Windows上における危険な処理の承認機構の提案

早川 顕太[1]　鈴木 秀和[1]　旭 健作[1]　渡邊 晃[1]

概要：マルウェアは多様化が進み，不正インストールやスパムメールの送信，情報漏えいといった様々な活動を行う．これらの活動はバックグラウンドで行われるため，ユーザがその危険な処理に気づくことができないという課題がある．本稿では，Windows 上において危険な処理の動的なユーザへの承認機構を提案する．プログラムが発行する Windows API を提案システムがフックすることにより，危険な処理が行われる直前にユーザへ承認ダイアログを表示する．これにより，マルウェアがバックグラウンドで行う危険な処理を，ユーザは自身の意図しないものとして拒否することが可能となる．本稿では，さらに，提案方式のプロトタイプシステムを実装し，正規ソフトウェアや独自に入手したマルウェアを用いて実験を行った．そして，実験によって得られた結果や既存技術との比較から提案方式の有用性を示す．

A Proposal of Approval Mechanisms for Dangerous Processing on Windows

Hayakawa Kenta[1]　Suzuki Hidekazu[1]　Asahi Kensaku[1]　Watanabe Akira[1]

1. はじめに

インターネットの急速な発展にともない，マルウェアによる被害がますます増加している．初期のマルウェアは自己顕示を目的として開発され，その活動はシステムの破壊や悪意のあるポップアップの表示等，ユーザから目に見えて分かるものが多かった．一方，現在は金銭を目的としたマルウェアの開発に移り変わっている．マルウェアは，不正インストールによりシステムに駐在し，バックドアによる遠隔操作により DDoS 攻撃やスパムメールの送信，情報漏えい等の活動を行う．これらの活動は，初期のマルウェアの活動と異なり，表面化されることがない．攻撃者はこれらのマルウェアを利用して，企業へ強迫を行い身代金を要求したり，クレジットカード等の暗証番号を盗み出すことで，不正に金銭を入手する．これら多様化したマルウェアの活動はバックグランドで行われるため，ユーザは自身が被害に遭う前に，その活動を認識・防止することができないという課題がある．本論文の目的は，これらマルウェアがバックグランドで行う活動を防止することである．

現在，マルウェアを検出する最も一般的な手法としてパターンマッチング法がある．パターンマッチング法は，事前にマルウェアの特徴的なシグネチャを定義しておき，実行ファイルを静的に検査することにより，そのシグネチャを含むプログラムをマルウェアとして検出する手法である．このため，既知のマルウェアについては，高い精度で検出することができ，なおかつ誤検知も発生しにくい手法である．しかしながら，シグネチャが定まらない未知マルウェアを検出できないという課題がある．

未知マルウェアを検出する手法として，事前にマルウェアらしい振る舞いを定義して，それを検出するヒューリスティック法がある．ヒューリスティック法は，実行ファイル内のコードを解析することにより静的に振る舞いを検出する静的ヒューリスティック法と，実際にマルウェアを動作させた上で動的に振る舞いを検出するビヘイビア法（動的ヒューリスティック法）がある．マルウェア開発者はパッカーと呼ばれる実行ファイルを実行可能なまま圧縮するツールを用いることにより，既知マルウェアから容易に暗号化・難読化された亜種マルウェアを作成することができる．特に独自に開発されたパッカーによりマルウェアが暗号化・難読化された場合，静的ヒューリスティック法においては，コードの解析が困難となり，これらのマルウェアを検出できない．これに対し，ビヘイビア法においては，コードが暗号化・難読化されても動作上の振る舞いは変化

[1] 名城大学院理工学研究科情報工学専攻

しないため，これらのマルウェアを検出することができるという利点がある．

しかしながら，ヒューリスティック法の共通課題として，そもそもマルウェアらしい振る舞いを定義することが難しいという課題がある．マルウェアの多様化により，全てのマルウェアを網羅的に検出可能な共通した振る舞いを定義することができない．さらに，マルウェアの個々の活動を検出しようとしても，その多くは正規ソフトウェアにおいても類似した活動が存在するため，マルウェアと正規ソフトウェアの分離が難しい．このことが，ヒューリスティック法において誤検知が絶えない原因となっている．

本論文では，Windows 上において危険な処理のユーザへの承認機構を提案する．これにより，マルウェアがバックグラウンドで行う危険な処理を，ユーザは自身の意図しないものとして拒否することが可能となる．提案方式が承認機構という方式を取ることで，正規ソフトウェアにおいても観測される複数の活動を検出対象にできる．さらに，マルウェアと正規ソフトウェアの分離には，ユーザの判断を借りることにより誤検知を少なくすることができる．

