情報処理学会論文誌 Vol. 50 No. 9 2173–2181 (Sep. 2009)

危険なシステムコールに着目したWindows向け異常検知手法

伊波 靖[†1] 高良富夫[†2]

近年，不正なプログラムの感染および拡大方法の多様化と伝搬速度の高速化により，シグネチャによる不正プログラム対策ソフトウェアの限界が議論され，プログラムの振舞いに基づくビヘイビア型異常検知システムがさかんに研究されている．しかし，ビヘイビア型異常検知システムでは，高い検知率を得ることにともなう False Positive の割合を減少させることが課題となっている．本論文では，Windows においてシステムの資源に影響を与える危険なシステムコールに着目した異常検知手法を提案する．提案方式は，まず，OS が管理する重要な資源に影響を与えるクリティカルなシステムコールを，不正なプログラムの振舞いから定義したシステムコールと引数によるルールを用いて検知する．次に，それ以前に発行されたシステムコールの履歴から Support Vector Machine（SVM）を用いて，検知したクリティカルなシステムコールが不正なプログラムによって発行された危険なシステムコールかどうかを識別することで異常を検知する．我々は，提案方式に基づくプロトタイプシステムを開発し，現実的な不正なプログラムおよび通常のプログラムを用いて実験を行った．実験では，提案方式の検知能力と False Positive の割合について評価を行った．

Detecting Anomalies on Windows Focused on Dangerous System Call

Yasushi Iha[†1] and Tomio Takara[†2]

In recent years, infection and expansion method of malicious program are diversified, and the propagation speed has been rapid. The limit of the detection of security systems by signature has been indicated. Therefore, the behavior type anomaly detection system based on the behavior of the program has been actively researched. However, decreasing the ratio of false positive according to obtaining a high detection rate becomes a problem in the behavior type anomaly detection system. In this paper, we propose the anomaly detection method of combining behavior of program and detection rule to detect a dangerous system call that affects important resource of Windows system. The proposed method first detects a doubtful system call by the detection rule using system call and argument. Then, a dangerous system call is identified by using Support Vector Machine (SVM) from the history of the system call, and execution is intercepted. We performed an experiment by developing the prototype system based on the proposed method, and using realistic malicious program and usual program. Through the experiments, we have evaluated the detection rate of the proposed technique and the ratio of false positive.

1. はじめに

近年，インターネットの急速な進展にともない，不正なプログラムの感染および拡大方法の多様化と伝搬速度の高速化が情報システムに対する大きな脅威となっている[1)]．特にクライアント環境として普及している Windows 系の OS をターゲットとした，ウイルス，ワーム，ボット，トロイの木馬などの多種多様な不正なプログラムが流通している．Windows 系の OS では，システムに不慣れな一般ユーザであっても管理者権限が付与されているケースが多く，不正なプログラムを誤って実行したり，未知の脆弱性を用いた攻撃が行われたりした場合，システムに大きな被害を与えることになる．

不正なプログラムへの対策として，開発メーカによって提供される脆弱性に対するパッチを適用したり，不正プログラム対策ソフトウェアなどを利用したりすることが推奨されている．しかし，これらの対策を施しても，パッチが配布される以前に攻撃コードが出現する，いわゆるゼロデイ攻撃により攻撃が成功するケースがある[2)]．また，不正プログラム対策ソフトウェアの多くは，あらかじめ解析により判明した不正なプログラムを特徴付けるシグネチャと呼ばれるパターンを用いて不正なプログラムを検出しているため，頻繁に出現する新種の攻撃コードや次々と登場する多岐にわたる亜種へのリアルタイムの対応が限界となりつつある．

上記の問題を解決するための方法として，不正なプログラムの検出をプログラムの振舞いに基づいて行うビヘイビア型の異常検知システムがさかんに研究されている．特に UNIX 系の OS においては，システムコール列の学習による異常検知など，プログラムの振舞いに着目した研究が早くから行われ，多くの手法が提案されてきた[3)–5)]．

†1 沖縄工業高等専門学校メディア情報工学科
Department of Media Information Engineering, Okinawa National College of Technology
†2 琉球大学工学部情報工学科
Department of Information Engineering, University of the Ryukyus

一方，Windows 系の OS においては，システムサービスの呼び出し列を N-gram 法と呼ばれる方法により特徴化し，あらかじめ通常のプログラムを実行して収集したデータに基づいて異常検知を行う手法が島本らによって提案されている[6]．また，システムコールを監視する異常検知システムとして，WHIPS が提案されている[7]．しかし，WHIPS では不正なプログラムが発行する危険なシステムコールをあらかじめ調査しアクセス制御データベースに登録する必要がある．この作業は一般の利用者にとって難しいため，登録されていない未知の不正なプログラムへの対応が課題となっている．

一般的に，ビヘイビア型の異常検知システムにおいて高い検知率を得るためには，通常のプログラムを誤って不正なプログラムと判定してしまう False Positive が増加する問題を解決する必要があった．たとえば，ソフトウェアをシステムに導入するために使用されるインストーラなどの通常のプログラムは，システムフォルダへのファイルの作成などの不正なプログラムと類似した動作を行うため，ビヘイビア型の異常検知システムでは不正なプロラムと判定されてしまう可能性が高い．そのため，セキュリティ対策ソフトの評価指標の 1 つとして，False Positive が小さいほど検知精度が高いということになる．

