平成26年度　卒 業 論 文

和文題目

自己複製挙動に着目した未知ワーム検知手法の提案

英文題目

Proposal of Unknown-worm Detection Method based on Self-Copy Behavior

情報工学科　渡邊研究室

(学籍番号: 110430052)

神谷　早紀

提出日: 平成 27 年 2 月 12 日

名城大学理工学部

内容趣旨

インターネットが社会で欠かせない存在となる一方で，マルウェアが爆発的に増加し被害が拡大しており，マルウェア対策は重要な課題となっている．特に，マルウェアの一種であるワームは自己増殖機能を持つため被害を拡大させている．新種や亜種のマルウェアが日々生み出されている．一般的にウイルス対策ソフトで用いられる手法ではこの増加に対応が追いつかず，未知のワームを検出できない事が課題となっている．本稿では，未知マルウェアを検出可能な，ビヘイビア法によりワームを検出する手法を提案する．ビヘイビア法とは，事前に定義したワームの挙動を検出する手法である．ワームの特徴である自己複製挙動に着目し，この挙動を検出することで，攻撃を未然に防ぎ，他の PC への拡散を防ぐ．API フックを用いて，プログラムの挙動を検査し自己複製挙動を検出するプログラムを実装し提案方式を実現した．このプログラムにより，既知ワームを検出できるか検知実験を行った．実験結果から，提案方式の有用性を確認した．

目 次

第1章　序論

インターネットは社会や経済を支えるインフラとして機能するようになった一方で，マルウェアが増加し，情報漏洩，迷惑メール，サービス妨害などの被害が絶えず，マルウェア対策は重要な課題となっている．現在，マルウェア作成者の目的は，自己顕示欲のためではなく金銭目的に変化している．マルウェアや，マルウェアによる DDos 攻撃などの「サービス」は，闇市場で販売され，アンダーグラウンドビジネスとして成り立ち，組織犯罪に発展するなど規模・頻度が拡大している．セキュリティベンダの報告 [1] によると，マルウェアのサンプル総数は 2014 年 11 月時点で 3 億を超え，1 分間に 307 件を超えるマルウェアが出現しており，マルウェアは日々刻々と増殖し蔓延している．

マルウェアの中でもボットと呼ばれる悪質なマルウェアが流行している．ボットは，遠隔操作により，DDos 攻撃，情報窃取，マルウェア配布，金融詐欺などの攻撃活動を行い，サイバー犯罪の温床になっている．ボットは，攻撃活動をボットに感染した PC 群で構成するボットネットと呼ばれるネットワーク単位で行う．そのためボットは，マルウェアの一種であるワームが備える自己複製機能と同等の機能を持つ．本研究の目的は，ワームを検出し，ワームの攻撃，拡散を防ぐ事である．

マルウェア検出は一般的にパターンマッチングを用いて行われる．これは，マルウェアを解析して，マルウェアのシグネチャをウイルスパターンファイルに記し，検査対象ファイルのバイナリ列が一致するかどうかを調べ検出する手法である．しかしこの手法では，新種のマルウェアが出現した際，その都度マルウェアを解析する必要があり，ウイルスパターンファイルを更新するまでの間は検出できない．マルウェアの種類が増え続ける原因は，亜種マルウェアが大量発生しているためである．マルウェアのソースコードが出回っており，攻撃者は自由に亜種を作ることができる．また，ZeuS や SpyEye などのボットを作成できるツールキットがアンダーグラウンドで販売されており，専門知識がなくとも GUI 操作により簡単に独自のボットを作成することができる．ツールキットにより，1 つのボットから，数百から数千の亜種ボットを作成できる．さらに，複製を作成する度に自分自身を改変するポリモーフィック型や，メタモーフィック型などの変異型マルウェアが開発されるなど，亜種が次々と生み出されている．このように，パターンマッチングを回避する技術も進化している．未知マルウェアが爆発的に増加しているため，シグネチャ作成が追いつかず，パターンマッチングだけでは対応できない状態である．

未知マルウェアを検出する手法として，マルウェアの挙動を解析する手法がある．これには，静的解析を行う静的ヒューリスティック法と，動的解析を行うビヘイビア法 (動的ヒューリスティック法) がある．静的ヒューリスティック法は，マルウェアが行うであろう処理 (コード) を定義しておき，逆アセンブルにより生成したプログラムコードを解析して検出する手法である．この手法

は，動作の特徴的コード群を検出するため，亜種毎のバイナリの変化に影響を受けず，未知のマルウェアを検出できる．しかし，攻撃者は，実行可能形式のままプログラム本体を圧縮するツールであるパッカーを利用して難読化を行う．特に未知のパッカーを使用した場合は静的ヒューリスティック法では検出することができない．一方，動的解析を行うビヘイビア法においては，難読化に影響を受けず未知マルウェアを検出することができる．ビヘイビア法は，実際にマルウェアを実行して，事前に定義しておいた「マルウェアらしい挙動」を検出する．耐解析機能による暗号化・難読化を行っていたとしても，実行された時に行われるマルウェアの挙動は変化しないため，ビヘイビア法はこれらに影響なく未知マルウェアを検出する事ができる．

本論文では，自己複製挙動を検出することで未知ワームを検出する手法を提案する．ワームの特徴である複製作成動作に着目し，自分自身のプログラムから自分の複製を作成するという特徴的な挙動 (以下，この挙動を自己複製挙動と呼ぶ) を，ワームの挙動として検出する手法である．提案方式の評価を行なうため，自己複製挙動を検出する検査プログラムを作成し提案方式を実装した．この検査プログラムを用いて，既知ワームに対して検知できるかどうか実験を行い，提案方式が有用である事を確認した．

以下，２章では既存研究とその課題について述べる．３章で提案方式について述べ，４章で実装方法，５章で既存研究との定性評価と，作成した検査プログラムを用いた検知実験による評価と考察について述べる．最後に６章でまとめを行う．

第2章　既存研究

本章では，ビヘイビア法によるワーム検出に関する既存研究の概要とその課題について述べる．以下2つの研究はどちらもワーム特有の自己複製という挙動に着目した検知方式である．

2.1　「自己ファイル READ」挙動検出による方法

自己ファイル READ の検出によるワーム検知手法 [2, 3] は，ワームが感染のために複製する際に，自分自身のファイルを全て読み込む（自己ファイル READ）という特有の挙動を，ワームの挙動として定義し検出する手法である．自己ファイル READ によって，プログラムが自身のプログラムの一部を複製する挙動を図 1 に示す．通常ユーザが行う操作でファイルコピーが考えられるが，ファイルコピーは OS 標準のプログラムが複製を作成する．この手法では単に複製生成を検出するのではなく，実行中のプロセスが自己のプログラムが格納されているファイルを全て読み込む挙動を検出する事で，正常の複製操作とワームの挙動を区別する事ができる．

