卒業論文　　2011年度（平成23年度）

コードの特徴に基づく悪性プログラムの分類

慶應義塾大学 環境情報学部

氏名：碓井 利宣

担当教員

慶應義塾大学 環境情報学部

村井 純

徳田 英幸

楠本 博之

中村 修

高汐 一紀

Rodney D. Van Meter III

植原 啓介

三次 仁

中澤 仁

武田 圭史

平成24 年2 月14 日

卒業論文要旨 - 2011 年度 (平成 23 年度)

コードの特徴に基づく悪性プログラムの分類

インターネット利用の普及に伴い，ウイルスやワームなどの悪意のあるソフトウェア（以下，マルウェア）による脅威が顕在化してきた．現在，様々な機能を持った多数のマルウェアが日々新たに出現しており，ユーザを危険に晒している．この問題に対して効果的な対応を迅速に実施していくためには，新たに発見されたマルウェアを短時間で解析する必要がある．

本論文では，静的解析手法を用いてソフトウェアによりマルウェアの解析および分類を行うことで，解析者に対して，解析に先立って事前に情報を提供することを目標とする．これによって，解析者は事前情報を持っていない場合と比較して，より短時間で解析を行えるようになることが期待される．それを実現するために，自動で分類を行うことでマルウェアの種を特定する手法と，マルウェアにその動作を抽象化したラベルを付与することによって動作の概要を特定する手法の 2 つの手法を提案した．

そして，提案した手法の有効性を実証するために，提案手法を用いて自動でマルウェアを解析し分類と動作の特定を行うシステムを実装した．それを用いて研究室の保有するマルウェア検体を用いた実験を行い，検証を試みた．その結果，マルウェアを分類することによって属するファミリーを特定するとともに，API を基に解析を行うことでマルウェアの動作の概要を取得することが出来た．これにより，提案した手法によって，事前に得ておくことで解析や対策立案の際に有用な情報を自動的に取得し，提供出来ることを確認した．

本論文の成果によって，マルウェアの解析や対策の立案に先立って事前に情報を提供することが可能であることが確認出来た．これにより，解析や対策の立案を効率的に行うことが出来，増加を続けるマルウェアによる脅威に対して効果的な対応をとっていくことが出来ると期待される．

キーワード:
1. マルウェア, 2. 静的解析, 3. 情報セキュリティ, 4. インターネット

慶應義塾大学 環境情報学部

碓井 利宣

Abstract of Bachelor's Thesis - Academic Year 2011

Malware Classification based on Code's Charactaristics

With the spread of the internet, the threat which is caused by malwares such as computer viruses or worms has been appeared. Various kinds of new malwares are generated day after day in the present circumstances. Therefore, to take effective measures to cope with this problem efficiently, we need to analyze the malware which is newly discovered.

In this thesis, we aim at providing analysts with the previous information which is obtained by the software which analyzes and classifies malwares based on static analysis before they analyze by themselves. Hereby, analysts with previous information are expected to become able to analyze malwares compared with the case that they don't have previous information. To attain this purpose, we propose two methods.

And, to verify the effectiveness of these proposing methods, we implimented a system which analyzes and classificates malwares with these methods. With this system, we experimented that system by analyzing and classifying malwares that our server retains, and verify the system. As a result, we could specify the family which malwares come into by classifying them and retrieve outlines of their behavior by analyzing them with APIs. From this, we confirmed that proposing methods can acquire information which is useful for analyzing and developing countermeasures against malwares automatically and previously, and provide them to the analyst or counter planner.

For the product of this thesis, the fact that proposing methods are capable of providing information before one analyzes malwares or takes measures is verified. Herewith, we can make malware analyzing and developing countermeasures efficient and are expected to take effective steps against threats of malwares which is increasing rapidly.

Keywords :
1.Malware, 2.Static Analysis, 3.Information Security, 4.Internet

Keio University, Faculty of Environment and Information Studies

Toshinori Usui

目 次

図 目 次

表 目 次

第1章　序論

本章では，研究の背景として，新種や亜種のマルウェアが増加したことにより，マルウェアの解析や対策の実施が難化してきているという現状を述べる．そして，マルウェアの動作およびファミリーを特定する手法を確立することによって，事前に情報を提供することで研究や解析，対策の立案を支援するという目的を明らかにし，本論文中で用いる用語を定義するとともにと本論文の構成を記す.