そこで，本論文では，Windows 系の OS において，ルールベースの検知とプログラムの振舞いを組み合わせた新たな異常検知方式を提案する．提案方式では，まず，クリティカルなシステムコールと呼ぶ OS の重要な資源に影響を及ぼす可能性のあるシステムコールを不正なプログラムの挙動に基づいてあらかじめ定義したルールで検知する．次に，検知したクリティカルなシステムコールについて，システムコールの発行履歴から N-gram 法[3] を用いて生成した特徴ベクトルを素性とする Support Vector Machine（SVM）により不正なプログラムにより発行された危険なシステムコールかどうか判定する．

我々は，提案方式に基づくプロトタイプシステムを開発し，有効性を確認するために，現実的な不正なプログラムおよび通常のプログラムを用いて実験を行い，有効性を評価するための様々な測定結果を得た．その結果，提案方式の検知能力が高いことと，False Positive を低減させることに有効であることが分かった．

本論文では 2 章で危険なシステムコールについて説明する．3 章で SVM について概説する．4 章で我々の提案システムについて述べ，その実験結果を 5 章で示す．6 章で結論と今後の課題について説明する．

2. 危険なシステムコールについて

2.1 システムコールの概要

システムコールは，Windows 上で動作するプログラムが Windows のカーネルの機能を利用するための仕組みである．システムコールを呼び出すことにより，カーネル以外のユーザモードと呼ばれる領域で動作しているアプリケーションなどからカーネルモードに制御が遷移し，Windows が管理する様々な資源に対する操作を行うことができる．システムコールには，ファイル操作，ディレクトリ操作，レジストリ操作，スレッド操作，プロセス操作などの Windows 系 OS における基本的な機能が含まれている．また，Windows XP 以降では，Window Manager と Graphics Device Interface に関する関数群も加えられている．Windows 系 OS ではバージョンアップのたびにシステムコールの数が増え，Windows XP Service Pack 2 では 1083 となっている．通常，ユーザモードで動作するプログラムがシステムコールを直接呼び出すことはなく，Windows が提供しているサブシステムを利用してシステムコールより抽象度の高い API（Application Program Interface）を利用する．システムコールはアプリケーションが呼び出すユーザモードで動作する API と区別するためにネイティブ API と呼ばれる[8),9)]．

2.2 危険なシステムコールの定義

本論文において着目している危険なシステムコールの定義について説明する．文献 7) では，システムコールとシステムコールの引数およびシステムコールを発行したプロセスの組合せにより危険なシステムコールを定義した．本論文では，文献 7) に基づき，危険なシステムコールを以下のように定義する．

定義 1　危険な引数とは，システムの可用性に対して影響を与える可能性がある引数である．

定義 2　クリティカルなシステムコールとは，危険な引数をともなって発行されるシステムコールである．

定義 3　危険なシステムコールとは，悪意を持ったプログラムによって発行されるクリティカルなシステムコールである．

まず，危険な引数とは，システムの可用性に対して影響を与える可能性がある引数とする．たとえば，Windows システムフォルダなどの重要なファイルシステムに対するアクセスや，プログラムの自動実行などに関係するレジストリへのアクセスなどが危険な引数となる．危険な引数の対象として考慮すべきシステムの資源には以下のものがある．

ファイルシステム　システムの運用に必要なファイルとそれを格納したディレクトリなど．不正なプログラムは感染のために自身のコピーをシステムディレクトリに作成する．

レジストリ　システムの運用に必要なデータを格納したレジストリなど．不正なプログラムは再起動時に，自身が自動実行できるようにレジストリへ設定を書き込む．

プロセス　ウイルス対策ソフトウェアなどの別プロセスに対する攻撃と，自身の隠匿を目的とした別プロセスの起動など．不正なプログラムは，ウイルス対策ソフトウェアなどのプロセスを停止させたり，別プロセスになりすまして自身を隠匿したりする．

ネットワーク　ネットワークを経由した自身の増殖と情報の漏洩など．不正なプログラムは，広範囲への自己増殖を目的としてネットワークを経由したコピーを行う．また，宿主のファイルなどの情報を外部へ漏洩する．

次に，クリティカルなシステムコールとは，危険な引数をともなって発行されるシステムコールと定義する．

たとえばファイルを作成するシステムコール NtCreateFile は，それ自体ではクリティカルなシステムコールとはいえないが，システムディレクトリ内へのファイルの新規作成や，存在しているシステムファイルに対する上書きを意図した危険な引数が指定された場合に，クリティカルなシステムコールとなる．

そして，危険なシステムコールとは，悪意を持った不正なプログラムによって発行されるクリティカルなシステムコールと定義する．悪意を持った不正なプログラムは利用者の意図しないクリティカルなシステムコールにより，システムの可用性に重大な影響を与える．

2.3　不正なプログラムによる例

実際に不正なプログラム（Worm）により，自己の複製を目的として行われた動作を**表 1** に示し，発行されたシステムコールに基づいてクリティカルなシステムコールについて説明する．

表 1　不正なプログラムによるシステムコール発行例
Table 1　Example of system call by malicious program.