しかし，パケットキャプチャツールである WireShark のアンインストーラでは, 自己ファイル READ が観測される．そのため正規プロブラムをワームと誤検知してしまう事がある．

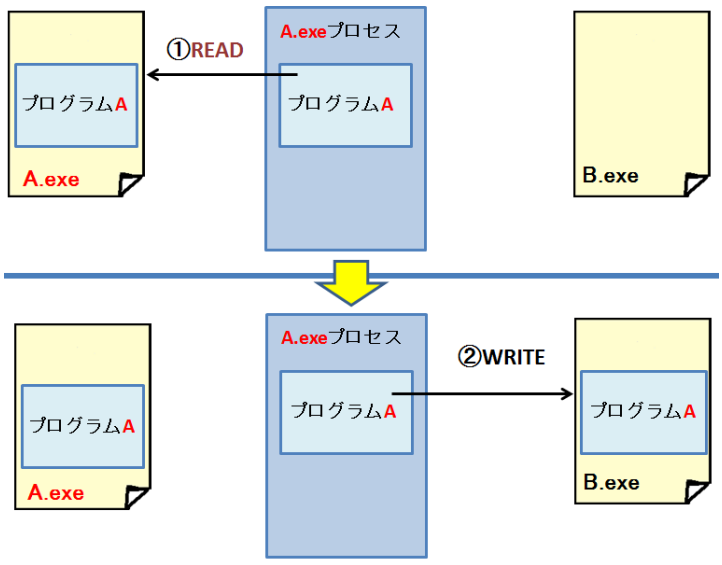

図 1　自己ファイル READ による自プログラムの複製

2.2 「侵入挙動の反復性」挙動検出による方法

侵入挙動の反復性を用いた検知方式 [4] は，実行環境の復元により，侵入挙動が繰り返されるという挙動を検出する．という挙動を検出する．ここでの侵入挙動とは，ファイル作成と，自動実行リスト（スタートアップフォルダ，レジストリ，サービスプロセス）への登録を指す．ワームは自己増殖が特徴であるため，意図的に実行環境を復元し，再実行する事により，再度侵入挙動が観測される．この反復性を利用して検出する手法である．概要を図 2 に示す．この手法で言う実行環境とは，挙動を変化させる実行ファイルの要因である．実行場所とファイル名と定義している．「A.exe」という名の実行ファイル α が,「A'.exe」という名の実行ファイル β を作成した時，β の名前を「A.exe」と変更した実行ファイル γ を元の α が存在するフォルダへ移動し，再度 γ を実行する．γ を実行した時，再び侵入挙動が行われた場合，α および β を検出するという手法である．

正規プログラムのインストーラにおいても，ここで定義された侵入挙動が起こるが，インストーラが作成したアプリケーションは，侵入挙動を行なうことはない．そのため，侵入挙動の反復は起こらず，誤検知が起こることはない．

しかし，この手法は，環境を復元する，再度実行ファイルを実行するなどの処理をする必要があるため，検出までに時間がかかり，検出に必要な処理が重い．

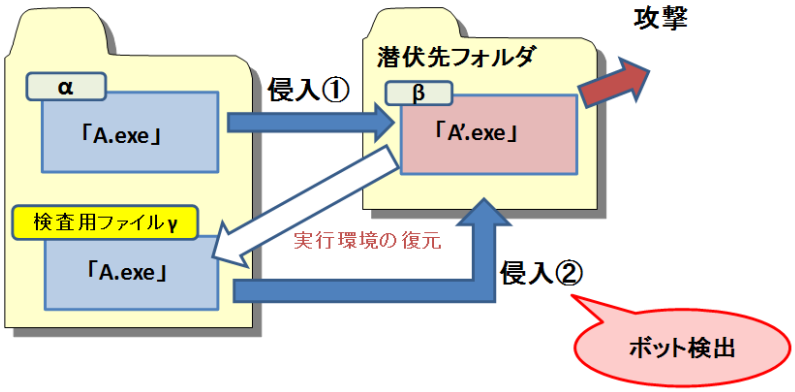

図 2　侵入挙動の反復性

第3章　提案方式

マルウェアが増えていく原因であるワームの自己複製挙動に着目したワーム検知手法を提案する．本章では，まず，ワームの挙動や正規プログラムの動作を考察し，次に提案方式のワーム検出手法を述べる．

3.1　ワームの挙動

3.1.1　自己複製挙動

ワームとは，独立したプログラムであり，自己複製をして，他の PC に拡散を行うマルウェアである．マルウェア開発者はユーザを騙すことによって，悪質な Web サイト上の，あるいはメールに添付されたワームをダウンロード・実行させる．ユーザにより実行されたワームはまず初めに侵入活動を行い，その後，攻撃活動を行う．

ワームは実行されると，最初に PC 内に常駐するための活動である侵入活動が観測される．多くのマルウェアに見られるようにワームも，基本的にユーザに気づかれないように動作する．まず，ユーザが，普段ほとんど開くことがなく，フォルダ内のファイルを把握していないために，変更しても気付かれにくいシステムフォルダ内などに複製を作成する．作成時，新規作成，または既存のファイルに上書きすることで，自身の複製を作成する．その次に，プログラムが再起動後も OS 起動時に自動的に実行されるように，自動実行に関するレジストリに，作成した複製ファイル (コピー) を登録する．直接自分自身 (オリジナル) をレジストリに登録しない理由は，ユーザに見える場所にある状態のオリジナルは，ユーザによって削除される可能性も高く，潜伏に適さないためである．そしてその後，多くのワームは潜伏のためオリジナルを削除する．このように，マルウェアは始めに，ユーザに気付かれないように PC 内に駐在するための侵入活動を行う．

マルウェアは侵入活動を終えた後に，目的である攻撃活動を行う．ワームはその特徴である自己増殖をして拡散活動を行う．USB などのリムーバブルディスクや，共有フォルダ，に複製を作成したり，メールに自身を添付して送信することで，他の PC への侵入を試み拡散する．さらに，クレジットカード情報やパスワードなどを盗み出す情報窃取，サーバなどの他の PC に大量のパケットを送信して負担をかけサービスを妨害する Dos 攻撃，大量のスパムメールを送信するなどの攻撃活動を行なう．

以上のように，ワームは，侵入活動，攻撃活動，共に自己複製を行う．特に，侵入時にまず自己複製を行うため，自己複製挙動を検出することで，攻撃活動が行われる前にワームを検出できる．本論文では，自己複製挙動をワーム必須の挙動として定義する．