1.1　マルウェアの現状

近年，インターネット利用者の増加に伴い，様々な脅威が顕在化してきた．それらの脅威の中でも，ウイルスやワームなどといったマルウェアは日々多くのユニークなものが生成され続けている．大手ウイルス対策ソフトウェアベンダである G Data Software 社の公開した資料である，G Data マルウェアレポート 2011 上半期 [1] によると，新種のマルウェアの発生数は増加傾向にあり，2005 年と比較するとおよそ 66 倍と大きく増加している．

マルウェアによる脅威への対策として，現在多くのユーザに利用されているものの 1 つに，ウイルス対策ソフトウェアがある．ウイルス対策ソフトウェアがマルウェアの検知に利用している手法は大きく分けて 2 つある．1 つ目は，シグネチャと呼ばれる，マルウェアのオブジェクトファイル内に見られるユニークな特徴をあらかじめ抽出しておき，それと検査対象とをマッチングすることで検知を行うパターンマッチング法である．2 つ目は，マルウェアに見られる特異な点が検査対象に探すことで検知を行うヒューリスティック法である．どちらもウイルスの検知において一定の効果を示しているが，いずれの手法も効果的に運用するためには既存のマルウェアの分析が必要不可欠である．しかし，マルウェアの巧妙化により解析が難化しているとともに，先に述べた通りマルウェアの発生数自体も増加していることにより，新しく出現したすべての検体を手作業によって解析することは限界を迎えている．したがって，より効率的に解析を進めることが出来るような手法が求められている．

1.2　本論文の目的

本論文の目的は，マルウェアをソフトウェアによって自動で解析することで動作の概要を特定するとともに，分類することによってマルウェアの種を識別し，亜種や同種などといったマルウェア間の関係性を明らかにすることである．これによって，マルウェアの解

析に先立って事前情報を解析者に提供することで解析を効率化し，より効果的な対策を迅速にとっていくことが出来るようになることが期待される．

1.3 本論文中の用語の定義

本論文中では，ラベルおよびサブルーチン動作ラベルという用語を用いる．ここでは，これらの用語についての定義を行う．ラベルとは，マルウェアあるいはサブルーチン内で利用されている Win32API の関数を動作の傾向ごとにまとめ，抽象化したものである．ラベルは，マルウェアあるいはサブルーチンに対応付ける形で付与される．付与されたラベルを参照することによって，マルウェアやサブルーチンの動作の概要を知ることが出来る．以下に，例を挙げる．あるマルウェアがファイル操作を行う Win32API を一定以上利用していた場合，そのマルウェアには [ファイル操作] というラベルを付与する．このラベルを参照することによって，そのマルウェアがファイルの書き換えや作成，削除などなんらかのファイル操作を行うことが推測できる．ラベルおよびサブルーチン動作ラベルの詳細については，第 4 章にその付与手法について，第 5 章にその実装例について詳しく述べている．

1.4 本論文の構成

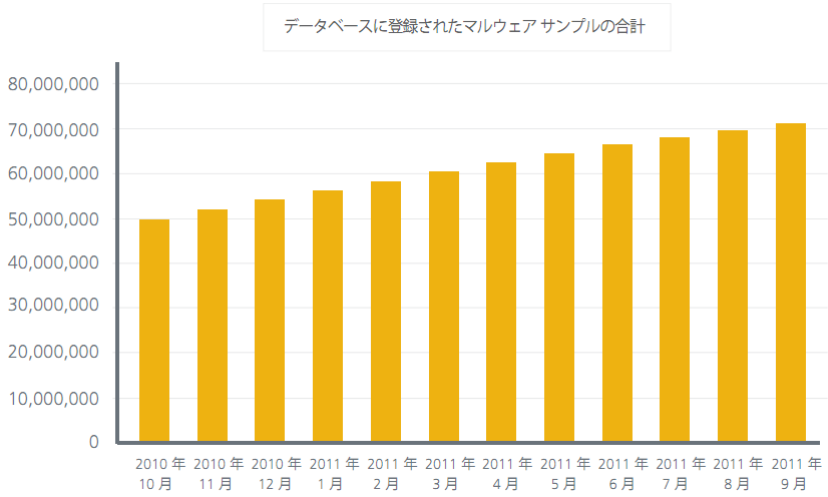
本論文は全 8 章から構成される．第 2 章では，マルウェアによる脅威とそれを取り巻く環境の現状について述べる．第 3 章では，第 2 章で述べた課題に取り組んでいる関連研究を紹介する．第 4 章では，マルウェアの動作およびファミリーを特定する手法を提案する．第 5 章では，第 4 章で述べた手法に基づいた実際にマルウェアを解析・分類するシステムの実装について述べる．第 6 章では，実装したシステムによって実際にマルウェアを解析・分類し，動作とファミリーを特定する実験を行い，その結果を評価するとともに議論を展開する．第 7 章では，第 6 章での結果を基に，考察を与える．最後に，第 8 章で本論文の結論と今後の展望を述べる．

第2章　マルウェアの脅威と対策の現状

本章では，研究の背景として，インターネット上に存在するマルウェアの現状を詳細に述べ，それらに対する対策と，対策をとる上で利用される解析手法の現状について記述する．そして，そのような状況の中での本論文の着眼点と，果たす役割を示す．