以下，2 章でビヘイビア法と承認機構に関する既存技術を紹介し，それらの課題を述べる．3 章では，承認機構である提案方式を述べ，4 章でそのプロトタイプシステムの実装方法について述べる．5 章では，実験による提案方式の有効性を検証と定性評価による既存技術との比較を行う．最後に 6 章でまとめを行う．

2. 既存技術

提案方式をマルウェア検知手法として見た場合，ビヘイビア法に分類される．従って，マルウェア検知手法の既存技術として，ビヘイビア法を取り上げる．また，提案方式は承認機構であるため，承認機構に関する既存技術を取り上げる．

2.1 ビヘイビア法に関する既存技術

ビヘイビア法は未知・暗号化（難読化）マルウェアを検出可能であるが，以下に示す課題がある．

- （課題 1）振る舞い定義の難しさ
  マルウェアは様々な活動を行うため，全てのマルウェアを網羅的に検出できるような振る舞いを一概に定義することができない．
- （課題 2）正規ソフトウェアとマルウェアとの分離
  マルウェアによる多くの活動は正規ソフトウェアにおいても類似した活動が存在するため，マルウェアと正規ソフトウェアの分離が難しい．

ビヘイビア法はマルウェアの振る舞いを定義する方法により，「マルウェアの特徴的な振る舞いを検出する手法」と「機械学習を用いた検出手法」の 2 つの手法に分類できる．以下に各手法の研究例とその課題を示す．

2.1.1 マルウェアの特徴的な振る舞いを検出する手法

この手法は，研究者の考察と実験により，ある種のマルウェアに特徴的な振る舞いを定式化し，それをビヘイビア法の検出対象の振る舞いとして定義する手法である．論文 [1] はワームが自己複製を行う際に自身の実行ファイルを READ する必要があるため，このワームに特徴的な挙動を検出する手法である．論文 [2] は，PC へインストールされたワーム（ボット）は実行環境を復元して再度実行することにより，インストール挙動（実行ファイルの作成と自動実行への登録）を反復するという特徴的な挙動をとるため，これを検出する手法である．論文 [3] は，Windows 上においてキーロガーに特徴的であるキー入力を取得するという挙動を網羅的に定式化し検出する．近年のマルウェアは，セキュリティソフトを強制終了する等のセキュリティ無効化攻撃を行うものや，デバッカを検知し活動を抑制する等の耐解析機能を持つものがある．これらのマルウェアの機能を逆用して，マルウェアを検知，あるいは活動を防止する研究が存在する [4][5]．論文 [6] では，マルウェアが解析者による解析を妨害するために，生成するファイル名などをランダムに変化させるという特徴的な挙動を，実行毎の API ログを比較することにより検出する．

これらの既存研究は，いずれもビヘイビア法の課題 1 のために，マルウェアの個々の活動を検出対象にした手法である．そのため，検出対象となるマルウェアの範囲はその活動を行うマルウェアに限定されてしまう．また，研究にもよるがビヘイビア法の課題 2 のために，完全にマルウェアと正規ソフトウェアを分離できているわけではない．

2.1.2 機械学習を用いたマルウェア検出手法

この手法は，ビヘイビア法の振る舞い定義にシステムコールの発行履歴による機械学習を用いる手法である．機械学習を用いることにより，マルウェアに特徴的な全ての振る舞いを検出対象にできるため，ビヘイビア法の課題 1 を解決している．論文 [7] は，ホワイトリスト方式による異常検知であり，事前に正常なソフトウェアにおいてシステムコールの履歴を N-gram 法で表現し機械学習する．実環境において，システムコールが呼ばれた際，最近 N の長さのシステムコールの履歴を参照しそれが学習されていなければ，それを異常として検知する手法である．論文 [8] は，システムの可用性に影響を与えるクリティカルなシステムコールを定義しており，事前に正規ソフトウェアとマルウェアの両方において，クリティカルなシステムコールが呼び出される直前のシステムコールの履歴（N-gram により表現される）を SVM(Support Vector Machine) によって教師あり機械学習する．実環境では，クリティカルなシステムコールの呼び出しをトリガーとして SVM による識別を行いマルウェアを検知する手法である．

これらの既存研究は，ビヘイビア法の課題 2 のため，いずれもある程度の誤検知が生じてしまうという課題があ

る．また，機械学習によりマルウェアを検知するため，検出時，なぜそのプログラムがマルウェアとして検出されたのかをユーザに説明できないといった課題がある．

2.2 承認機構に関する既存技術

現在，著者が知る限り Windows 上において動的な承認機構を提供するといった研究はない．承認機構に関連した既存技術としては，Windows Vista 以降に導入されたユーザアクセス制御（UAC ; User Access Control）が挙げられる．UAC により，例え管理者のユーザとしてログインしても，昇格プロンプトによるユーザの承認を得ない限り，標準ユーザの権限として動作する．これにより，マルウェアがシステム全体に悪意のある変更を加えることを防止できる．