3.1.2　変異型ワームの挙動

ポリモーフィック，またはメタモーフィックと呼ばれる，複製の度に自己改変してウイルス対策ソフトの検知を回避する変異型のワームが存在する．

ポリモーフィック型のワームは，複製を作成する度に，自身のコードを暗号化する．概要を図 3 に示す．ポリモーフィック型のワームは実行されると，まず，ワーム本体の復号化し，そのオリジナルのコードを実行する．複製の際には，自身のプログラムを再度，ランダムに生成した鍵を用いて，暗号化する．

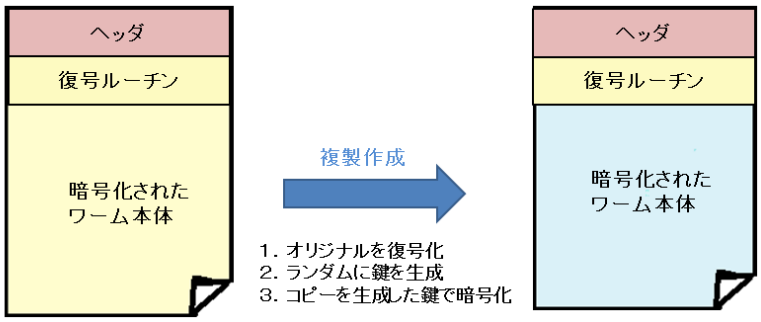

図 3　ポリモーフィック型ワーム

メタモーフィック型のワームはは，複製の度にコードを変化させたり，意味のないコードを加えることで自身を書き換える．概要を図 4 に示す．複製を作成する度に，マルウェアをいくつかのコードブロックに分けてブロックの順番を入れ替えたり，ジャンクコードをそれらのブロックに加えて自身を書き換え，難読化するワームである．

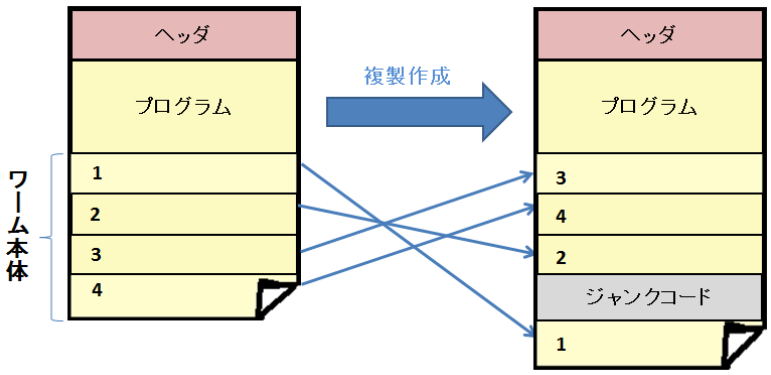

図 4　メタモーフィック型ワーム

このように，複製を作成する度に変化する変異型のワームが存在するが，バイナリが変化していたとしても検出できる手法を考える．

3.2 検出方法

3.2.1 検出方法の概要

3.2.1 項で述べたように，マルウェアは侵入活動で複製を作成する事が多く，ワームは拡散するという性質上，必ず複製を作成する．この複製作成時に観測される，自己複製挙動に着目し，提案方式では検出すべきマルウェアの挙動を「自己複製挙動」と定義する．自己複製挙動とは，自身のプログラムの複製を作成する事であり，自プロセスの実行ファイルを読み込み，他のファイルに書き込むという動作として現れる．複製であるので，読み込みが行われたオリジナルのファイルと，書込みが行われたファイルは，ファイルの内容が一致する．

複製を作成するためには，ファイルを新規作成する場合でも，既存のファイルに上書きする場合であっても,「書き込み」操作が必要である．そこで，書き込み操作を監視する．書き込み操作を検査することで，ワームの挙動を監視する．そして，書き込み処理が終了した時に，ファイルの比較を行ない，一致するかどうかにより複製かどうかを検査する．以上の検査を行なう事で自己複製挙動を検出する．提案方式の概要を図 5 に，検出のフローチャートを図 6 に示す．なお，フローチャートの (3) の分岐については，3.2.3 項で後述する．

（1）ファイルへの書き込みの検査

全プロセスの処理を監視し，書き込み操作の有無を検査する．

（2）ファイル比較

書き込みが終了した時，書き込みを行ったプロセスの実行ファイルと，書き込みが行われたファイルを比較する．

比較に際して，3.1.2 項で述べた変異型のワームは，オリジナルとコピーでは，プログラム領域のバイナリは変化する．そのため，単純にファイル同士の比較を行うと，同じ挙動を行うファイルであっても，コピーはオリジナルに比べファイルの一部が変化しているので一致しない．そこで，変異型であっても，変化しないヘッダを比較する事で複製の判定をする．さらにヘッダ内のどの項目を比較するかについては次節で詳細を述べる．

なお，一般的に Windows における実行ファイルの拡張子は「.exe」である．しかし，実際にはファイルの中身が実行ファイルの形式にさえ従っていれば，実行可能である．このため，ワームが作成した複製の拡張子が「.exe」でない可能性があるが，提案方式は実行ファイルの内容を比較するため，拡張子に関係なく検出することが可能である．