2.1　インターネット上におけるマルウェアの拡がり

現在，インターネット利用の拡大に伴って，様々な脅威も顕在化してきている．その中でもマルウェアによる脅威は多様な種類のものが存在し，日々，数多くのマルウェアが生み出されている．ウイルス対策ソフトウェアベンダである McAfee 社による McAfee 脅威レポート 2011 年第 2 四半期 [2] および McAfee 脅威レポート 2011 年第 3 四半期 [3] によると，McAfee Labs では毎月平均で約 200 万件の新しいマルウェア検体を収集してデータベースに登録しており，2011 年末には検体の総数は 7500 万件を超える可能性があることを示唆している．新しく出現したマルウェアには新種と亜種という分類がある．新種とはそれまでに確認されている検体にはない新しい機能や特徴を持ったマルウェアである．一方，亜種とは既存の検体に機能を追加するなどの拡張や，対象に合わせた改変などをしたものを指す．ウイルス対策ソフトウェアベンダである G Data Software 社による G Data マルウェアレポート 2010 年下半期 [4] によると，近年は，新しいマルウェアの発生件数が増加している一方,「種」（＝ファミリー）については必ずしも大きく増加しているというわけではなく，2009 年のマルウェア種の数は 3267 種であり，2010 年のマルウェア種の数は 3313 種であるというように，ファミリーは現在，3000 種ほどで安定していることが確認されている．これらの情報から，現在の存在するマルウェアの多くは亜種であるということが分かる．また，亜種が生み出される要因として，パッカーと呼ばれる実行コードを実行可能な形式のままで圧縮することで同じソースコードを基にして作成されたマルウェアであっても全く異なるバイナリを持つように圧縮するツールや，マルウェアの機能を GUI で選択していくことであまり知識がなくてもマルウェアを作成することが出来るツールなどがあり，さらに，攻撃対象に合わせて改変したマルウェアを利用する標的型攻撃も拡大していくと考えられている．そのため，今後も亜種が増加していくことが考えられる．



McAfee Labs データベースに登録されたマルウェアの種類の総数（亜種を含む）2010年10月 – 2011年9月
McAfee脅威レポート2011年第3四半期より抜粋

図 2.1: マルウェアの現状

2.2 マルウェアによる脅威と事例

現在，存在するマルウェアについて，どのような種類があり，どのような脅威をもたらすのかについてまとめ，対策の重要性を述べる．マルウェアは機能や感染経路によって，種類を分けることが出来る．以下に，一般的な種類の名称とそれらによる脅威について記述する．また，それぞれのマルウェアにより近年報告された事件などについてもここで紹介する．

- **コンピュータウイルス**
  コンピュータウイルスとは，自己伝染機能，潜伏機能，発病機能のいずれか一つ以上を持ち，意図的にデータの消去，改竄などを行う悪意のあるソフトウェアである．狭義では，感染の対象となるプログラムなどに感染して動作するものを指す．広義では，以下に紹介するマルウェアの総称としても用いられている．

- **ワーム**
  ワームとは，自信を複製して他のシステムに拡散するという自己増殖機能を持った悪意のあるソフトウェアである．宿主となる感染対象のプログラムを必要としない点で，狭義のコンピュータウイルスとは区別される．2009 年には Stuxnet と呼ばれるワームによってイランの核濃縮施設が攻撃されるという被害が起こっており，現在に至るまで様々な種類のワームによる脅威が確認されている．

- **バックドア**
  バックドアとは，IDやパスワードによって通信や使用を制限するコンピュータの機能を正規の許可なく利用するためにコンピュータ内に作成された通信接続機能を持ったソフトウェアである．2011年9月に三菱重工業がバックドアを添付されたメールによる攻撃を受けるなどの被害がある．

- **キーロガー**
  キーロガーとは，コンピュータへのキー入力を監視してそれを記録し，それを特定の相手に対して送信するソフトウェアである．オンラインショップなど入力したクレジットカードを始めとする情報やオンラインゲームのアカウント情報が奪取されるなどの被害のほか，企業の情報漏洩の原因ともなっている．2006年の時点でセキュリティベンダのWebsenseが行った調査によると，米企業の17%がキーロガーの被害を経験しており，大きな脅威となっている．2009年にはオレ様キーロガーと呼ばれるキーロガーがWeb上で配布されるなど，現在に至るまで被害が確認されている．

- **トロイの木馬**
  トロイの木馬とは，外見上はグッドウェアとして正常に動作しているように見える一方，実際には不正な振る舞いを行うように作成されたソフトウェアである．グッドウェアに不正な機能を付加する形で作り変えられたものが多い．近年では，SpyEyeやZeusといったオンラインバンキングの情報を奪取するトロイの木馬が多数流通している．また，Windows Live Messengerで画像を装ったトロイの木馬が感染した利用者の意思に関係なく送信されるといった事件も発生しており，大きな問題となっている．

- **スパイウェア**
  スパイウェアとは，一般ユーザのPC上で動作し，ユーザに感知されずに個人情報やパスワード，クレジットカードの情報などを収集して特定の相手に対して送信するソフトウェアである．2005年にスパイウェアを用いてインターネットバンキングの利用者から情報を不正に取得したとして逮捕者が出て以降，インターネットユーザにとって最も身近な脅威の一つであり，個人利用での感染が非常に多い．

