マルチメディア通信と分散処理　111－16
コンピュータセキュリティ　20－16
（２００３．２．２７）

未知のコンピュータ・ウイルス検出プログラムの開発

松浦　佐江子*1　　加藤　道子*2　小島　量*2

「未知のウイルス」を「ウイルスとしての悪意のある振舞いパターンをもつプログラム」と捉え、その振舞いを検出するモデルを構築した。本モデルに基づき、Windows 上で動作する検出プログラムを開発し、既存のウイルスの検出実験を行った。本論文では振舞いパターンの検出モデルについて説明し、既存のウイルスに対する検出実験結果を報告する。

Development of an Unknown Computer-Virus Detection Program

Saeko Matsuura, Michiko Kato and Ryo Ojima

An unknown computer-virus was caught with "a program with malicious behavior patterns", and a model which detects the behavior patterns was built. A detection program which operates on the Windows was developed based on this model, and a detection experiment of an existing virus was conducted. This paper explains the detection model, and reports the detection experiment.

1．はじめに

コンピュータ・ウイルス（ここでは広義にワームも含める）は他のコンピュータに入り込んで、ユーザの意図に反する「悪意のある行為」を実行するプログラムである。この場合「悪意のある行為」とは、不特定多数の相手にメールを送付して大量のメールデータを広範囲に発生させたり、コンピュータ内のデータを破壊することである。コンピュータ・ウイルスの検出・駆除プログラムは、市販アプリケーションとして広く普及しているが、利用者が継続的に最新の情報を更新していく必要があり、更新を怠ると対策が不十分になる危険性がある。また、これらのアプリケーションは既知のウイルスにおける固有のプログラム・コードを定義したファイル（これをシグネチャ定義ファイルと呼ぶ）を参照する方式をとっており、未知のウイルスを未然に防ぐことはできない。われわれは「未知のウイルス」を「ウイルスとしての悪意のある振舞いパターンをもつプログラム」と捉え、その振舞いを検出するモデルを構築し、Windows 上で動作する検出プログラムを開発した[1]。本論文では本検出モデルの説明を行い、既存のウイルスに対する検出実験について報告する。

2．検出モデル

「ファイルを削除する」、「ファイルをコピーする」、「メールを送る」といった個々の操作はそれ自体「悪意のある行為」ではない。むしろ通常の操作として行われるものであり、プログラムから操作することも普通である。しかし、「あるプログラムが自分自身をその実行環境に保存し、自動的にメールに添付して送る」という行為は通常のプログラムでは行われない。ウイルスの行為には「破壊行為を含めて自己を拡散させる」とういう目的を達成するためのいくつかの「振舞いのパターン」があると考えられる。われわれはこのようなウイルスを「悪意のある振舞いパターンを実装したコード」として捉え、これを検出するモデルを考案した。

2.1　ウイルスの振舞いパターン

ウイルスの振舞いを記述するためのテンプレートを設定し、ウイルスの種別（ファイル型、ブート型、Win32 型、マクロ型）に応じて複数の具体的なウイルスのコードを解析した。テンプレートはウイルスを特定する性質を自然言語で整理するためのものであり、表１の項目から構成される。

表１　テンプレートの構成要素

感染場所	型・感染対象の属性・環境定義
ウイルスの起動プロセス	トリガ・起動される処理
ウイルスプログラムの構成	プログラムの構成要素
メモリ常駐	いつ・どこに・常駐の条件・方法・環境定義
API や割り込みベクトル等の書き換え	種類・書き換えの方法・環境定義
感染処理	前提・感染の条件・方法・保存情報・参照情報・環境定義他
発病処理	前提・発病の条件・症状・保存情報・参照情報・環境定義他
ステルス処理	どの処理に対する隠蔽行為か
暗号処理	どこに・暗号化および復号化の方法

1芝浦工業大学　システム工学部　電子情報システム学科　Shibaura Institute of Technology Department of Electronic Information Systems
2（株）管理工学研究所　Kanrikogaku Kenkyusho,Ltd

