの体験，プライバシー，システムセキュリティに影響を与える重要な変更
- ユーザに無断でコンピュータにプログラムをインストールさせることを含む，システムリソースの使用
- 個人情報や機密情報の収集，使用，配布

以上のように，スパイウェアはウイルスとよく類似しているが，感染機能や増殖機能を備えていない．

### 3.2 スパイウェアの種類

代表的なスパイウェアの種類のいくつかを以下に示す[1,8]．

- アドウェア
  ユーザが予期しない広告や求めていない広告などをユーザに提供する．
- トラッキングクッキー
  Webサイトの閲覧URL履歴などを収集するために利用する．
- キーロガー
  キーボードからの入力を記録する．
- ブラウザハイジャッカー
  スタートページや検索ページなどのWebブラウザの設定を改ざんする．
- ダイアラー
  アダルトサイトなどの有料または特定の番号に接続させる．
- リモートアクセスツール
  システムの遠隔アクセスを許可，もしくは制御するように設計されたツール．

### 3.3 キーロガーの特徴

3.2節で示した通り，スパイウェアは多種多様であるた

め，全てのスパイウェアを一括して検知する挙動を規定することは非常に困難である．そこで本稿では，スパイウェアの中でも近年オンラインバンクの不正送金事件[9]など深刻な被害を引き起こしているキーロガーに焦点を当て，キーロガーらしい挙動を検出することでキーロガーを検知する方式を提案する．

3.2 節で述べたように，キーロガーとはユーザのキーボード入力を取得するスパイウェアであるので，「キーボード入力を取得する」という挙動をキーロガーらしい挙動と規定すれば，この挙動を検出することでキーロガーを検知できるのではないかと考えられる．なお本方式では，クライアント PC の OS として広く普及しており，キーロガーの被害が拡大している Windows に焦点を当て，Windows 環境下にて動作するキーロガーを対象とする．

### 3.4　キーボード入力の取得

Windows 環境下において，キーボード入力を取得する方法は，著者らが確認した限りでは二つ存在する．キー判定 API を用いる方法と，フックを用いる方法である．Windows 環境下で実行されるキーロガーは，上記のいずれかの方法を用いてキーボード入力を取得していると考えられる．以下，これらの方法とその検出方法について説明する．

#### 3.4.1　キー判定 API を用いる方法

キー判定 API である GetAsyncKeyState は，呼び出し時に指定したキーが押されているか，また前回の呼び出し時以降に指定したキーが押されたかどうかを判定する API である．GetAsyncKeyState は単に指定したキーが押されているか否かを判定する API であるため，任意のプロセスへのキーボード入力を取得することが可能である．

キーロガーがキー判定 API を用いてキーボード入力を取得する場合，ID やパスワードなどを盗み出すために少なくとも A～Z と 0～9 のキーボード入力を取得すると考えられる．そこで本方式では，キー判定 API を用いたキーロガーの挙動を，「GetAsyncKeyState によって A～Z，0～9 の全てのキーボード入力を取得する」と規定する．

#### 3.4.2　フックを用いる方法

フックとは，Windows OS がプロセスへ送信するメッセージを取得する方法であり，自身のプロセスへのメッセージのみを取得するローカルフックと，全プロセスへのメッセージを取得することが可能なグローバルフックが存在する[10]．Windows OS が送信するメッセージのうち，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，キーボード入力を取得することが可能となる．なお，フックを行う際は，SetWindowsHookEx という API を用いる必要がある．

例えば，キーロガーがフックを用いて Web ブラウザ上でユーザにより入力される ID やパスワードを盗むためには，ブラウザへのキーボード入力を取得する必要がある．キーロガーから見てブラウザは他プロセスであるため，キーロガーは必然的にグローバルフックを用いることになる．そこで本方式では，フックを用いるキーロガーの挙動を，「SetWindowsHookEx によるグローバルフックを用いてキーボード入力を取得する」と規定する．

## 4.　検査用 DLL を用いたプログラムの試実行によるキーロガーの挙動検知

### 4.1　検査用 DLL の作成

本方式は，キーロガーらしい挙動を検出するために，3.4.1 節，3.4.2 節に示した二つの API の動作を監視する．

Windows 環境下では，GetAsyncKeyState や SetWindowsHookEx は，user32.dll に内包されている．よって，user32.dll に手を加え，これら二つの API が使用される場合にはアラートが発生するように DLL を改造することができれば，API のリアルタイム監視が可能となる．しかし，Windows2000 以降ではシステムの DLL である user32.dll を書き換えることは許されていない．そこで，user32.dll を直接改造する代わりにラッパーDLL[11]を作成するという方法を採用する．ラッパーDLL の開発環境には Microsoft Visual C++ 6.0 を用いた．