- **スケアウェア**
  スケアウェアとは，実際には特別な機能がないにも関わらず，ユーザを脅迫することで恐怖心を煽り，金銭や個人情報を奪取することを目的としたソフトウェアである．セキュリティ上の脅威などの偽の警告を発し，その解決のために金銭や個人情報の入力を要求し，取得した情報を特定の相手に送信する．McAfee 脅威レポート2010年第4四半期 [5] では，マルウェアのうちスケアウェアが占める割合は23 %に達すると述べられており，今後も拡大していくことが考えられる．

- **ランサムウェア**
  ランサムウェアとは，ファイルやシステムを使用不能にし，その復旧と引き換えに金銭を要求するソフトウェアである．国内での被害例は多くないが，海外では主要な

マルウェアの一種類となっている．2011 年には，データを暗号化し，復号と引き換えに 125 ドルを請求する GPCode の亜種が出現し，広く流通するなど，問題となっている．

- **アドウェア**
  アドウェアとは，広告を目的としたソフトウェアである．基本的には無害であるが，一部にはマルウェアに分類される機能を持ったものもある．一般に，利用者に対する情報提供を十分に行わずに広告の表示を行うものがマルウェアと認識される．高頻度でポップアップ広告の表示を繰り返すポップアップ広告型や，Web サイトを閲覧中に広告サイトへ遷移させるリンク乗っ取り型などがある．2012 年 1 月にもアダルトサイトの料金請求画面を表示し続け，送金を促すというワンクリック詐欺を目的としたアドウェアによる被害が発生し，第 2.3.1 項に後述する不正指令電磁的記録供用罪容疑での逮捕者が出るという事件が起こっているなど，看過できない脅威である．

以上から分かる通り，マルウェアは多様な特徴を持っており，それらが引き起こす脅威が社会に与える影響も看過出来ないものである．前節で述べたようにマルウェアは増加し続けており，継続的に有効な対策をとっていくことが非常に重要である．

2.3 マルウェアへの対策

前節にて，マルウェアの対策が重要であることを確認した．ここでは，現在，どのような対策が実施されているのかについて述べる．まず，法律などの社会的にとられている対策について紹介し，その限界と技術的対策の必要不可欠性を示す．

2.3.1 社会的対策

マルウェアへの対策の 1 つとして，法律や条例などによってマルウェアの作成や配布などを禁止するというものが考えられる．ここでは，それらの国家や司法などによってにマルウェアを抑制する対策を社会的対策と呼ぶこととし，以下に紹介する．また，それらによる対策の限界について述べる．

電子計算機損壊等業務妨害罪

電子計算機損壊等業務妨害罪は，信用及び業務に対する罪として，刑法第 234 条の 2 に規定されている．コンピュータやデータを損壊する，あるいは不正な指令を与えるなどといった方法で業務を妨害した場合に適用される．後述のコンピュータウイルス作成罪が成立するまで，マルウェアによる犯罪の一部にはこの法律が適用されていた．一方，業務性を要すること，マルウェアの作成や所持などのみには適用出来ないことなどの限界がある．以下に，条文を示す．

【第三十五章　信用及び業務に対する罪】
（電子計算機損壊等業務妨害）
第二百三十四条の二　人の業務に使用する電子計算機若しくはその用に供する電磁的記録を損壊し、若しくは人の業務に使用する電子計算機に虚偽の情報若しくは不正な指令を与え、又はその他の方法により、電子計算機に使用目的に沿うべき動作をさせず、又は使用目的に反する動作をさせて、人の業務を妨害した者は、五年以下の懲役又は百万円以下の罰金に処する。
２ 前項の罪の未遂は、罰する。

電子計算機使用詐欺

電子計算機使用詐欺は，詐欺及び恐喝の罪として，刑法第 246 条の 2 に規定されている．主にコンピュータに不正な指令を与えることで詐欺を働いた場合に適用される．マルウェアによる犯罪としては，スパイウェアなどを用いて他人の銀行口座から不正に引き出すなどを行った場合に適用されていた．以下に，条文を示す．

【第三十七章　詐欺及び恐喝の罪】
（電子計算機使用詐欺）
第二百四十六条の二　前条に規定するもののほか、人の事務処理に使用する電子計算機に虚偽の情報若しくは不正な指令を与えて財産権の得喪若しくは変更に係る不実の電磁的記録を作り、又は財産権の得喪若しくは変更に係る虚偽の電磁的記録を人の事務処理の用に供して、財産上不法の利益を得、又は他人にこれを得させた者は、十年以下の懲役に処する。