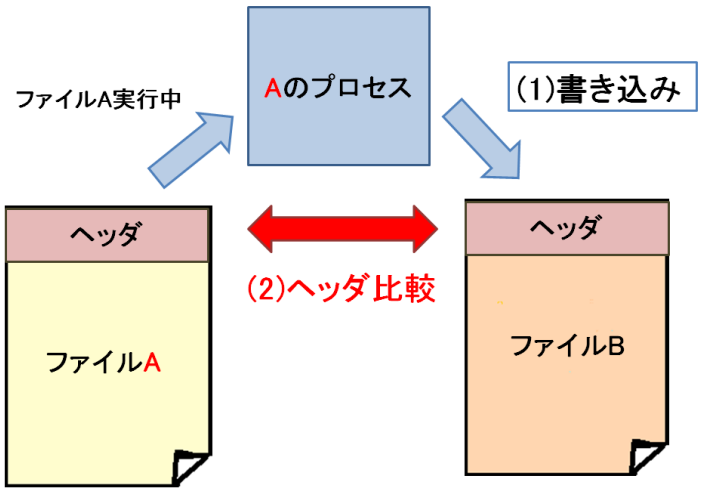

図 5　提案方式の概要

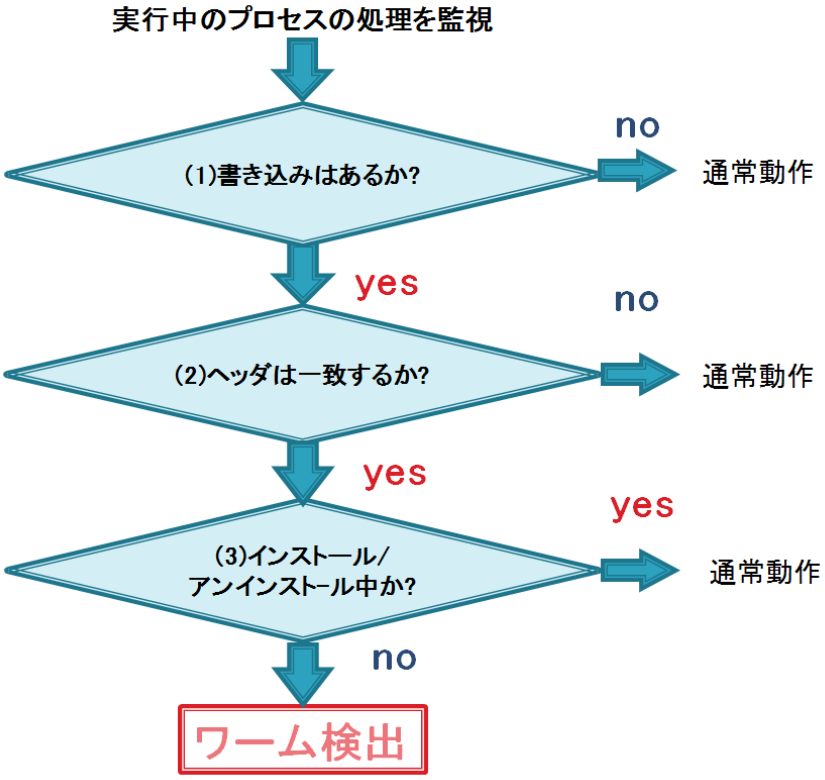

図 6　提案方式のフローチャート

3.2.2 比較項目について

独立したプログラムであるワームは，実行ファイルであり，Windows の実行ファイルは PE(Portable Executable) 形式に従っている．実行ファイルには必ずヘッダがあり，OS はヘッダ内の情報をもとに，実行ファイルをメモリ上にロードする．ヘッダは様々な情報で構成されており，現在では使用されていない項目から，なくては実行ができなくなる重要な項目まで多くの項目がある．このヘッダの中でも，変更が難しい項目を比較する事で複製かどうかを判定する．以上のことから．比較する項目を定めるにあたって，以下の 2 点の条件を満たす事が必須である．

- 実行ファイルごとに固有な値である
- 実行に必須であり，ランダムな値にするなどの変更が難しい

まず，固有な項目を定めるにあたって，論文 [5] の実行ファイルの固有情報からシグネチャを生成するために行われた調査を参考にする．互いに異なる Windows の実行ファイル 1000 個のうち任意の 2 つを取り出し，ヘッダの項目毎の一致率を調査したものである．表 1 に一致率の低い項目上位 10 個を示す．

Import Table と Address Of Entry Point フィールドは，それぞれ一致率が 1 万分の 1 の確率で十分に低く，固有値に近い項目だと考えられる．Import Table フィールドは，プログラムが使用する DLL（API）の情報を格納したインポートテーブルの場所を示す値である．Address Of Entry Point フィールドは，コードセクションのどの位置からプログラムを開始するかを示す値である．この値が正確でなければプログラムを実行できず重要なフィールドである．

この 2 つの項目により，比較するべき項目の条件を両方満たしている．以上のことから，Import Table と Address Of Entry Point フィールドをファイル比較で比較すべき項目とする．

表 1　ヘッダ項目の一致率

項目名	一致率 (%)
Import Table	0.02
Address Of Entry Point	0.09
Time Date Stamp	0.14
Size Of Code	0.63
Import Address Table	0.92
Resource Table	1.09
Size Of Initialized Data	1.14
Size Of Image	1.67
Base Of Data	3.32
Check Sum	19.6

3.2.3 誤検知の対策

正規プログラムにおいて，インストール/アンインストールの挙動がワームの特徴と類似する．その為，正規プログラムの中でインストーラ/アンインストーラにおいて誤検知が起こる可能性がある．

そこで，インストーラ/アンインストーラに対して自己複製挙動が行われるかどうかを，プロセスの処理を監視できる Procmon と，ファイルを比較できる CompFile というツールを用いて検証実験を行った．表 2 に実験結果を示す．

インストーラにおいては，3/9 体について自己複製挙動が観測された．WireShark と Lhaplus は，インストール時に付随してダウンロードされるプログラムが，アンインストーラをダウンロードし，このプログラムとアンインストーラが一致する挙動が観測された．GoogleChrome は，インストール時に GoogleUpdate が自分と同じプログラムである GoogleUpdate を作成し，即削除する挙動が見られた．アンインストーラにおいては，4/9 体について自己複製が観測された．これら 4 体は自身の複製を一時フォルダに作成していた．アンインストールに自分自身を含めてアプリケーションを削除する必要があるため，一時フォルダに一旦自身の複製を作成し，自身を削除するようである．この実験から，正規プログラムであるインストーラ/アンインストーラ共に，しばしば自己複製が行われる事が判明した．

そこで，自己複製挙動を検出した時に，ポップアップを出し，ユーザにインストールもしくはアンインストール中であるかどうかを訪ねて確認することで，正規プログラムとを区別する．

表 2　正規プログラムにおける自己複製挙動有無

検体	Installer	Uninstaller
WireShark	あり	あり
DropBox	なし	あり
AdobeReader	なし	あり
GoogleChrome	あり	なし
iTunes	なし	なし
Lhaplus	あり	なし
LINE	なし	あり
Skype	なし	なし
サクラエディタ	なし	なし

第4章　実装

提案方式の有用性を評価するため，自己複製挙動を検出する，自己複製挙動検査プログラムを作成した．本章ではその実装方法について述べる．

4.1　実装の概要

API フックを用いて提案方式を実装した．API とは，Application Programming Interfaces の略で，アプリケーションが OS の機能を利用するために用意されたインターフェイスである．API フックとは，アプリケーションの API 呼び出しをフック (横取り) して，別の独自コードへ処理を飛ばす技術である．この技術を用いて自己複製挙動を検出する．検査プログラムの動作概要を図 7 に示す．