しかし，UAC はプログラムの起動時に行われる承認機構であり，プログラムの実行中に行われる動的な承認機構ではない．従って，昇格プロンプトを表示した時点では，実際に行われる処理の内容やそのタイミングをユーザが把握することができない．また，標準ユーザの権限で行える，カレントユーザのみへのシステムの変更や，メールの送信などを防ぐことができないといった課題がある．

3. 提案方式

3.1 概要

本稿では，Windows 上における動的な危険な処理のユーザへの承認機構を提案する．図 1 に提案システムの概要を示す．アプリケーションが危険な処理を行うために発行する Windows API を提案システムがフックすることにより，その危険な処理が行われる直前に，ユーザへの承認ダイアログを表示する．ユーザは行われていようとしている危険な処理が自分の意図したものであるかどうかによって，その処理の許可／拒否を選択する．ユーザの応答により，提案システムはその処理を続行するか，あるいは処理を中断させアプリケーションにエラーを返す．これにより，マルウェアがバックグラウンドで行う危険な処理を，ユーザは自身の意図しないものとして拒否することが可能になる．

提案方式が承認機構として，満たすべき 4 つの要件は以下の通りである．

＜要件 1＞ 承認対象となる処理は，マルウェアの悪意のある活動であること．

＜要件 2＞ ユーザは処理を許可／拒否するための正しい判断が可能であること．

＜要件 3＞ 承認ダイアログ内に，ユーザの理解の助けとなる情報を提示すること．

＜要件 4＞ 承認機構により，ユーザビリティが著しく損なわれないようにすること．

これらの要件を満たすように，提案方式の詳細を以下に述べる．

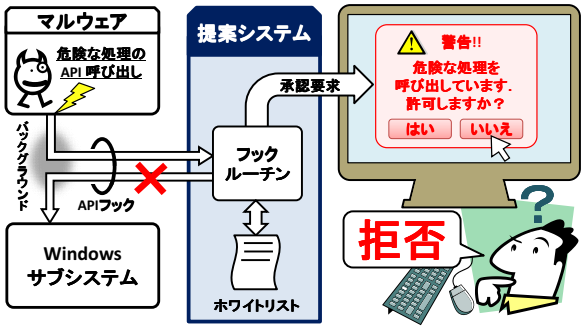

図 1 提案方式の概要

表 1 危険な処理の一覧

危険な処理	悪用された場合の被害
実行ファイルの作成	不正インストール
自動実行への登録	不正インストール
他プロセスの強制終了	セキュリティの無効化
キー入力の取得	キーロギング
メール送信	スパムメールの送信
スレッドの注入	他プロセスへの感染

3.2 危険な処理の定義

提案方式の＜要件 1＞と＜要件 2＞を満たすように，危険な処理を定義する．＜要件 1＞により，マルウェアによって悪用されることがない処理に対して，承認を求める必要がない．＜要件 2＞は，承認ダイアログが表示された際，その行われようとしている処理がユーザに身に覚えのないものならば，ユーザはそれを拒否できること（すなわち，そのプログラムがマルウェアであること）である．この要件を満たしていれば，提案方式はユーザによる誤検知を少なくできる．この要件を満たすために，承認対象とする処理が次の条件を満たす必要がある．

「危険な処理の条件：正規ソフトウェアはユーザが意図したタイミングのみにその処理を行う．」

これは，つまり正規ソフトウェアが勝手に行ってしまう可能性がある処理を承認対象とすべきではないということである．このような処理を承認対象にしてしまうと，承認ダイアログが表示された際，ユーザはその処理がマルウェアによるものなのか正規ソフトウェアによるものか判別できずに，誤検知を起こしてしまう可能性がある．

以上の考察から，これらの要件を満たすものとして危険な処理を表 1 のように定義した．表 1 内の各処理は，マルウェアによって悪用されることが想定されるため，これらの処理は＜要件 1＞を満たしていると考えられる．＜要件 2＞については実験による確認が必要である．

3.3 承認ダイアログについて

提案方式は表 3.2 に示した危険な処理を検出し，その処理が行われる直前に，ユーザへの承認ダイアログを表示す

図 2 承認ダイアログのイメージ図

る．承認ダイアログのイメージ図を図 2 に示す．承認ダイアログ内には以下の情報を表示する．

(1) プロセスに関する情報
- プロセス名
- プロセス ID
- 実行ファイルの絶対パス
- 実行ファイルのアイコン
- 電子署名の発行元

(2) 行おうとしている危険な処理の内容

(3) プロセスが表示しているウインドウ（複数存在する可能性あり）

(4) ユーザの選択肢

上記のうち，(1)〜(3) は提案方式の＜要件 3 ＞を満たすための情報である．(2) は現在行われようとしている危険な処理の内容であり，ユーザはこれが自身の意図したものであるかどうかによって，その処理の許可／拒否を判断する．(1) はユーザが危険な処理を呼び出しているプロセスを一意に特定するための情報である．これに加え，(3) を表示することによりユーザはそのプロセスを直観的に把握することができる．承認ダイアログが表示されると，ユーザはその後の処理を (4) から選択する．表 2 に選択肢とその選択場面，選択後の提案システムの動作を示す．