コンピュータウイルス作成罪

コンピュータウイルス作成罪は，正式には「不正指令電磁的記録に関する罪」という．マルウェアによる犯罪の増加に伴い，適切な対応をしていくために，2011 年に施行された．コンピュータに不正な指令を与える電磁的記録，すなわちマルウェアの作成や取得などに関する犯罪であり，刑法第 168 条の 2 および 3 において規定されている．これは正当な理由なく人のコンピュータにおいて実行する目的で，マルウェアを作成，提供，取得，保管することを禁止した法律であり，マルウェアの拡大を抑える目的で施行された．以下に，条文を示す．

【第十九章の二　不正指令電磁的記録に関する罪】
（不正指令電磁的記録作成等）
第百六十八条の二　正当な理由がないのに、人の電子計算機における実行の用に供する目的で、次に掲げる電磁的記録その他の記録を作成し、又は提供した者は、三年以下の懲役又は五十万円以下の罰金に処する。
一 人が電子計算機を使用するに際してその意図に沿うべき動作をさせず、又はその意図に反する動作をさせるべき不正な指令を与える電磁的記録
二 前号に掲げるもののほか、同号の不正な指令を記述した電磁的記録その他の記録
２ 正当な理由がないのに、前項第一号に掲げる電磁的記録を人の電子計算機における実行の用に供した者も、同項と同様とする。
３ 前項の罪の未遂は、罰する。

（不正指令電磁的記録取得等）
第百六十八条の三　正当な理由がないのに、前条第一項の目的で、同項各号に掲げる電磁的記録その他の記録を取得し、又は保管した者は、二年以下の懲役又は三十万円以下の罰金に処する。

これらの社会的対策がとられている一方で，その限界も存在する．マルウェアの現状として，マルウェアによる攻撃元の国を解析すると，日本国外からも多くの攻撃が行われている．また，マルウェアを組織的に作成，配布することで犯罪を行うといったケースも少なくない．さらに，手法が巧妙化しているため，作成者や配布者を特定することも容易ではない．これらの理由から，前述の社会的な対策によってマルウェアの広がりを完全に抑制することは難しい．したがって，社会的な対策のみによらず，技術的な対策をとることが非常に重要であることが確認出来る．

2.3.2　技術的対策

現在，マルウェアへの対策として広く利用されている技術として，ウイルス対策ソフトウェアがある．ここでは，多くのウイルス対策ソフトウェアで行われているマルウェアへの対策手法を挙げ，マルウェアに対して有効な対策を取るためには，事前に解析することが重要であることを確認する．マルウェアの検知に用いられている手法の主要なものとして，パターンマッチング検知手法とヒューリスティック検知手法の２つが挙げられる．以下に，それぞれの検知手法とその特徴についての詳細を述べる．

パターンマッチング検知手法

パターンマッチング検知手法では，マルウェアのオブジェクトファイル内に見られるユニークな特徴をシグネチャとしてあらかじめ抽出しておき，それと検査対象とを比較し

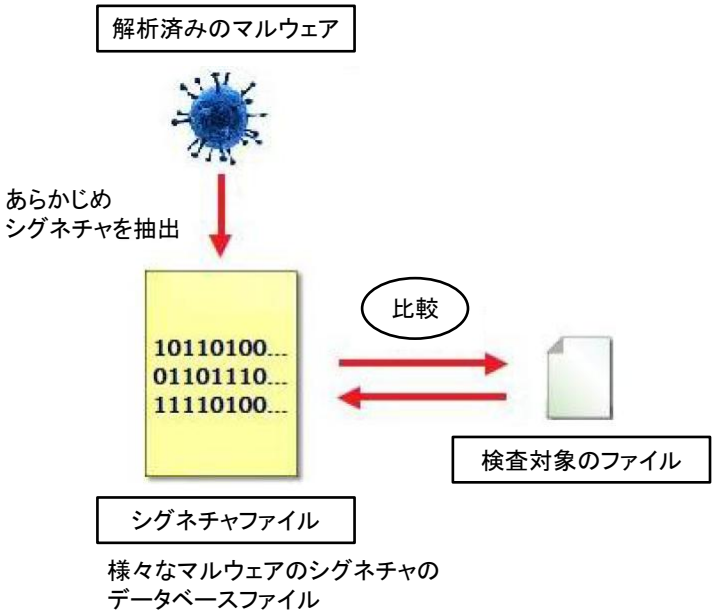


図 2.2: パターンマッチング検知手法の概要図

て検査対象にシグネチャが含まれるかどうかでマルウェアの検知を行う．既にシグネチャのあるマルウェアと同種のマルウェアであれば，非常に高い精度で検知を行うことが可能である．一方で，この手法が有効なのは既に解析されていてシグネチャが抽出済みのマルウェアのみである．また，この手法では，新種や亜種などのマルウェアに対応することが出来ない．そのため，パターンマッチング検知手法による検知を行っていくためには，第 2.1 節で述べた通り増加を続ける亜種を，解析し続ける必要がある．図 2.2 に，パターンマッチング検知手法の概要を示す．