テンプレートを基にウイルスの振舞いを分析した結果、ウイルスに共通の性質から以下に示す６つのウイルスの振舞いパターンを抽出した。例えば、他のファイルや HD 等に寄生しないワームファイルの場合におけるパターンは A),B),E)である。

A) Copy & Entry Change Pattern：自己の複製を作成し、自分自身を何らかのトリガにより実行可能な状態にする。
B) Hook & Additional Pattern：システム機能を書き換えて、通常の機能に加えて独自の処理を呼び出す。
C) Exec Original Code Pattern：Copy & Entry Change Pattern で自己の複製を別のファイルの一部とした場合に、オリジナルのファイルを実行する。
D) Exec Original IPL Pattern：Copy & Entry Change Pattern で自己の複製を HD/FD の一部とした場合に、オリジナルの IPL を実行する。
E) Disease Pattern：HD への大量の書き込み・無作為なファイルの削除等の破壊活動を行う。
F) Stealth Pattern：Hook & Additional Pattern で呼び出す独自の処理。ウイルスが変更したファイルのサイズや作成日時をオリジナルの値に置き換える。

メールに添付されるウイルスの先駆けでもある Win32/Ska（Windows 32 ビット型)の振舞いはこれらのパターンで図１のように表される。

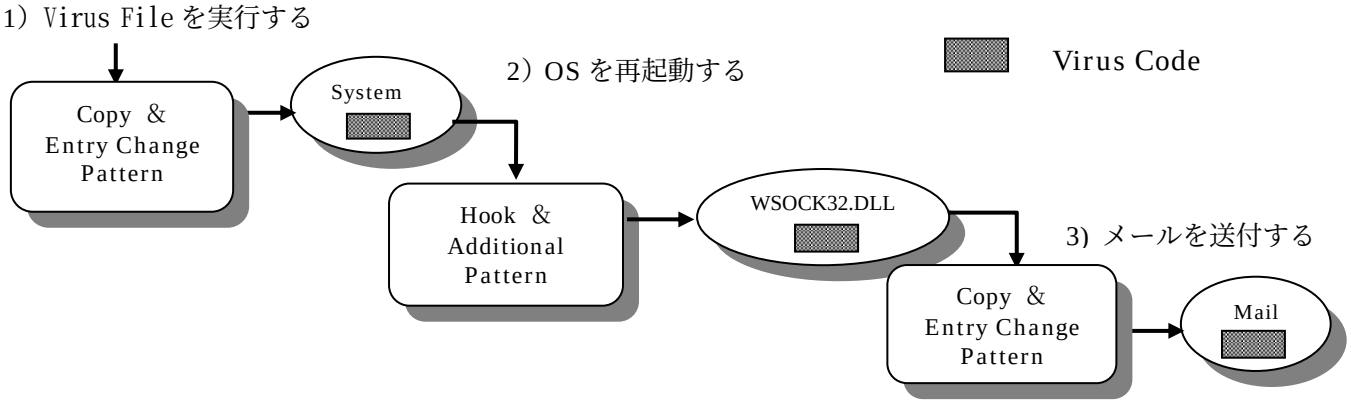


図１ W32/Ska Virus の振舞いのパターン

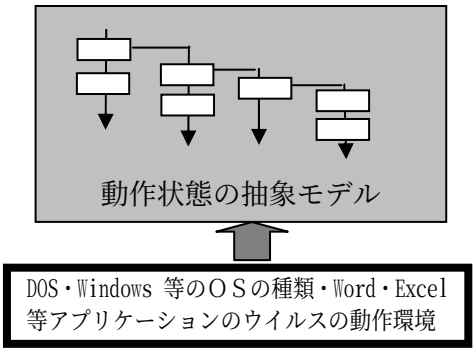
図１の読み方はつぎのとおりである。

1)［外部イベント］：ユーザにより、ウイルスコードが実行される。⇒ Copy & Entry Change Pattern を実行し、ウイルスコードがシステム上に永続化できるようにする。
2)［外部イベント］：ユーザにより、OS が再起動される。⇒ System 内のウイルスコードの起動により Hook & Additional Pattern を実行し、メールを自動送信するためのウイルスコードがメモリ上(WSOCK32.DLL)に存在するようにする。
3)［外部イベント］：ユーザがメールを送信する。⇒ メモリ上(WSOCK32.DLL)のウイルスコードにより Copy & Entry Change Pattern を実行し、ウイルスコード自身をメールの添付ファイルとする。