3.4 ユーザビリティの向上

提案方式は承認ダイアログにおいて，「ホワイトリストへ登録」や「この実行に限り常に許可」を選択することより，承認ダイアログを永続的に，あるいは一時的に省略できるようにする．これにより，ユーザビリティを向上させ，提案方式の＜要件 4 ＞を満たすようにする．

ユーザは承認ダイアログの「ホワイトリストへ登録」を選択することにより，そのプログラムをホワイトリストへ登録する．ホワイトリストへ登録されたプログラムは，OS やプログラムを再起動しても，その処理に関して承認ダイアログが永続的に省略される．ホワイトリストにはプログラムの絶対パスと常に許可したい危険な処理（複数可）が登録される．表 3 にホワイトリストの例を示す．例えば，Windows エクスプローラはユーザのドラック&ドロップにより実行ファイルをコピーすることがあるため，実行ファイルの作成をホワイトリストに登録する．

表 3 ホワイトリストの例

登録されたプログラム	許可された処理
C:/Windows/explorer.exe	実行ファイルの作成
C:/〜(省略)〜/iexplore.exe	実行ファイルの作成
C:/Windows/System32/taskmgr.exe	他プロセスの強制終了
C:/〜(省略)〜/Microsoft Outlook 2010	メール送信

ユーザは承認ダイアログの「この実行に限り常に許可」を選択することにより，そのプロセスはプロセスが終了するまで，その処理に関して承認ダイアログを一時的に省略することができる．「この実行に限り常に許可」はインストーラでの使用を想定している．インストーラは，通常複数の実行ファイルを作成するため，その都度承認ダイアログを表示していてはユーザビリティを損ねる．しかし，通常，インストーラは一度実行し，その後捨ててしまうためホワイトリストへ登録することも適切ではない．そこで，「この実行に限り常に許可」を選択することにより適切な許可を与えることができる．

4. 実装

提案方式の有用性を評価するため，プロタイプシステムを実装した．そこで，本章ではプロトタイプシステムの実装の詳細を述べる．

4.1 API フックの方法

提案方式は API フックを用いて実装を行う．API フックには複数の方法あるが，プロトタイプシステムの実装には，実装が容易で，比較的フックの回避が難しい Detours ライブラリを採用した．Detours ライブラリとは MicrosoftReserch が提供する API フックライブラリである [9]．Detours ライブラリは，プロセスの仮想メモリ上にロードされたフック対象の API の先頭の命令をフック関数への JMP 命令に書き換えることでフックを行う．

表 2　承認ダイアログにおけるユーザの選択肢

選択肢	選択場面	提案システムの動作
許可	ユーザがその処理を自身で意図した場合	その処理を一度だけ許可
拒否	ユーザがその処理に身に覚えがない場合	その処理を拒否
強制終了	ユーザがそのプログラムをマルウェアと判断した場合	そのプロセスを強制終了
この実行に限り常に許可	ユーザがそのプログラムを安全であると判断した場合	そのプロセスの終了時まで，その処理を一時的に許可し，承認ダイアログを省略
ホワイトリストへ登録	ユーザがそのプログラムを安全であると判断した場合	OS やプログラムの再起動後も，そのプログラムが行うその処理を永久的に許可し，承認ダイアログを省略

4.2　プロトタイプシステムの構成

プロトタイプシステムは，駐在プロセスである DllInjector とフック用 DLL（Hooker.dll）の 2 つの 32 ビット版の実行可能ファイルから構成される．図 3 にプロトタイプシステムの全体像を示す．以下にプロトタイプシステムの各要素を説明する．