ヒューリスティック検知手法

ヒューリスティック検知手法では，検査対象の挙動などを解析し，マルウェアに見られる傾向のある特異な点を探すことで検知する検知手法である．数理モデルや学習モデルを利用して検知を行う手法が一般的であるため，パターンマッチング検知手法とは異なり，未解析の新種や亜種であっても検知を行うことが可能である．ウイルス対策ソフトウェアベンダである Kaspersky によるマルウェアマンスリーレポート 2011 年 9 月 [6] によると，当月のユーザの PC 上で検知・駆除されたマルウェア数が 263,437,090 であったのに対し，ヒューリスティック検知数が 91,767,702 と，ヒューリスティック検知手法による検知が全検知数の 3 割を超えている．このことから分かる通り，ヒューリスティック検知手法はマルウェア対策の一端を担っている．一方，この手法には誤検知の可能性もある．誤検知は大別して 2 種類が存在する．マルウェアとして検知するべき検査対象を正しく検知出来ない

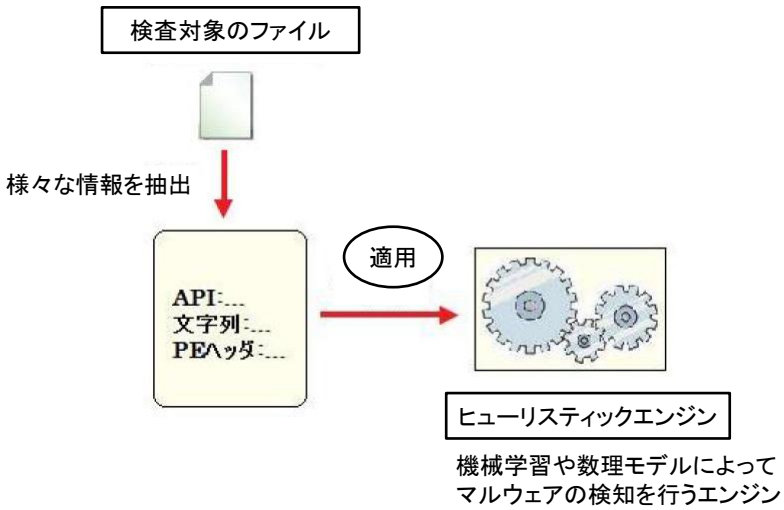


図 2.3: ヒューリスティック検知手法の概要図

フォールスネガティブと，マルウェアでない検査対象をマルウェアとして検知してしまうフォールスポジティブである．フォールスネガティブが存在することによって，マルウェア作成者は検知を避けることが可能であり，フォールスポジティブが存在することによって，グッドウェアをマルウェアとして検知しブロックしてしまうという，ユーザにとっての問題が発生する．誤検知を減らすためには，多くのマルウェア検体を検知に先立って解析し，結果を十分に蓄積する必要がある．そのため，検体を解析することは，ヒューリスティック検知手法においても非常に重要であると言える．図 2.3 に，ヒューリスティック検知手法の概要を示す．

2.4 マルウェアの解析手法の現状

マルウェアを解析する手法には，大きく分けて静的解析と動的解析の 2 つがある．以下に，それぞれの手法とその特徴についての詳細を述べる．

2.4.1 静的解析手法

静的解析手法とは，マルウェアを実際に動作させることなく，そのオブジェクトコードをデバッガや逆アセンブラを使用して解析し，感染活動を読み解くという解析手法であ

る．動作の解析を詳細に行うことが出来る，マルウェアを実際に動作させることがないため，比較的安全に解析を進めることが出来るなどの特徴がある．

2.4.2 動的解析手法

動的解析手法とは，マルウェアを実際に動作させ，その際に感染活動を記録するツールを動かしておくことで感染活動を明らかにするという解析手法である．比較的短時間で動作を解析することが出来る，要求される技術的ハードルが比較的低い，などの特徴がある．

2.5 マルウェア解析における課題

現状のマルウェアの解析手法には，それぞれ課題が存在する．動的解析手法の課題としては，すべてのマルウェアに対して有効とは言えないということが挙げられる．具体的には，動的解析ツールを検出する機能をマルウェアに付加することで，動的解析をしようとしていることを検知して妨害する，動作条件が設定されていることによって，一定の条件を満たすまで潜伏しており，解析が出来ない，などといったことがある．静的解析手法の課題としては，解析に時間がかかることが第一に挙げられる．また，オブジェクトコードの解析には幅広い知識と経験が必要である．また，手動で解析を行う場合，マルウェアの挙動を詳細に渡って把握することが出来るが，ソフトウェアなどにより自動で解析を行う場合，挙動を詳しく知る手法は現状では十分に確立されているとは言えない．