API フックには複数の手法があるが，実装が比較的容易で，フック回避が難しい Detours Library [6] を使用する方法を採用した．Detours Library とは，Microsoft Research Team からリリースされている API フック用のライブラリである．Detours Library によるフック (以下，Detours フックと呼ぶ) は，仮想メモリ上にロードされたプロセス内の API 関数の先頭命令を JMP 命令に書き換えて，独自の処理へ飛ばし，API 呼び出しを横取りする (図 8)．

Detours フックはプロセス単位のフックであるため，実際にマルウェアに対して Detours フックを仕掛けるためには，実行中のプロセスに対してこの処理を挿入 (Injection) する必要がある．提案方式では，DLL Injection を行なう．DLL Injection とは，DLL をプロセスに強制的にロードさせる技術である．DLL Injection の方法は複数の手法があるが，DLL がロードされるタイミングをコントロールでき，実装が容易な DetourCreateProcessWithDll 関数を利用する．この関数は，Detours Library が提供する，引数で指定した任意の DLL を強制的にロードした状態でプロセスを起動するための関数である．

DLL は，DLL のエントリポイントとなる DllMain 関数という関数を持つことができる．システムは，プロセスに DLL が初めてマッピングされた時と解除された時と，スレッドが作成された時と終了した時に，この関数を呼び出す．これらの呼ばれた理由 (タイミング) を DllMain 関数はパラメータで知ることができる．DLL は，呼ばれたタイミングに従って，タイミング毎に処理（初期化/終了処理）を行なう．DLL が初めてプロセスのメモリ空間にロードされる時に DllMain 関数が初めてにシステムに呼び出される．このタイミングで，プロセスに関する初期化処理として，検査対象 API に Detours フックを仕掛ける事で，これ以後，検査対象 API が呼び出される度に，呼び出しを横取りし，自己複製検査処理を行うようになる．

上記の API フックと，Dll Injection の 2 つの技術を用いて検査プログラムを作成した（図 7）．検査プログラムは，API フックにより自己複製を検出する「InspectSelfCopy.dll」とその DLL を検

査対象のプログラムへインジェクションするプログラム「DllInjection.exe」の 2 つから構成される．「DllInjection.exe」から検査対象のプログラムを指定して起動することで，提案方式によって自己複製挙動を検査することができる．