A 2839 980 1024 23 NtCreateFile (0x40110080,
24, 0, 0x40, 0, 1242188, ”￥??￥c:￥windows￥netcheck.exe”,
0x0, 32, 0, 2, 100, 0, 0, ... 168,
status=0x0, info=2,) == 0x0
…… 途中省略 ……
B 2848 980 1024 200 NtWriteFile (168,
0, 0, 0, ”MZP￥0￥2￥0￥0￥0￥4￥0￥17￥0￥377￥377”,
65536, 0x0, 0, ... status=0x0, info=65536,) == 0x0

A において Worm は NtCreateFile に与えられた引数から Windows のシステムフォルダにファイルを作成しようとしていることが分かる．この場合，与えられた引数が Windows フォルダへのパスとなっているため定義 1 により危険な引数となる．また，定義 2 より A の NtCreateFile はクリティカルなシステムコールとなる．A の NtCreateFile の実行により，戻り値としてファイルハンドル 168 が得られている．

次に，B において，A による戻り値であるファイルハンドル 168 が引数として与えられた NtWriteFile によりファイルの書き込みが行われていることが分かる．ファイルハンドル 168 は，A のクリティカルなシステムコールによる戻り値であるため，定義 1 より，危険な引数となる．このことより，B における NtWriteFile も定義 2 より，クリティカルなシステムコールとなる．

なお，NtWriteFile の 5 番目の引数に「MZ」で始まるデータが書き込み用のデータとして指定されているが，この文字列は Windows における実行形式ファイルのマジックナンバであることから B において実行形式のファイルを Windows システムフォルダに作成しようとしていることが分かる．この例の NtWriteFile ように，ファイルハンドルの値だけではクリティカルなシステムコールかどうか確認することができないが，別のクリティカルなシステムコールの戻り値を危険な引数とすることで，クリティカルなシステムコールになる場合もある．

不正なプログラムが自分自身を Windows のシステムフォルダへコピーする振舞いを考えた場合，(1) 自己ファイルのオープンと読み込み，(2) Windows システムフォルダへのファイルの作成と読み込んだ自己ファイルの書き出しが行われる．すなわち，この振舞いを，自己ファイル読み込みブロックとファイル書き出しブロックの 2 つのブロックによって行われていると考えることができる[10]．

本論文で提案するルールベースの検知では，システムコール単独でクリティカルなシステムコールを検知するのではなく，自己ファイルの読み込みブロックやファイルの書き出しブロックのように複数のシステムコールをファイルハンドルなどで関連付けることでブロックとしてルールを定義している．

3.　Support Vector Machine（SVM）

SVM は，Vapink によって提案された，統計的学習理論に基づく新しい 2 クラスのパター

ン認識手法である．ニューラルネットワークなどの従来法と比較して汎化能力が高い点と最適解が求まる点に特徴があり，学習に用いていないデータに対しても高い認識率を示す．SVM が優れている理由に，クラス分類を行う識別面を一意に決定するために「マージン最大化」という明確な基準が設けられている点と，線形分離することが不適切な場合「カーネル法」により非線形の判別問題への拡張ができることがあげられる．また，SVM は，その学習に認識誤りと汎化性能の両面から最適化が行われ，これが 2 次の凸計画問題として定式化されているため最適解を求めることができる[11]．

3.1 SVM の概観

与えられた学習用データ集合 S を，

$$(\mathrm{x}_1, y_1), (\mathrm{x}_2, y_2), \cdots, (\mathrm{x}_n, y_n) \quad (1)$$
$$\forall i, \quad \mathrm{x}_i \in \mathbb{R}^N, y_i \in \{-1, 1\}$$

とする．ここで x_i はデータ i の特徴ベクトルであり，y_i はデータ i のクラスラベルで，$+1$（正例）か -1（負例）を表す．

SVM は**図 1** に示すように，データ集合 S を正しく分離するために式 (2) で与えられる超平面のうち，マージン（分離超平面とベクトルの距離）が最大になるような分離超平面が最も汎化能力が高いものと判断する．

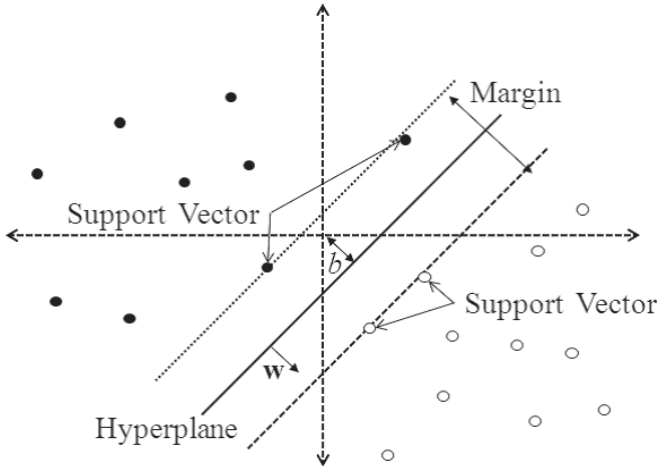


図 1　線形 SVM
Fig. 1　Linear SVM.

$$f(x) = \mathrm{w} \cdot \mathrm{x} + b \quad (2)$$
$$\mathrm{w} \in \mathbb{R}^N, \ b \in \mathbb{R}$$