2.2 動作環境の抽象モデル

ウイルスの振舞いはそれぞれのウイルスが動作する環境において具体化される。すなわち、ウイルスが Windows で動作するには、振舞いが Windows の API やレジストリの仕組みによって表現されるので、ウイルスの振舞いは動作環境の定義に依存することになる。ウイルスの多様性には、動作環境に依存した実装の多様性と、ウイルス自身の機能の進化の多様性の 2 種類がある。多様なウイルスの振舞いを捉えるためには、2 つの多様性を分離して検出モデルを構築する必要がある。本研究では第一の多様性に対処し、ウイルスの振舞いおよび検出方法の汎用性を高めるために、動作環境における具体的なイベントを抽象化したレベルで捉えた「プログラムの動作状態の抽象モデル」を定義した（図 2 抽象モデル)。そして、この上でウイルスの振舞いパターンとその検出方法を定義している。抽象モデルはプログラムが動作する環境 System をそのメモリ上にあるプログラムの Function を識別できる Address、Function を生じさせる Event、そして Event の結果の状態の組で表すものである。そして、プログラムが生成する Event に基づき、ある場所の Function がどのよう

な State の変化を起こしているかを抽象モデル上の Pattern として捉える。




```
System := Empty| (Id , Address, Event , State),System
State := Empty| (Function , Address), State
Function := アドレス|コード|ファイル|ファイルマッピング|値| …
Address := コード|アドレス|テーブル|レジストリの値
          |セクションデータ …
Event:= メモリの呼び出し|メモリへの書き込み|ファイルの生成
        |ファイルのオープン|ファイルのクローズ|レジストリの設定
        |ディレクトリの生成|…
Pattern := ((Function, Address, State), (Function, Address, State))
```

図 2　抽象モデル

2.3　検出プログラムの構成

抽象モデル上でのパターンと検査対象プログラムの具体レベルの振舞いをつぎのように対応付けることによって、検出プログラムを実現する（図 3 参照)。

① 構造解析部：検査対象プログラム Program を逆アセンブルし、プログラムの構造を静的に解析する。アセンブラ・コードから、コントロール・フロー・グラフを生成し、これを用いて、シミュレータにより実行された部分の特定や、ループ構造などプログラムの構造の静的解析を行う。

② シミュレータ部：検査対象プログラム Program の振舞いを動的に検査する。本開発における動作環境の OS は Windows である。Windows のシステム機能を使用する命令およびデータ転送命令に対して、割り込みを発生させ、API の呼び出し状況のデータを収集する検査プログラムを Program に埋め込む(参照 APISPY32[3])。埋め込みを行った Program を実行し、API 呼び出しのパラメータや戻り値の情報を収集する。さらに命令の実行前と実行後のメモリ上のデータの状態変化に関するデータを生成する。

③ 検出モデル生成部：構造解析部ならびにシミュレータ部から得られたデータからプログラムの動作状態の抽象モデル上のデータを生成する。抽象モデルの定義に基づき、前述のような検査対象ファイルと検査対象ファイルの実行履歴（実行された時のイベントならびに状態の遷移:図 2 の System がこれに相当する）から構成されるデータを生成する。

④ 検出部：検出モデル生成部が生成した抽象モデル上の Program の実行履歴から、ウイルスの振舞いパターンを検出する。各パターンの組み合わせによりウイルスであるか否かの判定を行うルールに基づき、判定を行う。検出データが十分でない場合は、その理由を付加して、再度、①からのステップを繰り返す。ルールは「Copy & Entry Change Pattern が検出されたときに、その複製に Hook & Additional Pattern があり、その独自の処理が自己複製や破壊行為である」といった Pattern の組み合わせで定義される。