2.6 本論文の着眼点

本論文は，第 2.5 節で述べた，静的解析手法の課題に焦点を当て，ソフトウェアによる静的解析の中で，マルウェアの動作を知ることが出来る形を提案する．特に，ソフトウェアによって自動的に分類する際を対象とする．これは，増加し続ける多数のマルウェアの解析に際して，第一段階の情報を得るための高速な解析手法として，プログラムによる静的解析が有効であり，手動での解析に先立って分類しておくことが一助となるためである．本論文では，分類結果からマルウェアの挙動の傾向を知ることが出来る形でマルウェアの分類を行う手法を提案し，実際に分類を行うシステムを実装する．これによって，マルウェアの解析や対策の立案を効率化する．

2.7 まとめ

本章では，マルウェアによる脅威の拡大と現状を示すとともに，現在とられている対策とその根幹をなす解析手法について述べ，その課題を取り挙げた．増加し続けるマルウェアの脅威に対して効果的な対策をとり続けるためには解析を続ける必要がある．しかし，

解析が必要となる新種や亜種のマルウェアの増加に対して，解析の効率化は十分とは言えない．そこで，ソフトウェアによって自動的に解析および分類を行う手法を提案する．ラベルを用いたマルウェアの動作の傾向の解析や，分類によるファミリーの特定を行うシステムを実装し，それによって，マルウェアの解析や対策に先立って有効な情報を提供する．また，実験を行い，本手法によって得られる情報がどの程度有効なのかを検証する．

第3章　関連研究

本章では，マルウェアの静的解析や分類に関する既存研究について述べる．また，機械学習手法を用いて分類を行っている既存研究についても言及する．

3.1　マルウェアの分類に関する研究

ここでは，静的解析手法に基づいたマルウェアの分類についての既存研究について分析し，記述する．

3.1.1　機械語命令列の類似性に基づく自動マルウェア分類システム

岩村らの研究 [7] では，マルウェアのコード領域の機械語命令列から動的計画法を用いて Longest Common Subsequence（以下，LCS）を抽出し情報として用いている．また，抽出した LCS の情報をビットベクトル化することで演算を高速化する手法を提案している．これによって，100 倍以上の速さでの高速な分類が可能となっている．一方，単純な命令の比較による類似度算出手法に基づく分類であるため，最適化のレベルなどといったコンパイラオプションの変更やコンパイラの種類の違いにより生まれるオブジェクトコードの差異への対応が今後の課題となっている．また，分類のみに焦点を当てており，マルウェアの動作などは把握できない．

3.1.2　マルウェアコードの類似度判定による機能推定

安本らの研究 [8] では，未知のマルウェアと機能の判明している既知のマルウェアのバイナリコードを比較し，高い類似度が得られた判定結果を基にマルウェアの持つ機能を推定している．具体的には，複数のマルウェアとの距離を基に機能集合の包含関係を把握し，重みと閾値を用いて各関数がマルウェアに含まれるかどうかの指標を算出している．本手法によって，既知のマルウェアの亜種の機能を解析することなく推定することが可能であるが，低い類似度しか得られていない場合には機能を推定することが出来ないとも述べられている．また，機能の推定の際に用いる重みや閾値によって必ずしも正確な機能が得られるとは言えず，持っているはずの機能を検出出来ないフォールスネガティブや，持っていない機能を検出してしまうフォールスポジティブが起こっていた．