ここで，w は n 次元の法線ベクトル，$(\mathrm{w} \cdot \mathrm{x})$ は w と x の内積を表す．

SVM では，学習事例が線形分離不可能な場合には，カーネル法を組み合わせることでアルゴリズムを容易に非線形に拡張することができる．よく使用されるカーネル関数としては，線形カーネル $(\mathrm{x}_i \cdot \mathrm{x}_i)$，多項式カーネル $(\mathrm{x}_i \cdot \mathrm{x}_j + 1)^d$，RBF（Radial Basis Function）カーネル $\exp\{-|\mathrm{x}_i - \mathrm{x}_j|^2/2\sigma^2\}$ などがある．

SVM は，カーネル関数を用いることにより，素性ベクトルを高次元の素性空間に写像し，素性空間において線形分離を行う．この写像によって SVM は線形分類器でありながら，非線形分離が可能となっている．

4. 提 案 方 式

4.1 前提となる考え方

通常のプログラムでは，システムに危険を及ぼす行為をする場合，事前にユーザへの確認などを行うのが一般的であるが，悪意を持った開発者によって作成された不正なプログラムは，システムに危険を及ぼす行為を，ユーザの意図しない状況で行うと考えられる．つまり，不正なプログラムと通常のプログラムでは，危険なシステムコールを実行する前のシステムコール発行履歴に違いがあると考えることができる．たとえば，通常のプログラムはユーザとのインタラクション（確認ダイアログの表示，メッセージの表示，了解ボタンの処理など）が行われるが，不正なプログラムの場合は，ユーザとのインタラクションがなく，ユーザの了解なしに行われる．そこで，クリティカルなシステムコールをルールで検知し，そこに至るシステムコールの発行履歴から振舞いの違いを N-gram 法[3] により特徴を表現することにより，危険なシステムコールを検知することが可能であると考える．

4.2 基本的な枠組み

提案方式の流れを**図 2** に示す．本論文で提案する異常検知手法では，まず過去の不正なプログラムの振舞いを分析し，クリティカルなシステムコールとなりうるシステムコールと引数のリストをあらかじめ作成し，クリティカルシステムコールデータベース（CSCDB）を構築しておく．

文献 12) では，既存のビヘイビアブロッキング法で規定されている 6 つの「ワームらしい振舞い」を取り上げている．本研究では，これを基に，2.2 節の定義に基づいて**表 2** に示すように，不正なプログラムらしい振舞いに基づくクリティカルなシステムコールの候補と

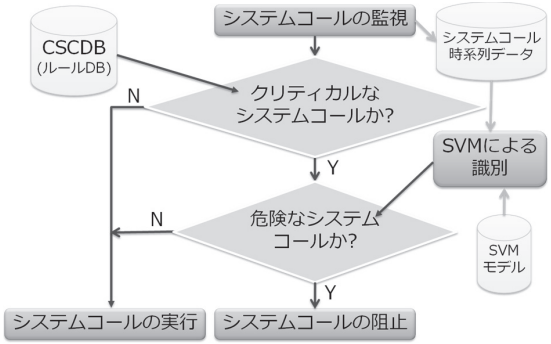


図 2　提案方式の流れ
Fig. 2　Flow of proposed method.

表 2　クリティカルなシステムコール一覧
Table 2　Critical system call list.

振る舞い	概要	システムコールブロック	システムコールの組合せ	引数
自己増殖のためのファイルコピー	自身をコピーするために Windows のシステムフォルダなどに書き込む	自己ファイルの読み込み	(NtCreateFile | NtOpenFile) → (NtReadFile | (NtCreateSection → NtMapViewOfSection))	システムフォルダやプログラムフォルダへのパス
		フォルダ内のファイルの検索	(NtCreateFile | NtOpenFile) → NtQueryDirectoryFile	
		ファイルの書き込み	(NtCreateFile | NtOpenFile) → NtWriteFile	
OS 起動時の自動実行	OS が再起動された際に自身を実行できるように自動実行に関わるレジストリへ書き込む	レジストリへの書き込み	(NtCreateKey | NtOpenKey) → NtSetValueKey	OSの起動に関係するレジストリキー
別プロセスの停止・起動	自身を守るために，ウイルス対策ソフト等の稼働中のプロセスを停止させたり別のプロセスを起動することによって自身の行動を隠そうとする	プロセスの操作	(NtTerminateProcess | NtCreateProcessEx)	他のプロセスのID
メモリへの書き込み	キーボードのログの取得等を目的として稼働している DLL 等にフックするためにメモリへの書き込みを行う	メモリの操作	NtWriteVirtualMemory	他のプロセスのメモリ領域等
ネットワークを経由した感染	ネットワークを経由して他のクライアントへの感染行動をする	ネットワーク操作	(NtSecureConnectPort → NtRequestWaitReplyPort) | ((NtCreateFile | NtOpenFile) →	ネットワーク上の他のホスト

引数を定義した．

なお，表 2 において，システムコールの組合せは，単独または複数のシステムコールの組合せによってシステムコールブロックを表すもので，(A | B) は，A または B のシステムコールを表し，A → B は A と B のシステムコールがファイルハンドルやレジストリハンドルなどのハンドルによって関連付けられることを表す．