- フック用 DLL（Hooker.dll）：フック用 DLL は監視対象となる全てのプロセスで，強制的にロードされる DLL である．フック用 DLL には，フック対象となる API 毎にフックルーチンを実装する．DLL の初期化関数内で Detours ライブラリを呼び出し，フック対象の API をそれに対応するフックルーチンへとフックする．フック対象となる API は，まず危険な処理を行う API であり，その詳細は 4.3 節で述べる．また，この他にも子プロセスを生成する API である CreateProcess API をフックしており，そのフックルーチン内で代わりに Detours ライブラリの DetourCreateProcessWithDll 関数を呼び出す．この関数は指定した DLL を注入した状態でプログラムを起動する関数である．これにより，既にフック用 DLL が注入済みのプロセスから CreateProcess API によりプロセスが生成される場合，子プロセスの起動直後からフック用 DLL の注入が行える．
- DllInjector : CreateProcess API 以外の API で生成されたプロセスなどは，フック用 DLL がロードされていない．DllInjector はこれらのプロセスに強制的にフック用 DLL を注入する役割を持つ駐在プロセスである．DllInjector は OS 起動時から管理者権限でシステムに駐在し，動作中の各プロセスを監視する．そして，定期的に各プロセスにフック用 DLL が注入されているかを調査し，注入されていない場合，そのプロセスへ DLL インジェクションを行う．DLL インジェクションには RemoteCreateThread API と LoadLibrary API を組み合わせた方法を採用した．

Windows Vista 以降，セッション ID=0 で動作するシステムプロセスなどは GUI（つまり，承認ダイアログ）を表示することができない．このため，プロトタイプシステムでは，カレントユーザが所有するプロセスのみを監視対象としており，これにより検知漏れが生じることが判明している．典型的な例として，Windows Installer サービスを利用してソフトウェアをインストールした場合，実行ファイルの作成がシステムプロセスによって行われるため，実行ファイルの作成を検出できないといった検知漏れが生じる．現在のプロトタイプシステムは，提案方式の有用性を評価するためのものである．このため，現状はマルウェア開発者がプロトタイプシステムの実装を理解した上で回避策を立てることや，プロトタイプシステムが Windows システムに与える負荷を考慮しているわけではない．今後は検知漏れや実用上のマルウェアに対する堅牢さを考慮して，実装方法をカーネルレベルでシステムコールのフックを行うなどの対策が必要である．

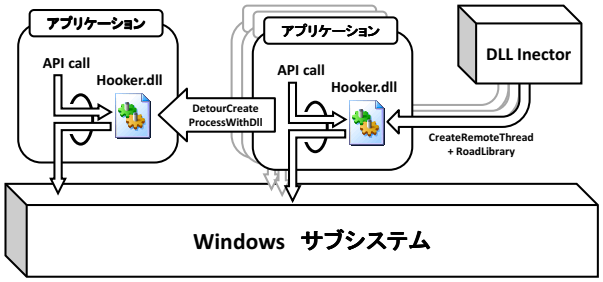

図 3　プロタイプシステムの構成図

4.3　危険な処理として検出する API

本節では危険な処理として定義した表 1 に対して，実際にどの API をフックするかを述べる．現在は表 1 の内，実行ファイルの作成と OS 起動時の自動実行への登録が実装済みである．それ以外の危険な処理は，今後，検討・実装を行う予定である．実行ファイルの作成と自動実行への登録を検出するために，フックする API を表 4 に示す．各々の危険な処理について，そのフック対象となる API が指定した引数の条件を満たして呼ばれた際に，簡易的な承認ダイアログを表示する．OS 起動時の自動実行への登録に関しては，その方法にレジストリ，スタートアップフォルダ，タスクスケジューラによる複数の方法がある．プロトタイプシステムではマルウェアによってよく利用されるレジストリを用いた方法のみをターゲットとした．表 5 に監視対象にした自動実行に関するレジストリエントリを示す．

表 4　危険な処理に対して検出対象となる API

危険な処理	フック対象となる API	API の引数の条件
実行ファイルの作成	NtCreateFile	ファイルを新たに生成し，そのファイルの拡張子が”.exe”である場合
	NtSetInformationFile	操作がファイル名のリネームで，リネーム後のファイルの拡張子が”.exe”である場合（ただし，リネーム前のファイルの拡張子が”.exe”の場合を除く）
OS 起動時の自動実行への登録	NtSetValueKey	キー名やエントリ名に表 5 示した文字列を含むレジストリエントリへ書き込みを行う場合

表 5　検出対象とした自動実行に関するレジストリエントリ

キー名	エントリ名
SOFTWARE/Microsoft/Windows/CurrentVersion/Run	—
SOFTWARE/Microsoft/Windows NT/CurrentVersion/Winlogon	Shell
SOFTWARE/Microsoft/Windows NT/CurrentVersion/Winlogon	Userinit
SOFTWARE/Microsoft/Active Setup/Installed Components	StubPath
SYSTEM/ControlSet001/Services	ImagePath
SYSTEM/ControlSet002/Services	ImagePath

5. 評価

本章では，提案方式の有用性の評価を行う．まず，4 章で述べたプロトタイプシステムを用いて，正規ソフトウェアとマルウェアの双方で実験を行った結果を示す．続いて，2 章で述べた既存技術と提案方式の定性評価を行い，両者を比較する．