図 1 が user32.dll をラッピングすることにより作成した検査用 DLL の_ser32.dll である．検査用 DLL は基本的には内部で user32.dll を呼び出すだけのサブルーチン（ラッパー）であるが，GetAsyncKeyState，SetWindowsHookEx の二つの API が 3.4.1 節，3.4.2 節で規定した挙動通りに呼び出された場合にはアラートが発生するようになっている．

```
_ser32.dll (API_call) {
  if (API_call = = GetAsyncKeyState (0-9 && A-Z))
        alert ;
  if (API_call = = SetWindowsHookEx (Global_Hook))
        alert ;
   user32.dll (API_call) ;
}
```

図 1：検査用 DLL(_ser32.dll)

さらに本方式では，DLL の明示的なロードに対応するため，上記の二つの API の他に LoadLibrary についても監視の対象に加える．LoadLibrary は kernel32.dll に内包されているため，kernel32.dll についても同様に検査用 DLL

である_ernel32.dll（図 2）を作成した．なお，_ernel32.dll の動作の詳細は次節を参照されたい．

```
_ernel32.dll (API_call) {
    if (API_call = = LoadLibrary (DLL_call)) {
        if (DLL_call = = kernel32.dll) {
            DLL_call = _ernel32.dll ;
        }
        else if (DLL_call = = user32.dll) {
            DLL_call = _ser32.dll ;
        }
        else if (DLL_call ! = システム DLL) {
            Replace (DLL_call) ;
            # 呼び出される DLL のインポートセク
            # ションに列挙されている DLL 名の置換
        }
    }
    kernel32.dll (API_call) ;
}
```

図 2：検査用 DLL(_ernel32.dll)

4.2　検査用 DLL を用いてのキーロガー検知

本節では，検査用 DLL を用いてのキーロガー検知の手順を示す．

（1）検査対象となるプログラムに，kernel32.dll と user32.dll の代わりに検査用 DLL をロードさせる：

プログラムにロードされるDLLは，プログラムのインポートセクションに列挙されている．そのため，検査対象プログラムのインポートセクションをチェックし，kernel32.dllとuser32.dllが列挙されていた場合，その箇所を_ernel32.dllおよび_ser32.dllという名前に置換することで，検査対象プログラムに検査用DLLをロードさせることができる．なお，DLL自体も自身のインポートセクションを持っており，プログラムにロードされたDLLが他のDLLをロードすることも可能である．そのため，検査対象のプログラムにロードされたDLLのインポートセクションの中に列挙されるDLL名に対しても，同様の置換操作が必要である[1].

さらにWindows環境下では，LoadLibraryというAPIを用いて，プログラムの実行中に必要に応じてDLLを動的にロードすることも可能である．これを「DLLの明示的ロード」という[12]．明示的にロードされるDLLはインポートセクションに列挙されない．そこで_ernel32.dllによりLoadLibraryを監視し，検査対象プログラムに明示的にロードされたDLLがあった場合には，そのDLL内のインポートセクションについてもチェックを行い，そこに列記されているDLL名に対しても同様の置換操作を行う[2].

（2）DLL 名の置換を施した検査対象プログラムを実際に実行させる：

DLL 名が置換されているため，検査対象プログラムは検査用 DLL がロードされた形で実行されることになる．検査用 DLL は，検査対象プログラムの中で GetAsyncKeyState，SetWindowsHookEx が 3.4.1 節，3.4.2 節で規定した挙動通りに呼び出された場合にアラートを発生させる．アラートの発生は，検査対象プログラムの中に 3.4.1 節，3.4.2 節に示した「キーロガーらしい挙動」のいずれかが検出されたことを意味するため，アラートが発生したプログラムに対しては「キーロガーの疑いあり」という判断が下される．

プログラムを試しに実行させるという意味では，本方式は，ウイルス検知における動的ヒューリスティック法的なアプローチによるキーロガー検知方式であるといえる．すなわち，検査用 DLL という仮想的な DLL を用いてプログラムを試実行させることが，動的ヒューリスティック法における「仮想環境での実行」に相当している．

なお，実際にキーロガーを検知するにあたっては，すべてのプログラムを本方式によって検査する必要がある．しかしファイルシステム内のすべてのプログラムを検査するには多大な時間がかかるため，例えば定常時に稼動プロセスを監視しておき，新たなプロセスが起動した際に，順次そのプロセスに起因するプログラムを検査するという方式を採用することが得策であると考える．実行の瞬間に感染活動を開始するウイルスと違って，キーロガーによる被害は遅効性があるため，プログラムが起動された後にその検査を行っても大きな問題は発生しないと予想される．

## 5.　実験

本章では，キーロガー検知実験を行い，4 章で示した提案方式の実現可能性を示す．さらに規定した挙動によって，正規のプログラムがキーロガーとして検知されてしまう可能性もあるため，誤検知実験を行い，その結果について考察する．

---

[1] 一般のプログラムはWindows OSが提供しているシステムDLLも同時にロードするが，システムDLLは正規のプログラムであり，かつWindowsの保護機能[13]によってその正当性も保証されるため，システムDLLのインポートセクションについては置換操作を行わない．

[2] LoadLibraryによってkernel32.dllもしくはuser32.dll自身が明示的にロードされる場合には，それらに代えて検査用DLLをロードしてやれば事が足りる．

### 5.1 実験環境

本実験は，本学の LAN から物理的に切り離された PC 上で行った. 実験に使用したPCのスペックを表1に示す.