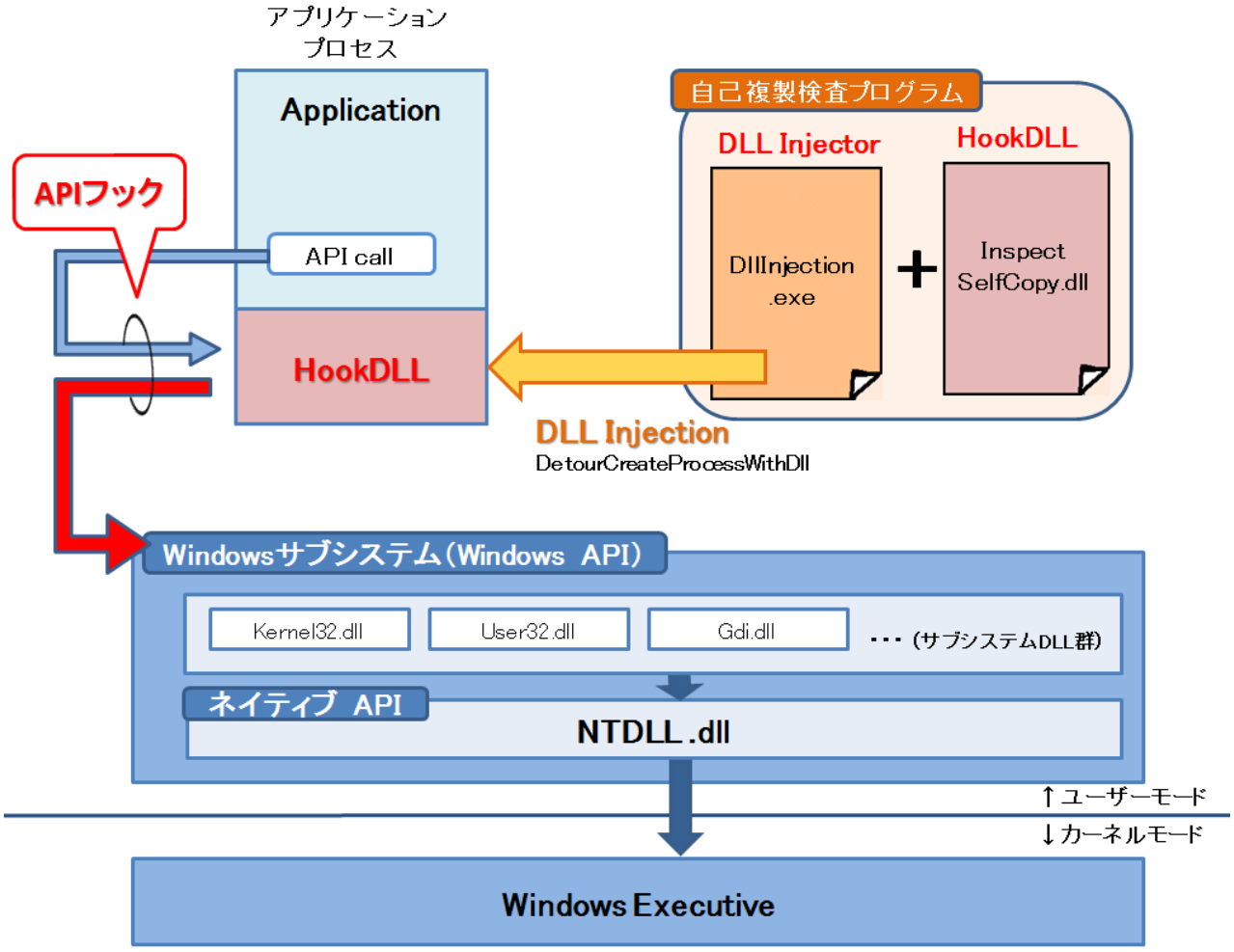

図 7　検査プログラムの動作概要

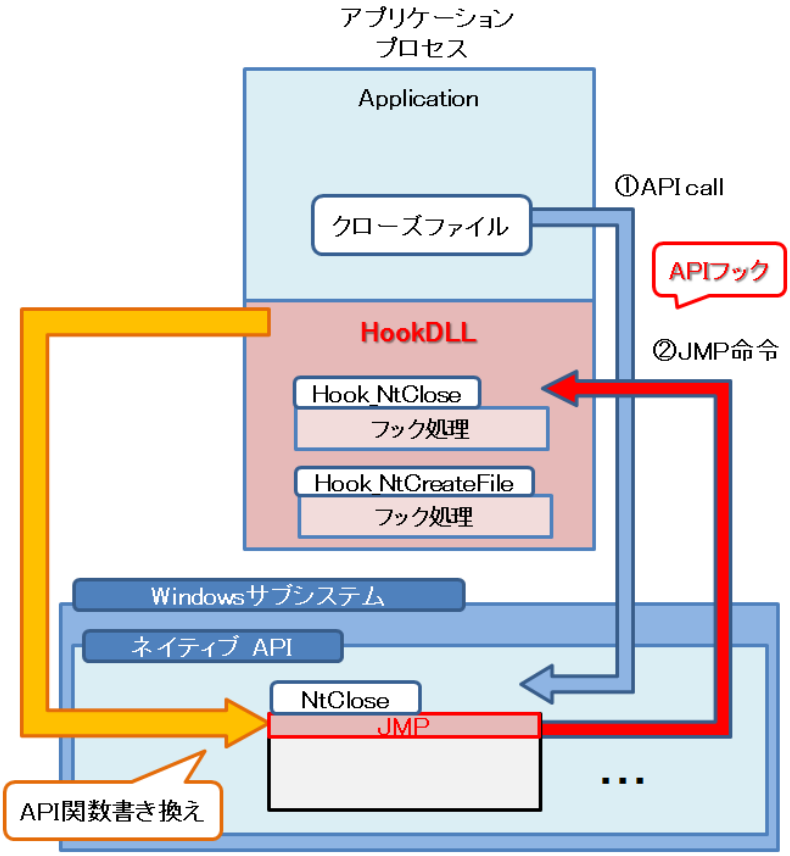

図 8　Detours フックの仕組（NtClose API フックの例）

4.2　フック処理について

自己複製挙動を検出するためには，書き込み操作を検知し，その後にヘッダを比較する必要がある．まず，書き込み操作を検知するために API フックにより挙動を検査する．書き込みがあった場合に，ヘッダ比較処理を行なう．

フックを仕掛けるべき API について考察する．書き込み操作を行なうネイティブ API として，NtWriteFile が用意されている．しかし，NtWiteFile が呼び出された時，書き込み先のファイルは操作中であるため，このファイルへの共有アクセス権がなければこのファイルを読み込む事ができず，比較を行なう事ができない．また，操作対象のファイルのパスをパラメータから直接は得る事ができないため処理が増える．

そこで提案方式では，NtCreateFile　API と NtClose　API にフックを仕掛けて，書き込み操作の有無を監視する．NtCreateFile とは，ファイルを作成したり開いたりする際に呼び出されるネイティブ API 関数である．NtClose とは，ファイルハンドルがクローズされた時，つまりファイルがクローズされた時に呼び出されるネイティブ API 関数である．書き込みを行なうには必ず，ファイルを開く必要があり，その時 NtCreateFile が呼び出される．NtCreateFile をフックし，操作中の

ファイルに対するアクセス権（読み込み，書き込みなど) を調査することで，ファイルへの書き込みを検知する．ただし，このタイミングではまだ書き込みは行われていないため，書き込み処理が終了した時に比較を行なう必要がある．そこで，ファイルをクローズし，NtClose が呼び出された時にヘッダ比較を行なう．以下に，NtCreateFile フック時 (呼び出し時)，NtClose フック時の処理の概要を述べる．

- **NtCreateFile フック後の処理 - Hook_NtCreateFile 関数 -**
  NtCreateFile 呼び出しを横取りした後，飛ばされる先の関数を「Hook_NtCreateFile」と命名した (図 8).

  （1）書き込み権限の有無を検査

  NtCreateFile のパラメータには，DesiredAccess という引数があり，ファイルへのアクセス権限が示されている．この DesiredAccess 引数で，書き込み権限を持つ GENERIC_WRITE または GENERIC_ALL のフラグがあるかどうかを検査する．書き込み権限を持つ場合は続いて下記（2）処理を行なう．

  （2）ファイルパスとハンドルを保存

  書き込み権限を持っていた時，ファイルを管理する値であるファイルハンドルをキーとして，アクセスしようとしているファイルのパスを，ハッシュテーブルに登録する．ハンドル値はユニークな値である．

- **NtClose フック後の処理 - Hook_NtClose 関数 -**
  NtClose 呼び出しを横取りした後，飛ばされる先の関数を「Hook_NtClose」と命名した (図 8).

  （1）書き込み処理終了を検査

  この NtClose のファイルハンドルと一致するハンドルがハッシュテーブルに登録されているかどうかを検索する．一致するものがあった場合，書き込みアクセス権を持って開かれたファイルが閉じられようとしており，書き込みが終了したという事であるので，続いて下記（2）ヘッダ比較処理を行なう．

  （2）ヘッダ比較

  ハッシュテーブルに登録されたパスにあるファイル（書き込み先ファイル）と，NtClose を行った（書き込みを行った）プロセスの実行ファイル，つまり自分自身のプロセスの実行ファイルのヘッダを読み込み，ImportTable と AddressOfEntryPoint フィールドの比較を行なう．これらが共に一致した時，自己複製挙動であるとして検出する．

この検査プログラムはユーザーモードで動作する．そのため，プログラムが自発的にファイルをクローズしない場合，ユーザーモードで API フックする Detours フックでは，OS が行うクローズを検知できない．この問題に対する対策は，ファイルシステムのフィルタドライバを改造し，そこでファイルクローズを検出することで解決する．

第5章　評価

本章では，提案方式の有用性の評価を述べる．5.1 節では，既知ワームに対して提案方式による検知実験を行った．実験の方法と結果と考察について述べる．5.2 節では，既存研究との定性評価を述べる．

5.1　検知実験

5.1.1　実験方法

提案方式の有用性を検証するため，ワームを用いた，レポートによる調査，ツールによる検知実験，検査プログラムによる検知実験を行った．その方法や，環境について以下に述べる．

検体は，マルウェア配布サイト [7,8] から独自に入手したマルウェアの内，検出種別がワームであった 25 体を使用した．実験環境は，仮想マシン VMware 上の Windows7 SP3 で，ネットワーク環境はホストオンリーとした．管理者権限により行った．