すべてのプログラムが発行したシステムコールは，System Service Descriptor Table（SSDT）の書き換えを用いて監視され捕捉される．この捕捉されたシステムコールを用いて，CSCDB の検知ルールにより引数との組合せからクリティカルなシステムコールかどうかをチェックすることができる．SSDT には，各システムコールの処理ルーチンのアドレスや引数の数などが記述されている．ユーザモードのプログラムから発行されたシステムコールは，カーネル内においてシステムコールの番号をインデックスとして SSDT の要素を参照することにより，システムコールの処理ルーチンを実行する．したがって，デバイスドライバにおいてあらかじめ用意した関数のアドレスで SSDT のアドレスを置き換えることにより，カーネルのシステムコール処理ルーチンが実行をされる前に処理をフックして，各システムコールを監視することができる[6]．監視はルールベースで行うため，CPU への処理負荷は先行研究と同程度以下になる．

クリティカルなシステムコールと判定した場合は，当該システムコールに至る過去のシステムコールの時系列を用いて N-gram 法により特徴を表現した素性と，あらかじめ学習用データにより構築してある SVM のモデルを用いて SVM による識別を行う．その結果，クリティカルなシステムコールが悪意を持ったプログラムによって発行された危険なシステムコールであると判定された場合は，システムコールの実行を阻止する．具体的には，カーネルによるシステムコールの実行を回避し，呼び出し元の不正なプログラムにエラーを返すことでシステムコールを停止させる．また，この呼び出し元のプロセスが特定できているため，当該プログラムを強制終了させることもできる．

4.3　SVM による危険なシステムコールの認識

本論文では，ルールベースの識別によってクリティカルなシステムコールと判断されたシステムコールについて，そのシステムコールに至る過去のシステム発行履歴に基づいて危険なシステムコールかどうかを SVM を用いて識別する．使用した SVM は tinySVM[13] であり，カーネルには線形カーネル[⋆1]を用いた．なお，識別に用いる SVM のモデルは，あらかじめ通常のプログラムおよび不正なプログラムの学習用データによって学習を行い認識に用いるモデルを生成する．生成した SVM のモデルは，学習用プログラムが呼び出したシステムコールの時系列から N-gram を用いて特徴ベクトルを抽出して構築し，別のデータセットで評価しているので，環境にかかわらず共通に利用可能である．

SVM により識別を行うためには，システムコール発行履歴の時系列が持つ特徴を素性ベ

⋆1 文献 11) pp.39–pp.42

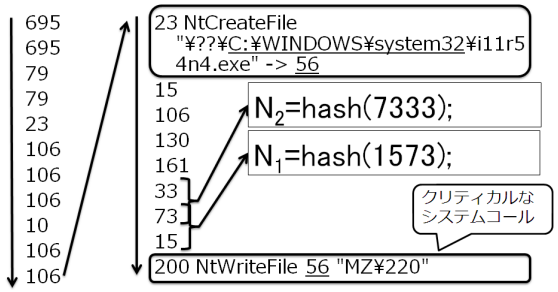


図 3　N-gram 法による素性データ
Fig. 3　Future data by N-gram method.

クトルとして表現する必要がある．提案方式では，素性ベクトルとして，言語モデルとして広く採用されている N-gram モデルを採用した．具体的には，システムコール時系列から得られた N-gram と N-gram の共起頻度によって特徴表現した．

図 3 に $N = 2$ の場合の N-gram モデルによる特徴表現の概要を示す．クリティカルなシステムコールが検知された場合，それ以前のシステムコール時系列より N-gram 法を用いて N-gram 集合を抽出する．抽出したそれぞれの N-gram は，ハッシュ関数を用いて要素番号 N_i に変換する．次に，変換した要素番号 N_i について重複するものの頻度 h_i を求める．素性ベクトルは，要素番号 N_i を昇順にソートし，対応する頻度 h_i をペアにすることで生成する．図 3 から生成された素性ベクトルは

$$f = \{N1:1,\ N2:1,\ N3:1,\ N4:1,\ N5:1,\ N6:1,\ N7:1,\ N8:1,\ N9:1,$$
$$N10:1,\ N11:3,\ N12:1,\ N13:1,\ N14:1,\ N15:1,\ N16:1\}$$

となる．通常のプログラムの実行ログから得られた素性ベクトルを正例とし不正なプログラムの実行ログから得られた素性ベクトルを負例とした．なお，特徴ベクトルの次元数が大きく，また疎で冗長なデータ空間となるため，次元数を減らして密なデータ空間に変換するためにハッシュ関数を用いている．

5. 評 価 実 験

提案方式の有効性を検証するために評価実験を行った．評価実験では，CSCDB に基づいたルールベースによるクリティカルなシステムコールの検知性能と SVM による危険なシステムコール識別性能を確認することを目的としている．なお，本論文では，提案方式の有効性を検証するために，あらかじめシステムコールの監視により取得した通常のプログラムと不正なプログラムのログファイルに対してバッチテストで行った．

表 3　実験用システム構成
Table 3　Experimental system configuration.