表 1：実験環境

| | 実験機 |
|---|---|
| OS | Windows 2000 Professional |
| CPU | PentiumⅢ 800MHz |
| メモリ | 256MB |

### 5.2 キーロガー検知実験

本方式の実現の可能性を示すために，図 1，図 2 の検査用 DLL を用い，実際にキーロガーの検知実験を行った. ただしキーロガーは検体の入手が非常に困難であるため，実験には商用のキーロガーやインターネット上で公開されているキーロガーを用いた. 本実験で使用したキーロガーとその実験結果を表 2 に示す.

表 2：キーロガー検知実験結果

| | キー判定 API | グローバルフック |
|---|---|---|
| SpyAnywhere [14] | × | ○ |
| Perfect Keylogger [15] | × | ○ |
| Activity Logger [16] | ― | ― |
| XPCSpy [17] | × | ○ |
| きいろがぁ [18] | ○ | × |
| キーロガー [19] | ○ | × |
| Parasite [20] | × | ○ |
| WingKEY [21] | × | ○ |
| キーのログをとる者 [22] | ○ | × |
| Casper [23] | × | × |

表 2 から，一部のキーロガーを除いたほぼ全てのキーロガーを本方式によって検知可能であることが示された. 検知できなかったキーロガーについては，以下でその理由を考察する.

Activity Logger は，試実行時に「ファイルが汚染されている」という旨のエラーメッセージが表示され，終了してしまうため，実験自体を行うことができなかった. これは Activity Logger が自身のファイルの改ざんチェックを行っており，インポートセクションの置換操作がファイルの改ざんと判定されてしまったためと推測される.

Casperは，自身がキー判定APIやグローバルフックによるキーボード入力の取得を行わないため，本方式では検知することができなかった. Casperは寄生先となるプロセスにキーボード入力を取得するためのスレッドを立てさせることで，寄生先プロセスにキーロガーの役割を果たさせる寄生型キーロガーである. なお，Casperの寄生先プロセスはexplorer.exeであることが確認できた. このような寄生型のキーロガーに対しては，他のプロセスにスレッドを立てるためのAPIであるCreateRemoteThreadを本方式と同様に監視し，「他のプロセスへスレッドを立てる」という挙動を検出する方式が考えられる. 他のプロセスへスレッドを立てるという挙動はスパイウェアらしい挙動であり，正規のプログラムでは通常は用いられないと思われるため，この挙動を追加した場合の誤検知の心配も少ないと予想される[3].

### 5.3 誤検知評価実験

次に正規のプログラムを用いて，本方式における誤検知の評価実験を行った. 誤検知の評価実験に用いるアプリケーションには，MS Office 製品やブラウザ，メーラ等のアプリケーションに加え，誤検知の可能性のあるアプリケーションを重点的に選択した. しかしこれらのアプリケーションは多機能であり，全ての機能を網羅して実験を行うことは非常に困難である. そこで本実験では，基礎実験として各アプリケーションの使用頻度の高い機能を実行した場合に対してのみの挙動監視となっている. 本実験の結果を表 3 に示す.

表 3：誤検知評価実験

| | キー判定 API | グローバルフック |
|---|---|---|
| MS Word | × | × |
| MS Excel | × | × |
| MS PowerPoint | × | × |
| Internet Explorer | ― | ― |
| Firefox | × | × |
| Outlook Express | ― | ― |
| Thunderbird | × | × |
| xkeymacs [24] | × | ○ |
| AltIME [25] | × | ○ |

表 3 から，使用頻度の高い機能を使用する限りの検査においては，一部のアプリケーションを除いて正規のアプリケーションを本方式によって誤検知するようなことはないことが示された. 誤検知してしまったアプリケーションに

[3] この挙動の検知を検査用DLLに追加実装したところ，Casperの「他のプロセスへスレッドを立てる」という挙動を検出することができた. しかし，寄生先のプロセスが強制終了してしまうという不具合が生じてしまった. これについては今後検討を重ねていきたい.

ついては，以下でその理由を示す.

xkeymacs と AltIME は，グローバルフックを用いてキーボード入力を取得しキーバインドを置換するアプリケーションである．そのため，本方式ではキーロガーとして誤検知されてしまった.