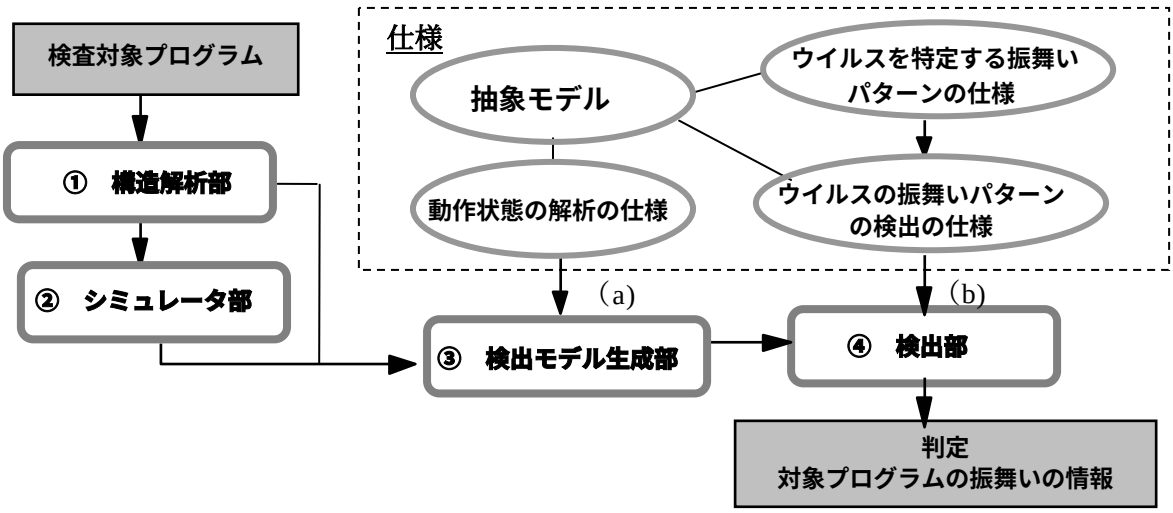


図3　検出プログラムの構成

われわれの提案する方法の第一の特徴はウイルス特有の振舞いを抽象モデル上のパターンとして捉えることである。第二の特徴は、ウイルスの第二の多様性である機能の進化に適応するために、プログラムがその仕様や動作環境の変化に伴い、進化するための仕組みであるEVA[2]に基づいて検出プログラムが設計されていることである。EVAは形式的プログラム開発技術と開発プロセスの定式化に基づくプログラムの変更メカニズムである。EVAに基づく設計はつぎのように行われている。

I. 抽象モデルに従い、**動作状態の解析に関する仕様**を定義し、この仕様に基づき検出モデル生成部のプログラムを定義する(図3 (a))。

II. 抽象モデルと**ウイルスを特定する振舞いパターンの仕様**に従い、**ウイルスの振舞いパターンの検出の仕様**を定義し、検出部のプログラムを定義する(図3 (b))。

3. ウイルスの検出実験

検出モデルに基づき、Windows 上で動作する検出プログラムを開発した。本検出プログラムにより「メール機能を利用したウイルス」の検出実験を行った。実験環境および検出プログラムの開発言語はつぎのとおりである。

表 2 実行環境と開発言語

実行環境 OS	Windows Me	**構造解析部**	Standard ML of new Jersey, Version 110.0.3
逆アセンブル	Visual C++, Version 6.0	**検出モデル生成部**	
シミュレータ		検出部	

実験はつぎの項目を実証することを目的とした。

(a) W32/Ska を検査対象とし、自分自身をメールに添付して送付するウイルスであることを検出する。すなわち、図 1 に示す W32/Ska ウイルスの振舞いパターンが検出できることを実証する。