5.1 実験

5.1.1 実験目的と実験環境

正規ソフトウェアとマルウェアの各々における実験の目的を以下に述べる．

- ＜正規ソフトウェアにおける実験の目的＞
  提案方式の要件として，検出対象となる処理は危険な処理の条件を満たしている必要がある．つまり，任意の正規ソフトウェアにおいてユーザの意図しないタイミングで，危険な処理が行われてはならない．実験により，この条件を満たしているかを調査する．
- ＜マルウェアにおける実験の目的＞
  実際にどの程度のマルウェアが提案方式で検出可能であるか（つまり，承認ダイアログを表示させることができるか）を調査する．

実験環境は仮想マシン上の 32 ビット版の Windows XP SP3 である．ゲスト OS 上に提案方式のプロトタイプシステムを導入し，正規ソフトウェアとマルウェアのそれぞれで実験を行った．

5.1.2 正規ソフトウェアに関する実験

プロトタイプシステムでは前述したように監視対象外のプロセスがあるため，検知漏れが存在する．そのため，正規ソフトウェアに関する実験にプロトタイムシステムを用いるのは適切ではない．そこで，ProcMon というファイルやレジストリ等の処理をリアルタイムで表示するツールを用いた [10]．ProcMon の一部はカーネルで動作し，全てのプロセスの処理を監視できる．ProcMon において，表 4.3 に示した実行ファイルの作成と OS 起動時の自動実行の登録が検出できるようにフィルタをかけ，危険な処理を監視した．ProcMon の監視下において，様々な正規ソフトウェアを実行・操作した．ProcMon が実行ファイルの作成あるいは OS 起動時の自動実行の登録を検出した際，それをユーザの行った直前のイベントと関連付けることができるかどうかを調査した．

表 6 に実行ファイル作成における実験結果，表 7 に OS 起動時の自動実行における実験結果を示す．なお，プロトタイプシステムによる検出結果では，表 6 の Chrome インストーラと Skype インストーラによる実行ファイルの作成と，表 7 の sc コマンドによる自動実行への登録において検知漏れが生じている．

表 6,7 の実験結果から，危険な処理が検出された際，いずれもその検出をユーザの直前のイベントと関連付けることができ，ユーザはそのイベントからその危険な処理を行うことであろうことを推測可能である．例えば，表 6 において，Dropbox のインストーラを実行した際に，実行ファイル作成が検出されている．ユーザは実行ファイルを作成することを想定してインストーラを実行しているため，実行ファイルを作成しようとしていることを通知する承認ダイアログが表示されていても正しく許可を発行できる．このように，今回実験を行った実行ファイルの作成と OS 起動時の自動実行の登録については，ユーザの意図したタイミングのみにその処理が行われている．従って，これらの処理は危険な処理としての要件を満たしており，危険な処理として定義できる．今後，未実装である他の危険な処理についても同様の調査を行っていく予定である．

5.1.3 マルウェアに関する実験

実験に使用したマルウェアは，マルウェア収集サイトである Offensive Computing[11]　と VX Vault[12] から独自に収集したものの内，ウイルス対策ソフトである Symantec

表 6　実行ファイル作成における実験結果

実行ファイル作成が観測された際のユーザのイベント
インストーラ (Dropbox) の実行
インストーラ (Rainlendar) の実行
インストーラ (Lhaplus) の実行
インストーラ (Google Chrome) の実行
インストーラ (Skype) の実行
Lhaplus による実行ファイルを含む ZIP の解凍
Web ブラウザ (IE) による実行ファイルのダウンロード
Web ブラウザ (Chrome) による実行ファイルのダウンロード
Explorer による実行ファイルのコピー (ドラッグ&ドロップ)
copy コマンドによる実行ファイルのコピー
Visual Studio C++ 2010(link.exe) によるビルド

表 7　自動実行への登録における実験結果

自動実行への登録が観測された際のユーザのイベント
インストーラ (Skype) の実行
インストーラ (Rainlendar) の実行
インストーラ (Chrome) の実行
Skype の設定による自動実行への変更
レジストリエディタによる自動実行のエントリへの書き込み
reg コマンドによる自動実行に関するエントリ値への書き込み
sc コマンドによるサービスの登録

Endpoint Protection により検出された実行可能なマルウェア 53 体である．なお，Symantec 社によるマルウェアの検出名が同名であり，実験結果が等しいものは省いた．