Internet Explorer と Outlook Express については，Activity Logger と同様に自身のファイルの改ざんをチェックする機能を持っており，さらにプログラムの自動修復機能を持っているため，実験自体を行うことができなかった.

## 6. まとめ

本稿では「キーボード入力を取得する」というキーロガーの挙動に着目し，キーボード入力に用いられる API を検出することによりキーロガーの検知を行う方式を提案した．検知実験や誤検知評価実験の結果から，本方式がキーロガー検知に有効であることを示した．また本方式ではパターンマッチング法に依らないキーロガー検知方式であるため，例え特定の相手を狙うようにカスタマイズされたキーロガーであっても，Windows 環境下で実行するプログラムである限り，本方式で検知することが可能であると期待できる.

今後は検知できなかったキーロガーへの対処や，誤検知の軽減を検討したい.

## 参考文献


[1] 与那原亨・大谷尚通・ 馬場達也・稲田勉, トラフィック解析によるスパイウェア検知の一考察, 情報処理学会研究報告, Vol.2005, No.70, 2005-CSEC-30, pp.23-29, 2005 年 7 月

[2] トレンドマイクロ,“ スパイウェアの種類” ,http://www.trendmicro.co.jp/soho/spyware/spywarelist/

[3] トレンドマイクロ,“ ウイルス検出技術” ,http://www.trendmicro.com/jp/security/general/tech/overview.htm

[4] 情報処理推進機構,“ 2005 年のコンピュータウイルス届出状況” , http://www.ipa.go.jp/security/txt/2006/documents/2005all-vir.pdf, 2006/01/10

[5] @IT,“ 急速に広がるスパイウェアの脅威“ , http://www.atmarkit.co.jp/fsecurity/special/86antispy/antispy01.html, 2005/11/30

[6] 情報処理推進機構,“ 未知ウイルス検出技術に関する調査” , http:/www.ipa.go.jp/security/fy15/reports/uvd/index.html

[7] Anti-Spyware Coalition, ”Definitions”, http://www.antispywarecoalition.org/documents/definitions.htm

[8] Anti-Spyware Coalition, ”Glossary”, http://www.antispywarecoalition.org/documents/glossary.htm

[9] ITmedia,“ スパイウェアによる不正送金被害が拡大, みずほ銀行やジャパンネット銀行でも“ , http://www.itmedia.co.jp/enterprise/articles/0507/06/news024.html, 2005/07/06

[10] Microsoft,“ SetWindowsHookEx” http://www.microsoft.com/japan/developer/library/jpwinpf/_win32_setwindowshookex.htm

[11] Microsoft MSDN,“ ラッパーDLL の使用” , http://msdn.microsoft.com/library/ja/default.asp?url=/library/ja/jpdnsql7/htm/btrieve5.asp

[12] Jefferev Richter（著）, 長尾高弘, 株式会社ロングテール（翻訳）,“ Advanced Windows 改訂第 4 版” ,アスキー出版局, 2001/05

[13] Microsoft,“ Windows ファイル保護機能について” , http://support.microsoft.com/default.aspx?scid=kb;ja;222193

[14] Symantec Security Response,Remacc.SpyAnywhere, http://www.symantec.com/region/jp/avcenter/venc/data/jp-remacc.spyanywhere.html

[15] Symantec Security Response, Spyware.Perfect.B, http://www.symantec.com/region/jp/avcenter/venc/data/jp-spyware.perfect.b.html

[16] Symantec Security Response,Spyware.ActivityLog, http://www.symantec.com/region/jp/avcenter/venc/data/spyware.activitylog.html

[17] Symantec Security Response, Spyware.XpcSpy, http://www.symantec.com/region/jp/avcenter/venc/data/spyware.xpcspy.html

[18] Vector, きいろがぁ, http://rd.vector.co.jp/soft/win95/util/se322072.html

[19] Vector, キーロガー, http://www.vector.co.jp/soft/win95/util/se334334.html

[20] Vector, Parasite, http://www.vector.co.jp/soft/winnt/util/se327656.html

[21]Vector, WingKEY, http://www.vector.co.jp/soft/winnt/util/se263226.html

[22] Vector, キーのログをとる者, http://www.vector.co.jp/soft/win95/util/se369025.html

[23] S-CENTER, Casper, http://www.s-center.net/index.php

[24] xkeymacs, http://www.cam.hi-ho.ne.jp/oishi/

[25] AltIME, http://www.chombo.com/

[26] 杉村友幸・鈴木功一・馬場達也・前田秀介・西垣正勝,“ 送受信データ間の相関に基づく未知ウイルス検知方法の提案” ,2005 年暗号と情報セキュリティシンポジウム予稿集,Vol.IV, pp.1735-1740,2005 年 1 月