- **Symantec 社のレポートによる調査**
  Symantec 社のレポート [9] から，検体に自己複製挙動があるかどうかを調査した．自分自身 (実行ファイル) のコピーを作成すると記述がある検体は，自己複製挙動が認められる．
- **ツールによる検知実験**
  ツールを使って，仮想環境上でマルウェアを実際に実行し，挙動を監視する事で，自己複製挙動があるかどうかを検査する実験を行った．挙動の監視は，Procmon [10] を用いて行った．このツールは，プロセスが行った処理をリアルタイムに表示させることができる．書き込み操作の検査は，Procmon により実行ファイルの生成を検査することで行った．ヘッダ領域の比較は，２つのファイルを比較できる CompFile というツールを用いて行った．ワーム実行中に生成された実行ファイルと，そのファイルを生成したプロセスの実行ファイルを比較し，先頭から何バイトまで一致するかを検査する事で判定した．一致した場合，自己複製挙動である．
- **検査プログラムによる検知実験**
  作成した自己複製挙動を検出する検査プログラムを用いて，マルウェアを検出できるかどうか実験を行った．実験環境はツールによる検知実験と同じで，仮想環境上でマルウェアを実行し，検査プログラムを使用して検出できるかどうかを実験した．

5.1.2 結果

上記の 3 つの検証方法による調査結果，実験を行った．その結果を表 3 に示す．

表 3　検知実験の結果

マルウェア名	発見日	Symantec 社レポート	ツールによる検知実験	検査プログラムによる検知実験
W32.Cridex	2012/1/20	○	○	○
W32.Yimfoca	2010/5/2	○	○	○
W32.Koobface	2008/8/3	○	○	○
W32.Koobface.B	2008/8/3	○	○	○
W32.Badday.A	2007/10/3	○	○	○
W32.SillyDC	2006/10/4	○	○	○
W32.Mytob.BE@mm	2005/4/21	○	○	○
W32.Mydoom.F@mm	2004/2/20	○	○	○
W32.Klez.gen@mm	2004/2/18	○	○	○
W32.Mimail.Q@mm	2004/1/7	○	○	○
W32.IRCBot.NG	2011/4/7	○	△	×複製有
W32.Pilleuz!gen2	2010/2/25	○	△	×複製有
W32.Pilleuz	2009/9/29	○	△	×複製有
W32.Fubalca.E	2007/4/1	○	△	×複製有
W32.Changeup!gen20	2012/11/27	○	×	動作を停止
W32.Changeup!gen23	2012/8/22	○	×	動作を停止
W32.Disttrack	2012/8/16	○	×	×
W32.Buzus	2009/12/10	○	×	×
W32.Ircbrute	2008/6/20	○	×	×
W32.SillyFDC	2007/2/27	○	×	×
W32.Evaman.C@mm	2004/8/3	○	?	○
W32.Feebs.J@mm	2006/1/16	×	○	×
W32.Zimuse	2010/1/23	×	×	×
W32.Waledac	2008/12/23	×	×	×
W32.Downadup	2008/11/21	×	×	×

レポートによる調査では，自分自身 (実行ファイル) のコピーを作成すると記述があった検体は○を，なかった検体を×とした．調査結果から，検体の 8 割以上となる 25 体中の 21 体が自己複製を行うことが判明した．したがって，自己複製をワーム特有の挙動と定義した提案方式の正しさが示された．

ツールによる検知実験では，自己複製挙動が観測された検体を○，ないものを×とした．△には自身ではなく他のプロセスがワームの実行ファイルを複製した検体であり，詳細は次項で述べる．「?」 は，実行すると即座に PC が強制ログオフされ，検知実験を行えなかった検体である．15 体のワームで複製挙動があり，その内 11 体のワームは自分のプロセスから直接複製を作成していた．半数以上ののワームについて実際に自己複製挙動が観測され検出できることが確認できた．

検査プログラムによる検知実験では，ポップアップが出た検体は○を，出なかった検体は×とした．実験の結果，検査プログラムにより 11 体のワームを検出できることが確認できた．しかし，検査プログラムでは，ツールの実験で確認できていた他のプロセスをによる複製を検出できていない．これらの検体は次項で考察する．また，2 体の検体でエラーにより動作が停止してしまった．

以上の結果から，提案方式が有用であり，実装については一部の特殊なワームを除いて，提案通りの検出方法が実現できたと言える．以下，検出できなかった検体についての考察を述べる．

5.1.3 考察

提案方式の適用範囲

Symantec 社のレポートによると，4 体の検体については，侵入時，実行ファイル以外の拡張子で届き，複製するファイルも実行ファイルではないタイプ，または，侵入活動で複製を作成せず，且つ，メールでのみ拡散活動を行うタイプのマルウェアであった．提案方式では，複製を作成しないメールでの拡散活動を検出できないため，侵入活動を行わず，メールでのみ拡散を行なうこのタイプのワームは検出できない．実行ファイル形式でないマルウェアは，拡散の点でワームの特徴をもつが，独立して動作するワームではないため提案方式の適用範囲外のマルウェアである．提案方式では独立して動作する実行ファイル形式のワームを検知対象とする．

環境による挙動の変化

レポートでは自己複製挙動があるとされているのに，実際に検知実験を行なうと，自己複製を行わない検体 6 体については，実行環境による要因が考えられる．まず，この実験はインターネットに接続していない環境であるため，本来の動作が行われず挙動が変わった事が考えれられる．

また，ワームが想定している OS のバージョンが異なることで，動作しなかったことが考えれる．WindowsXP 上で，同様のツールによる検知実験を行ったが，多数のワームで Windows7 上とは異なる挙動が観測された．Windows7 上では自身のプロセスから複製を作成しなかった 2 体の検体が，WindowsXP 上では自己複製を行った．

さらに，この実験は仮想環境で行っているため，ワームが耐解析機能 [11,12] を備えており，仮想環境で実行していることを検知し，動作を中止するなど挙動を変えた可能性が考えられる．耐

解析機能をもつマルウェアは，2002 年には発見されており，使用した検体全てについてこの機能を持つ可能性がある．

このように，環境による要因によりワームの挙動が変化し自己複製を行わなかったと推測される．

他プロセスに複製処理を委託

ツールによる検知実験で，△とした 4 体の検体において，ワームのプロセスではない他のプロセスが，ワームの複製を作成していた．これら 4 体のワーム実行中に，Explorer.exe や，Internet-Explorer.exe のプロセスが実行ファイルを作成していた．この生成された実行ファイルと，実行中のワームのファイルとのファイル比較を行うと，一致し，この実行ファイルはワームの複製であることがわかった．つまり，ワームが複製作成処理を他プロセスに委託している事がわかった．

検査プログラムによる検知実験では，これら 4 体のワームは上記のように，複製を作成しているにもかかわらず，検出できなかった．実装した検査プログラムは，書込みを行ったプロセスの実行ファイルと，書き込まれたファイルを比較する事で自己複製を検出する．他プロセスに複製処理を委託した場合，書込みを行ったのはワームではない他プロセスのため，比較対象のファイルは，ワームのファイルではなく，処理を委託されたプロセスの実行ファイルになる．このように比較すべき対象のファイルがすり替わってしまうため，現在の検査プログラムでは他プロセスを介した自己複製を検出できない．