仮想環境	Virtual PC 2007
ホスト OS	Windows XP SP2
ゲスト OS	Windows XP SP2 × 2 Vine Linux 4.2（DNS サーバ）
ネットワーク環境	ゲスト OS 間のみ

5.1　実験用データセット

バッチテストで用いるデータセットを取得するために通常のプログラムおよび不正なプログラムを後述するデータセット収集環境において実行し，システムコールの履歴を取得した．学習用データセットとして通常のプログラム 120 種類，不正なプログラム 120 種類を取得した．評価用データセットとして通常のプログラム 60 種類，不正なプログラム 60 種類を取得した．

不正なプログラムは，実際に流通したウイルス，ワーム，トロイの木馬，ボットなどである．また，いくつかの不正なプログラムについては，数種類の亜種も用意した．

通常のプログラムは，ソフトウェア配布サイトから無作為にダウンロードしたインストーラプログラムとした．評価実験では，通常のプログラムとして，不正なプログラムと挙動が似ているインストーラソフトウェアを用いているが，これは，ルールベースの検知で発生する False Positive の割合を高め，SVM による識別により False Positive の割合が減少するかどうかを確認するためである．

5.2　システムコール履歴の収集

学習用データセットおよび認識用データセット用のサンプルとして用意した通常のプログラムおよび不正なプログラムを**表 3** に示す仮想環境において実行し，システムコールの履歴を取得した．Windows XP SP2 をホスト OS とし，Virtual PC 2007 による仮想環境を構築し，ゲスト OS として Windows XP SP2 を 2 セットと DNS サーバとして Vine Linux を 1 セットの合計 3 セットの OS を動作させた．Windows XP（ゲスト OS）の 1 つにおいて Strace for NT を用いて，インストーラプログラムなどの通常のソフトウェアと，ワーム，ウイルス，トロイの木馬などの不正なプログラムを起動し，システムコールのログを採取した．もう 1 つの Windows XP（ゲスト OS）はネットワークを用いて増殖を行うタ

表 4　クリティカルなシステムコール検知結果
Table 4　Critical system call detection result.

通常のプログラム		不正なプログラム	
学習用	評価用	学習用	評価用
95.8%	96.7%	98.3%	100.0%

イプのワームなどのターゲットにするためと，ネットワーク上を流れるデータを取得するために用いた．Vine Linux については，不正なプログラムが発行する DNS クエリーに応えるための DNS サーバとして用いている．また，安全のためにネットワーク環境はゲスト OS 間のみで行われるように Virtual PC において設定を行った．なお，仮想環境を用いたのは，不正なプログラムを実行するため，毎回環境を復元する必要があるためである．

5.3　SVM で使用するモデルの構築

SVM を用いて，危険なシステムコールの識別を行うためには，あらかじめ SVM で使用するモデルを作成しておく必要がある．そのため，取得した通常のプログラムおよび不正なプログラムの学習用データセットからすべてのクリティカルなシステムコールを CSCDB により検知し，それ以前のシステムコールの履歴から N-gram 法により生成した素性ベクトルを SVM に与えて学習を行い，識別に用いるモデルを構築した．

5.4　CSCDB の検知性能に関する実験

提案方式によるクリティカルなシステムコールの検知率を調べるために取得したシステムコール列を用いて実験を行った．実験の結果を**表 4** に示す．実験は，不正なプログラムと通常のプログラムのすべてのログについて CSCDB に基づく検知ルールにより，クリティカルなシステムコールが発行されているかどうかを調査したものである．実験結果から，不正なプログラムが発行したクリティカルなシステムコールを学習用データについて 98.3%，評価用データについて 100.0%検知することができた．また，通常のプログラムが発行したクリティカルなシステムコールを学習用データで 95.8%，評価用データでは 96.7%検知している．実験では，通常のプログラムとしてインストーラプログラムによるログを用いているため，クリティカルなシステムコールの検知率が高くなっている．これは，False Positive の割合がかなり高いことを示しており，このことから不正なプログラムの挙動とインストーラプログラムの挙動は様々な面で類似性が高く，ルールベースのみで識別するのは困難であることが分かる．なお，不正なプログラムで検知できなかったものについてログを解析した結果，異常終了などにより，システムに影響を与える動作を行わずにプログラムが終了していることが判明した．

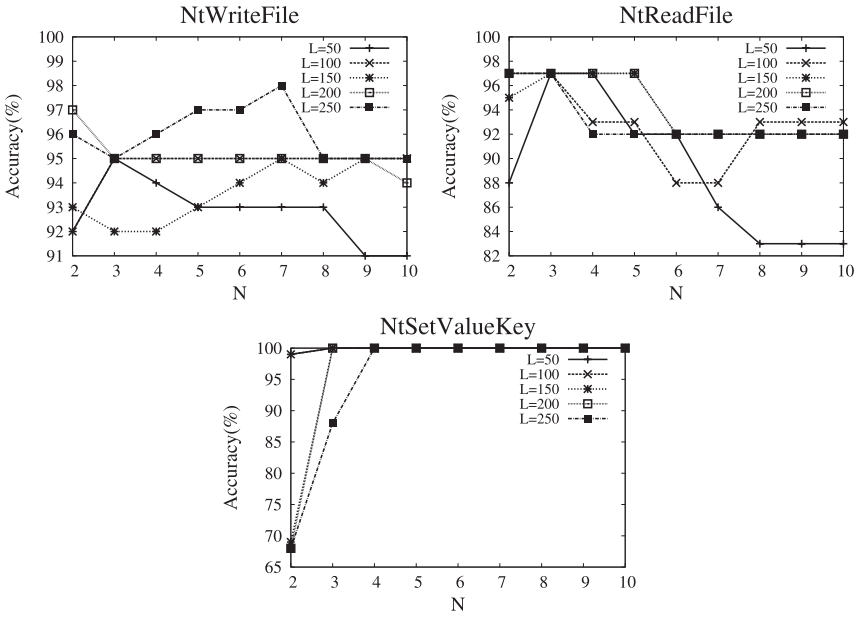


図 4　時系列の長さ L と N-gram の N を変えた SVM による認識結果
Fig. 4　Recognition result by SVM that changes length L of time series and N of N-gram.