(b) ウイルスの振舞いは外部イベントに応じて起動するために 1 度のシミュレーションですべての振舞いデータを収集できるわけではない。そこで、前述の①～④のプログラムを、繰り返し実行することが必要となる。この時、繰り返しを必要とする理由とそのデータが個別のウイルスに依存しない、汎用性のあるものであるかを検査する。

検出プログラムによる W32/Ska の検出実験の結果を付録に示す。Copy & Entry Change Pattern、Hook & Additional Pattern が検出されたパターンの種類を示している。空欄は、該当するパターンが検出されなかったことを意味する。各フェーズの概要はつぎのとおりである。

① Phase 0 では、検査対象ファイル W32ska.exe が C:\Windows\System\Ska.dll と C:\Windows\System\Ska.exe にコピーされ、レジストリのキーが書きかえられて、OS の再起動時にコピーされた Ska.exe が実行されるように設定されている。このフェーズでは Copy & Entry Change Pattern のみが検出され、Hook & Additional Pattern は検出されなかったことから、つぎのフェーズで、検査対象ファイルの未実行部分を実行に失敗した理由（ファイルのオープンに失敗した）から判別することによって、検査対象ファイルの未実行部分を再度実行する。

② Phase 1 では、W32ska.exe の未実行部分を実行するように変更した W32ska-1.exe を検査対象ファイルとする。ここでは、W32ska-1.exe が C:\Windows\System\Ska.dll にコピーされている。さらに、Hook & Additional Pattern では、Tsock32.dll （これはシステムファイル C:\Windows\System\wsock32.dll のコピーであり、危険防止のためにシミュレータ部が置き換えている）の export function の内、connect と send のコードが検査対象ファイルのコードで置き換えられている。検査対象ファイルがウイルスであるか否かは、この置き換えられたコードの振舞いに依存することから、置き換えられたコードの振舞いをつぎのフェーズで検査する。

③ Phase 2 では、システムファイル wsock32.dll の export function、connect と send の振舞いを検査する。ここでは、send が呼ばれた時に、検査対象ファイルをコピーした Ska.exe を send のデータとしていることが検出されている。これは、検査対象ファイル自身がメールの添付ファイルになったということである。

以上の検出結果から、検査対象ファイルはウイルスの振舞いをもつと判定した。

4. 評価

実験の目的の観点から検出プログラムの評価を行う。

(a) 固有のデータに依存せずに、ウイルスの振舞いパターンによって、「検査対象ファイルが自分自身をメールに添付して送付するウイルスである」ことを検出することができた。

(b)　検出における①～④のプログラムの再実行のトリガはつぎの通りである。

A)　検出したパターンがウイルスであることのルールを満たしていない。

B)　実行中にロードされたファイルが検査対象ファイルまたはそのコピーである。

C)　実行されていないコードがある。

D)　Windows のシステム機能を使用する命令の内、ファイルへの書きこみまたはファイルのオープンに失敗した。

ここで、A)はウイルスであるか否かを判定する「パターンに基づく汎用性のあるルール」である。B)はあるファイルが検査対象ファイルであるか否かということである。C)および D)はウイルスとは無関係の条件である。これらのことから、繰り返しを必要とする理由とそのデータは個別のウイルスに依存しない、汎用性のあるものであると言える。ただし、C)に関するつぎの問題は現在、検討中である。

- 一般の条件分岐でコードが実行されていない場合に、すべての条件分岐の組み合わせで再実行を行うことは効率が悪い。時間等、ウイルス特有の条件に限定して条件分岐の条件の変更を行うべきである。

5. まとめ

既存のウイルス検出技術では、プログラムの静的解析ならびに動的解析により得られたシグネチャ定義により、ウイルスであることを特定している。本研究では、これらの解析により、人間が検査対象がウイルスであるか否かの判断材料としている知識をウイルスの振舞いに関する知識として、その動作環境と分離した形式で蓄積している。そして、これらの知識を利用することによって、ユーザの意図に反する「悪意のある行為」を実行するプログラムをより広く検出する方法を提案した。実際の Windows 32 ビット型のウイルス Ska を検出することができた。しかし、ウイルスによっては DLL の呼び出しが直接取り出せないような細工が施されている場合がある。今後は、実験を重ね、本方法の有効性を検証するとともに、未知のファイルを実行する前にその振舞いを検査するプログラムとして改良を行う予定である。