表 8 にマルウェアにおける実験結果を示す．実験の結果，53 体中 31 体のマルウェアがバックグランドで実行ファイルを作成，あるいは OS 起動時の自動実行への登録を行い，プロトタイプシステムによって承認ダイアログを表示させることができた．従って，これらのマルウェアに対しては，提案方式が有効であると判断できる．また，31 体の内 2 体のマルウェアは Windows エクスプローラにスレッドを注入し，Windows エクスプローラに危険な処理を行わせるものであった．53 体中 12 体のマルウェアは，承認ダイアログが表示されずに実行が続いたが，これらのマルウェアは不正インストールを行わないマルウェアであるといえ，その点においては比較的に安全なマルウェアである．その他，53 体中 8 体のマルウェアはプロトタイプシステムを導入したことにより，実行時にエラーが発生した．これについては，今後，原因を調査する必要がある．

今回の実験は危険な処理としてインストール（実行ファイルを作成と OS 起動時の自動実行への登録）のみを対象とした．しかしながら，先述のような高い検出率を得ることができた．今後，その他の危険な処理を実装することにより，さらなる検出率の向上が期待できる．

5.2　既存技術との定性評価

5.2.1　既存の承認機構との比較

承認機構の既存技術である UAC との比較を行う．UAC はプログラム起動時の承認機構であるため，昇格プロンプトを表示した時点では，実際に行われる処理やそのタイミングが分からなかった．しかし，提案方式はプログラムの実行中に行われる承認機構であるため，危険な処理を行う直前に，その処理内容を反映した承認ダイアログを表示できる．また，UAC では防ぐことができなかったカレントユーザへのインストールやメールの送信を，提案方式では防ぐことができる．なお，プロトタイプシステムでは表 3.2 に示す不正インストールの検出のみが実装済みでり，その他のメール送信の検出などは未実装である．

表 8　マルウェアにおける実験結果 (全 53 体)

検体数	結果
31	承認ダイアログが表示される
12	承認ダイアログは表示されず，実行を続ける
2	マルウェアによりシステム全体が操作不能に陥る
8	エラーにより実行できず

5.3　ビヘイビア法の既存研究との比較

マルウェアの特徴的な振る舞いを検出する手法と機械学習を用いたマルウェア検出手法と提案方式の 3 手法で定性評価を行った．表 9 に提案方式と従来のビヘイビ法の比較を示す．

- 振る舞いの検出範囲：図 4 に各手法の検出可能な振る舞いの範囲を示す．表中の橙色の楕円が，一つの研究において検出される振る舞いの範囲を示している．マルウェアの特徴的な振る舞いを検出する手法では，その特徴的な振る舞いを持つマルウェアしか検出対象にできないため，検出範囲は狭い．機械学習を用いたマルウェア検出手法では，システムコールの発行履歴に着目するため，検出可能な振る舞いの範囲は広い．特に，一般には知られていないようなマルウェア固有の振る舞いも検出対象にできる．提案方式は，全てではないにしろ，危険な処理として定義した複数の振る舞いを検出対象にできる．
- 誤検知：図 4 に各手法で誤検知が発生する振る舞いの範囲を示す．マルウェアの特徴的な振る舞いを検出する手法では，マルウェア固有の振る舞いを検出するため，基本的に誤検知が少ない．機械学習を用いたマルウェア検出手法では，現状ある程度の誤検知が生じてしまうという課題がある．提案方式は，危険な処理を正規ソフトウェアが勝手に（バックグラウンドで）行うことがないような処理に限定したため，ユーザによる誤検知はないと考えられる．ただし，危険な処理がその条件を満たしていなかった場合に図 4 の (c) 提案方式のように誤検知が生じる．
- 検出時のユーザへの説明：この評価項目は，その手法によりマルウェアが検出された際，なぜマルウェアと

表 9　提案方式と従来のビヘイビア法の比較
(a) マルウェアの特徴的な振る舞いを検出する手法
(b) 機械学習を用いたマルウェア検知手法

評価項目	(a)	(b)	提案方式
振る舞いの検出範囲	×	○	△
誤検知	△〜○	△	○
検出時のユーザへの説明	△〜○	×	○
ユーザビリティ	○	○	△

して検出されたのかをユーザに説明できるかという点に着目した評価である．ユーザに説明することができれば，ユーザに一種の安心感を与えることができる．マルウェアの特徴的な振る舞いを検出する手法では，検出する振る舞いが特徴的であるために，PC に詳しくないユーザではその理由を理解することが難しい場合がある．機械学習を用いたマルウェア検知手法では，機械学習により識別を行うため，ユーザへ検出理由を説明することができない．提案方式は，ユーザがマルウェアの判断を行うため，そもそもユーザへ説明する必要がない．

- ユーザビリティ：従来のビヘイビア法（マルウェアの特徴的な振る舞いを検出する手法と機械学習を用いたマルウェア検知手法）はプログラムが自動的にマルウェアの判断を行うため，ユーザビリティは損なわれない．しかし，提案方式は承認機構であるため，危険な処理を呼ぶ度にユーザへの承認ダイアログが表示され，ユーザビリティが損なわれる．しかし，ホワイトリストを導入することにより，著しくユーザビリティが損なわれることを防止できる．