5.5　SVM の識別性能に関する実験

クリティカルなシステムコールが検知された際に，それ以前のシステムコールの履歴から N-gram 法を用いて素性ベクトルを生成するが，その際，パラメータとして，N-gram の大きさ N とシステムコール履歴の時系列の長さ L を決定する必要がある．SVM に与える素性ベクトルを生成する際に使用するパラメータの決定と，SVM の識別性能を評価するために，実験を行った．実験の結果を**図 4** に示す．

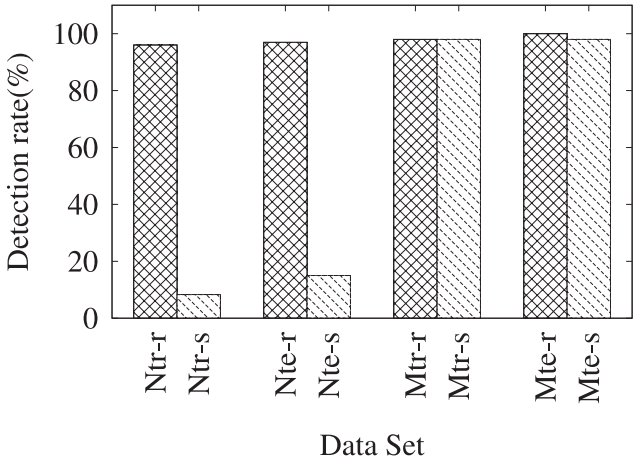
実験では，N と L を変化させ，学習用データセットのすべてのクリティカルなシステムコールについて素性ベクトルを生成し，SVM のモデルを構築した．構築した SVM のモデルを用いて，評価用データセットのすべてのクリティカルなシステムコールについて，危険なシステムコールの識別を行った．グラフから，素性ベクトルを生成する際に用いるパラメータ N と L については，システムコールによって相違があるため，実験の結果から，$N = 5$ および $L = 200$ を用いることにした．

5.6　不正なプログラムの検知結果

本論文で提案する手法を用いて，クリティカルなシステムコールを発行したプログラムが不正なプログラムかどうかを SVM を用いて識別する実験を行った．実験結果を**図 5** に示す．

図 5 から，ルールによって検知した不正なプログラムが発行したクリティカルなシステムコールを，学習用データセットについては 100.0%，評価用データセットについては 98.3%，危険なシステムコールとして識別していることが分かる．この値は，まだ出現していない未知の不正なプログラムに対する識別率であり，ある意味で，不正なプログラムを予測しているといえるため，現時点では高い識別率だと考える．また，通常のプログラムについては，学習用データセットについては，クリティカルなシステムコールとして 95.8%検知したものが，SVM の識別により危険なシステムコールとして識別したものが 8.3%に減少した．評価用データセットについても，96.7%だったものが 15.0%に減少した．

実験結果から，不正なプログラムの検知率が高く，通常のプログラムを誤検知する False Positive の割合が減少していることが分かり，提案方式の有効性が高いことがいえる．また，ルールベースで検知するクリティカルなシステムコールが，通常使用するアプリケーションにおいて発生する頻度は非常に小さい．また，日常におけるインストーラープログラムの実行頻度も少ない．そのため，実験結果から，通常のプログラムを不正なプログラムに誤検知する 1 日あたりの頻度は，きわめて小さい．



N: normal program　　M: malicious program
tr: training data　　te: test data
r: rule　　s: SVM

図 5　提案方式による不正なプログラムの検知結果
Fig. 5　Detection result of malicious program by proposal method.

5.7　考　　察

Windows 系 OS において，システムコールによる振舞いを SVM により異常検知するシステムがいくつか提案されている．文献 14) の研究は，Windows レジストリアクセスについて SVM を用いた異常検知手法を提案している．これに対し，本研究の方式は，Windows 系 OS のシステムコール全般を用いているため，レジストリアクセスをともなわない不正なプログラムについても検出することができる．文献 15) の研究は，Forrest らによって提案された，UNIX 系 OS におけるシステムコール列の N-gram を用いた侵入検知を Windows 系 OS において実現した．通常および不正なプログラムの学習用 N-gram データを用いて SVM のモデルを構築し，テスト対象プログラムのシステムコールの時系列から抽出した N-gram データを SVM によって識別する方法を提案している．そのため，テスト対象プログラムがシステムコールを発行するつど，SVM による識別を行い，不正なプログラムと判別するまで実行する．それに対し，本研究は，クリティカルなシステムコールをルールによって検知し，検知したシステムコールが危険なシステムコールかどうかを SVM によって識別しているため，SVM による識別の回数を減らすことができる．また，SVM に用いる素性データの生成方法にも相違があり，本研究では効率化を図っている．文献 16) の研究は，Windows 系の異常検知をユーザモード上の WIN32API のシステムコール時系列に対して，ラフ集合により特徴を抽出し，SVM を用いて識別を行う手法を提案している．それに対し，本研究はカーネルモード上のシステムコールを用いて，ルールベースの検知と SVM による識別を組み合わせているので，より確実に検知することができる．