他プロセスに処理を行う方法は，子プロセスを作成する，他プロセスに対してコードを注入する Code Injection を行うなど複数考えられる．このようなワームを検出するため，プロセスの処理を追いかけ，比較すべきファイルを正確に把握する必要がある．

5.2 定性評価

2 章で述べた既存研究と提案方式との定性評価を行なう．表 4 に定性評価を示す．

表 4　既存研究との定性評価

評価項目	自己 READ	侵入挙動の反復性	提案方式
変異型ワーム	○	○	○
侵入挙動の検出	○	○	○
拡散挙動の検出	○	×	△
誤検知	×	○	△
処理の重さ	○	×	○

（1）変異型ワーム

自己ファイル READ を検出する手法は，変異型であっても一旦自分自身を読み込む必要があるため検出できる．侵入挙動の反復性を検出手法は，バイナリに影響を受けず「反復」という挙動を検出する手法のため検出できる．提案方式では，変異型ワームでも変化する事が困難なフィールドを比較するため，このタイプのワームも検出できる．

（2）侵入挙動

侵入挙動の反復性を検出する手法は，侵入挙動を検出する事こそがワームビヘイビアの定義であり検出できる．ワームの侵入挙動には，自己 READ や自己複製挙動がみられるため，提案方式や自己ファイル READ を検出する手法においても侵入挙動を検出する事ができる．

（3）拡散挙動

自己ファイル READ を検出する手法は，複製の度に自身が読み込まれるため，拡散挙動を検出することができる．侵入挙動の反復性を検出する手法は，自動実行への登録の監視を含めた侵入挙動のみに焦点をおいた手法であるため，拡散挙動を検出できない．提案方式は，リムーバルディスクへの複製など，実際にファイルが作成される場合は検出できるが，リンクを貼るだけのメールへ添付して拡散する挙動は検出できない．

（4）誤検知

自己ファイル READ を検出する手法では正規プログラムでもこの挙動が観測されるため誤検知が生じる．侵入挙動の反復性を検出する手法は，正規プログラムでは，侵入挙動と類似した挙動がインストーラにあるが，インストールでは作成されたアプリケーションの実行ファイルが，侵入挙動を行なうことはないため，反復はされず誤検知はない．提案方式は，正規プログラムであるインストーラ/アンインストーラで自己複製が見られるが，ユーザにインストール/アンインストール中であるかどうかを問い合わせる事で誤検知を回避する．しかし，有用な正規のプログラムに見せかけた不正プログラムを，ユーザがユーザの意思でインストールする場合は，誤検知が発生する可能性がある．

（5）処理の重さ

自己ファイル READ を検出する手法では，自身のファイルが読み込まれた時，どこまで読み込まれたかを監視する必要がある．特に，複数回にわたってファイル全体を READ することがあるため，それをチェックする必要性がある．しかし，自己ファイル READ 自体が通常のプログラムではあまり見られないので，処理は重くないと考えられる．侵入挙動の反復性を検出する手法は，環境を復元する事と，再度実行して挙動を監視する必要があり，検出までに時間がかかり重い処理となる．提案方式では，実行ファイルに書き込みがあった時に，ヘッダの 2 つのフィールドを比較するだけであり，処理は重くないと考えられる．

第6章　まとめ

本論文では，ワームの特徴そのものである自己複製挙動に着目し，自己複製挙動を検出することで未知ワームを検出する手法を提案した．API フックにより提案方式を実装し，自己複製挙動を検出する検査プログラムを作成した．提案方式が有効であるかを評価するため，仮想環境上でワームを実行し，検査プログラムで検出できるかどうか検知実験を行った．また，ツールを用いた検知実験も行い，自己複製挙動の有無を検証した．ツールによる検知実験の結果，既知マルウェア 25 体の検体の内，15 体が複製を作成していた．検査プログラムによる検知実験の結果，この実際に複製挙動を行ったワーム 15 体の内 11 体を検出する事ができ，提案方式が有用である事を確認した．複製を作成したワームの内 4 体は，自身のプロセスからではなく，他のプロセスを介して複製を作成していた．今後は，このように他プロセスに処理を委託されても，処理を追いかけて複製を検出する手法を検討し，提案方式を改善していく．

謝辞

本研究を進めるにあたり，多大なる御指導とご教授を賜りました，指導教官である　渡邊晃教授に心から感謝致します.

本研究を進めるにあたり，ご意見並びにご助言を賜りました，旭健作助教，鈴木秀和助教に心より感謝致します.

本研究を進めるにあたり，常日頃からご教示くださり，数々のご助言を賜りました，早川顕太氏に心より感謝致します.

最後に，本研究を進めるにあたり，多くの討論の場において有益なご意見を賜りました，渡邊研究室及び鈴木研究室の先輩方，そして同期の皆様に心より感謝致します.

参考文献

[1] : McAfee 脅威レポート 2014 年第 3 半期 (2014). http://www.mcafee.com/jp/threat-center/report/download88.aspx.

[2] 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 (2007).

[3] 酒井崇裕，長谷　巧，竹森敬祐，西垣正勝：自己ファイル READ/DELETE の検出によるボット検知の可能性に関する一検討，コンピュータセキュリティシンポジウム 2008 論文集 (2008).

[4] 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 (2010).

[5] 中谷直司，小池竜一，厚井裕司，吉田等明：メール型未知ウイルス感染防御ネットワークシステムの提案，情報処理学会論文誌，Vol. 45, No. 8, pp. 1908–1020 (2004).

[6] : Detours. http://research.microsoft.com/en-us/projects/detours/.

[7] : Offensive Computing. http://www.offensivecomputing.net/.

[8] : VX Vault. http://vxvault.siri-urz.net/ViriList.php.

[9] : セキュリティレスポンス. http://www.symantec.com/ja/jp/security_response/.

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

[11] 松木隆宏，荒井　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 (2009).

[12] 高橋正和，村上純一，須藤年章，平原信昭，佐々木良一：フィールド調査によるボットネットの挙動解析，情報処理学会論文誌，Vol. 47, No. 8, p. 2512 (2006).

研究業績

学術論文（査読あり）

なし

研究会・大会等（査読なし）

（1） 神谷早紀，早川顕太，旭健作，鈴木秀和，渡邊晃，”自己複製挙動に着目したワーム検知手法の提案” 平成 26 年度電気・電子・情報関係学会東海支部連合大会論文集，Sep.2014.