提案方式は図 4 から分かるように従来のビヘイビア法では検出できなかった，あるいは検出しても誤検知を起こしてしまう振る舞いをユーザの判断を借りることにより誤検知なしに検出できるという利点がある．従って，提案方式は従来のビヘイビア法と組み合わせることによって，より多くのマルウェアを検出することが可能となり，有用な手法であると言える．

6. まとめ及び今後の方針

本論文では，Windows における危険な処理のユーザへの承認機構を提案した．提案方式は，承認対象となる危険な処理を複数設定することにより，従来のビヘイビア法の課題 1 を解決する．また，正規ソフトウェアが勝手に行うことがないような処理を承認対象にすることにより，ユーザによる誤検知をなくし，従来のビヘイビア法の課題 2 を解決する．承認対処となる振る舞いは，従来のビヘイビア法よる検出対象の振る舞いと範囲が異なることから，提案方式によって，従来のビヘイビア法では検出できなかったマルウェアを検出することが期待できる．

提案方式のプロトタイプシステムに危険な処理の一部を

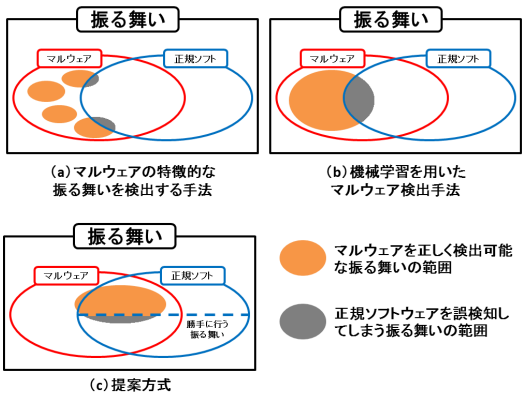

図 4　各手法の検出可能な振る舞いの範囲

実装し実験を行った．その結果，プロトタイプシステムによって多くのマルウェアがバックグランドで行う危険な処理を検出できることを確認し，提案方式の有用性を示した．

今後は，Windows Vista 以降の OS での実験や未実装の危険な処理についても実装を行い，より多くのマルウェアを検出できるようにしていく予定である．また，実装方法の再検討を行い提案システムをより強固なものにしていく．

参考文献

[1] 松本隆明, 鈴木功一, 高見知寛, 馬場達也, 前田秀介, 西垣正勝. 自己ファイル READ の検出による未知ワーム・変異型ワームの検知方式の提案. 情報処理学会論文誌, Vol. 48, No. 9, pp. 3174–3182, September 2007.

[2] 酒井崇裕, 竹森敬祐, 安藤類央, 西垣正勝. 侵入挙動の反復性を用いたボット検知方式. 情報処理学会論文誌, Vol. 51, No. 9, pp. 1591–1599, September 2010.

[3] 松本隆明, 高見知寛, 鈴木功一, 馬場達也, 前田秀介, 水野忠則, 西垣正勝. 動的 API 検査方式によるキーロガー検知方式. 情報処理学会論文誌, Vol. 48, No. 9, pp. 3137–3147, September 2007.

[4] 松本隆宏, 新井悠, 寺田真敏, 土居範久. セキュリティ無効化攻撃を利用したマルウェアの検知と活動抑止手法の提案. 情報処理学会論文誌, Vol. 50, No. 9, pp. 2127–2136, September 2009.

[5] 松本隆宏, 新井悠, 寺田真敏, 土居範久. マルウェアの耐解析機能を逆用した活動抑止手法の提案. 情報処理学会論文誌, Vol. 50, No. 9, pp. 2118–2126, September 2009.

[6] 笠間貴弘, 吉岡克成, 井上大介, 松本隆明勉. 実行毎の挙動の差異に基づくマルウェア検知手法の提案. コンピュータセキュリティシンポジウム 2011 論文集, Vol. 3, pp. 726–731, October 2011.

[7] 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, Vol. 47, pp. 420–429, September 2006.

[8] 伊波靖, 高良富夫. 危険なシステムコールに着目した windows 向け異常検知手法. 情報処理学会論文誌, Vol. 50, No. 9, pp. 2173–2181, September 2009.

[9] Microsoft Research. Detours. http://research.microsoft.com/en-us/projects/detours/.

[10] Process monitor. http://technet.microsoft.com/ja-jp/sysinternals/bb896645.aspx.

[11] Offensive computing. http://www.offensivecomputing.net/.

[12] Vx vault. http://vxvault.siri-urz.net/ViriList.php.