6. ま と め

Windows における危険なシステムコールに着目した侵入検知を行うシステムを提案した．提案方式では，ルールを用いて，OS の資源に影響を与えるクリティカルなシステムコールを検知する．さらに，検知したクリティカルなシステムコールを，Support Vector Machine（SVM）を用いることで，不正なプログラムの発行した危険なシステムコールかどうか認識し，通常のプログラムと不正なプログラムを区別する．実験による評価の結果，従来のルールベースによる検知では false positive になる割合が高かったインストーラの false positive

率を減少させることができ，提案方式の有効性を確認できた．

今後の課題として，SVM に与える素性の検討による検知率の向上とデバイスドライバとして実装した異常検知システムの構築があげられる．

参 考 文 献

1) 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
2) eWeek: IE Under Attack: Microsoft Ponders Emergency Patch (2006). http://www.eweek.com/article2/0,1895,1942566,00.asp
3) Forrest, S., Hofmeyr, S.A., Somayaji, A. and Longstaff, T.A.: A sense of self for Unix processes, *IEEE Symposium on Security and Privacy*, pp.120–128 (1996).
4) Sekar, R., Bendre, M., Dhurjati, D. and Bollineni, P.: A Fast Automaton-Based Method for Detecting Anomalous Program Behaviors, *IEEE Symposium on Security and Privacy*, pp.144–155 (2001).
5) Feng, H.H., Kolesnikov, O.M., Fogla, P. and W. Lee, W.G.: Anomaly Detection Using Call Stack Information, *IEEE Symposium on Security and Privacy*, pp.62–75 (2003).
6) 島本大輔，大山恵弘，米澤明憲：System Service 監視による Windows 向け異常検知システム機構，情報処理学会論文誌：コンピューティングシステム，Vol.47, No.SIG 12 (ACS 15), pp.420–429 (2006).
7) Battistoni, R., Gabrielli, E. and Mancini, L.V.: A Host Intrusion Prevention System for Windows Operating Systems, *Proc. 9th European Symposium on Research in Computer Security* (*ESORICS 2004*), pp.352–368 (2004).
8) Nebbett, G.：WindowsNT・2000 ネイティブ API リファレンス，ピアソン・エデュケーション (2000).
9) Schreiber, S.: *Undocumented Windows 2000 Secrets: A Programmer's Cookbook*, Addison-Wesley Professional (2001).
10) Skormin, V., Volynkin, A., Summerville, D. and Moronski, J.: Prevention of information attacks by run-time detection of self-replication in computer codes, *Journal of Computer Security*, Vol.15, No.2, pp.273–302 (2007).
11) Cristianini, N. and Shawe-Taylor, J.：サポートベクターマシン入門，共立出版 (2005).
12) 松本隆明，鈴木功一，高見知寛，馬場達也，前田秀介，水野忠則，西垣正勝：自己ファイル READ の検出による未知ワームの検知方式，情報処理学会論文誌，Vol.48, No.9, pp.3174–3182 (20070915).
13) 工藤　拓：TinySVM: Support Vector Machines. http://www.chasen.org/~taku/software/TinySVM/
14) Zhang, X., Gu, C. and Lin, J.: Support Vector Machines for Anomaly Detection, *The 6th World Congress Intelligent Control and Automation* (*WCICA 2006*), pp.2594–2598 (2006).
15) Wang, M., Zhang, C. and Yu, J.: Native API Based Windows Anomaly Intrusion Detection Method Using SVM, *Proc. IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing -Vol 1* (*SUTC'06*), Washington, DC, USA, IEEE Computer Society, pp.514–519 (2006).
16) Zhang, B., Yin, J., Tang, W., Hao, J. and Zhang, D.: Unknown Malicious Codes Detection Based on Rough Set Theory and Support Vector Machine, *International Joint Conference Neural Networks* (*IJCNN '06*), pp.2583–2587 (2006).

(平成 20 年 11 月 30 日受付)
(平成 21 年 6 月 4 日採録)

伊波　靖（正会員）

1961 年生まれ．1984 年琉球大学工学部電子・情報工学科卒業．同年沖縄県立八重山商工高等学校電気科教諭．1998 年琉球大学大学院工学研究科修士課程電気・情報工学専攻修了．2004 年沖縄工業高等専門学校メディア情報工学科講師，2007 年同准教授，2009 年同情報処理センター長．現在，琉球大学大学院理工学研究科博士後期課程総合知能工学専攻在学中．修士（工学）．情報セキュリティとコンピュータフォレンジックスに関する研究に従事．

高良　富夫（正会員）

1952 年生まれ．1976 年鹿児島大学理学部物理学科卒業．1979 年東京工業大学総合理工学研究科博士前期課程，1983 年後期課程物理情報工学専攻修了，工学博士．1991〜92 年米国カーネギー・メロン大学客員研究員．1995 年琉球大学工学部情報工学科教授，2002 年総合情報処理センター長，2006 年学長補佐．日本音響学会，電子情報通信学会，IEEE 各正員．2006 年日本音響学会九州支部長．1990 年沖縄研究奨励賞．共著：『やわらかい南の学と思想』（2008 年），『沖縄の先端技術』（1988 年）．音声の分析，合成，認識，獲得，言語処理，琉球語とアジアの言語の音声分析と合成の研究に従事．