参考文献

1. 未知ウイルス検出プログラム, NEDO 実用化開発助成事業成果展示会, 2002, 松浦, 加藤, 小島.
2. EVA : A Flexible Programming Method for Evolving Systems, IEEE Transactions on Software Engineering, Special Issue on Formal Methods in Software Practice, Vol.23, No.5, pp.296-313,1997, Matsuura S., Kuruma H. and Honiden S.
3. http://www.internals.com/

謝辞

本研究は、新エネルギー・産業技術総合開発機構（NEDO）産業技術実用化開発費助成事業「進化機構をもつコンピュータウィルス対策ワクチンプログラムの開発」において行われたものである。

付録

```
Phase 0
================================================================================
Copy & Entry Change Pattern
================================================================================
src     initial state
        from File :
                Code : ##
where   File : C:\Windows\System\Ska.dll
                Address : start = 00450078, end = 00452077
        from Memory : start = 00400000, end = 0044ffff
                Current Module : W32ska.exe
--------------------------------------------------------------------------------
src     initial state
        from File :
                Code : ##
where   Registory MainKey : HKEY_LOCAL_MACHINE,
                SubKey : Software\Microsoft\Windows\CurrentVersion\RunOnce
                File : Ska.exe
        from File : C:\Windows\System\Ska.exe
                Current Module : W32ska.exe
================================================================================
```

```
========
Hook & Additional Pattern
========
========
----> Failed Instruction : CreateFile of C:\Windows\System\wsock32.dll
There are copy patterns. We will check the other unexecuted codes.

Phase 1
=====================================================
Copy & Entry Change Pattern
=====================================================
src     initial state
        from File :
                Code : ##
where   File : C:\Windows\System\Ska.dll
                Address : start = 00450078, end = 00452077
        from Memory : start = 00400000, end = 0044ffff
                Current Module : W32ska-1.exe
=====================================================
=============================================================================
Hook & Additional Pattern
=============================================================================
old     Export function : connect in File : C:\Windows\System\Tsock32.dll
                Address : start = 83a34462
        from File : C:\Windows\System\Tsock32.dll
                File : C:\Windows\System\wsock32.dll
new     Export function : connect in File : C:\Windows\System\Tsock32.dll
                Address : start = 83a353b0
        from Memory : start = 00400000, end = 0044ffff
                Current Module : W32ska-1.exe
-----------------------------------------------------------------------------
old     Export function : send in File : C:\Windows\System\Tsock32.dll
                Address : start = 83a34738
        from File : C:\Windows\System\Tsock32.dll
                File : C:\Windows\System\wsock32.dll
new     Export function : send in File : C:\Windows\System\Tsock32.dll
                Address : start = 83a353f7
        from Memory : start = 00400000, end = 0044ffff
                Current Module : W32ska-1.exe
=============================================================================
----> Original File of C:\Windows\System\Tsock32.dll is C:\Windows\System\wsock32.dll
There are hook patterns. We will check the other unexecuted codes.

Phase 2
=================================================================
Copy & Entry Change Pattern
=================================================================
src     initial state
        from File :
                Code : ##
where   Parameter Name : send and
                Parameter Value : start = 0064fd10, end = 00650d0F
                Address : start = 0064fd10, end = 00650d0F
        from File : C:\Windows\System\Ska.exe
                File : C:\Src\Demo\Test\W32ska-1.exe
-----------------------------------------------------------------
src     initial state
        from File :
                Code : ##
where   Parameter Name : send and
                Parameter Value : start = 00650d10, end = 0065241f
                Address : start = 00650d10, end = 0065241f
        from File : C:\Windows\System\Ska.exe
                File : C:\Src\Demo\Test\W32ska-1.exe
=================================================================
========
Hook & Additional Pattern
========
========
----> This file contains Virus codes.
```