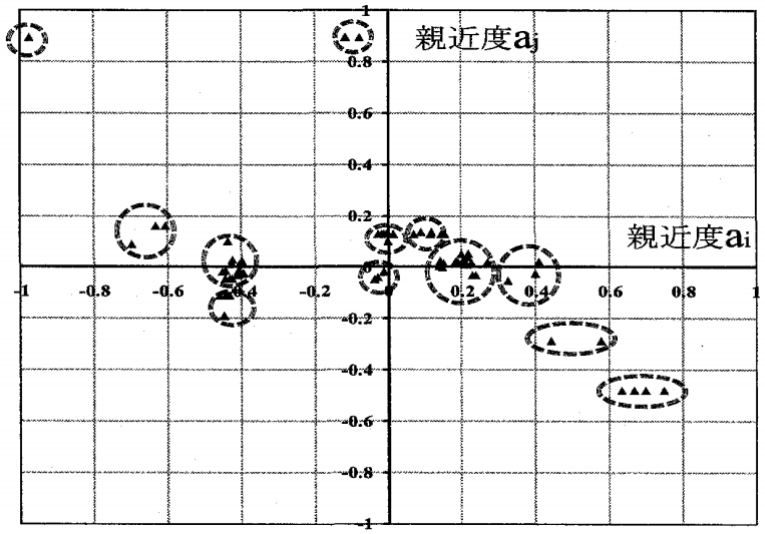


図 分類結果からの未知検体グループ作成
マルウェアの分類方法とその応用に関する考察 2009, 勝手

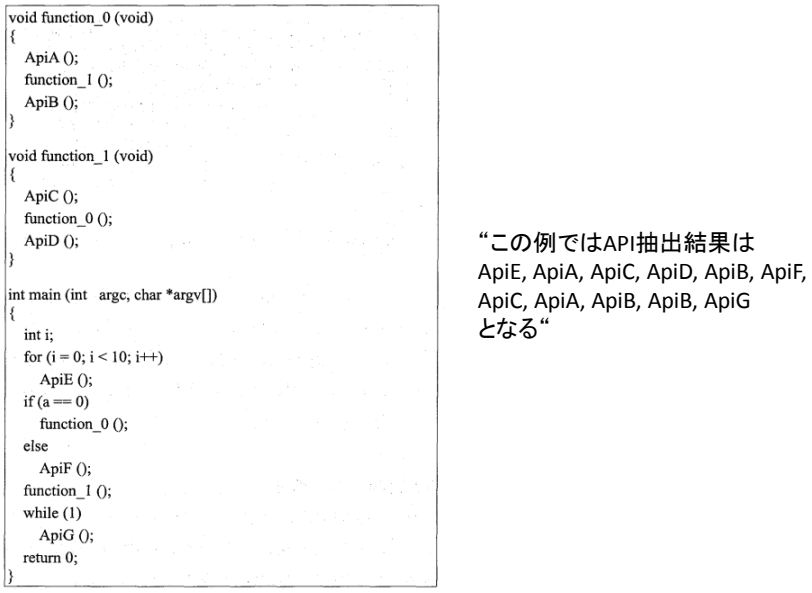
図 3.1: 分類結果の可視化表現

3.1.3 マルウェアの分類方法とその応用に関する考察

勝手らの研究 [9] では，バイナリコードの類似度と使用 API の種類からそれぞれ親近度を算出し，それに基づいた数量化 IV 類によって分類結果を多次元空間内に配置した．これによって，分類結果の直感的な理解を可能とするために，検体間の関係性の可視化表現を与えた．論文内で提示されている分類結果の可視化表現の様子を図 3.1 に示す．また，親近度が高く空間上での配置が密な検体同士の機能や特性が同一であり，ウイルス対策ソフトウェアでも亜種として検知されることを確認した．亜種の特定に有効な手法であると言えるが，機能推定については，API を基にマルウェアの親近度を計算していることより，新しい機能が追加された亜種の場合は親近度が低く出る可能性があり，必ずしもすべてのマルウェアにおいて正確に機能を推定出来るとは限らない．

3.1.4 コンピュータウイルスのコード静的解析による特徴抽出と分類について

岩本らの研究 [10] では，関数の呼び出し関係に着目した API 解析と制御フロー解析，API 推移解析の 3 つを行っている．これによって亜種間の距離を算出し，分類を行っている．いずれの手法でも良好な結果が得られているが，関数呼び出しに着目した API 抽出では，関数の内部でコードの順番を入れ替えたり，独立して行われる処理の順番を入れ替えるなどの方法によって解析を困難にすることが可能であることが述べており，制御フ



```
void function_0 (void)
{
   ApiA ();
   function_1 ();
   ApiB ();
}

void function_1 (void)
{
   ApiC ();
   function_0 ();
   ApiD ();
}

int main (int   argc, char *argv[])
{
   int i;
   for (i = 0; i < 10; i++)
      ApiE ();
   if (a == 0)
      function_0 ();
   else
      ApiF ();
   function_1 ();
   while (1)
      ApiG ();
   return 0;
}
```

“この例ではAPI抽出結果は
ApiE, ApiA, ApiC, ApiD, ApiB, ApiF,
ApiC, ApiA, ApiB, ApiB, ApiG
となる“

コンピュータウイルスのコード静的解析による特徴抽出と分類について 2007, 岩本

図 3.2: 関数の呼び出し関係による API 解析

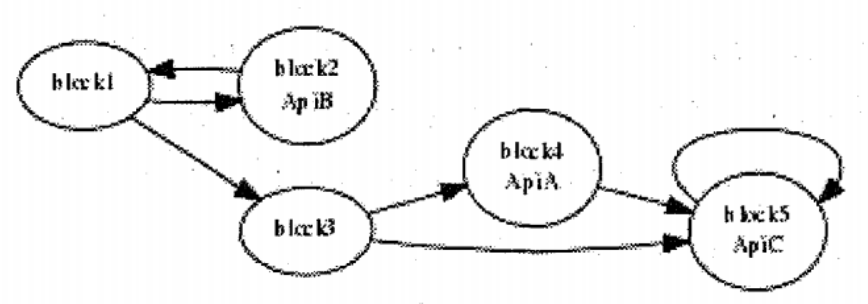
ロー解析や API 推移解析には時間がかかると言及している．また，類似度の算出のみに焦点を当てており，動作の掲示などは行っていない．図 3.2 に関数の呼び出し関係による API 解析を，図 3.3 に制御フロー解析を，図 3.4 に API 推移解析を示す．

3.2 マルウェアの解析に関する研究

ここでは，分類に留まらず，マルウェアの検知などに際して，特に本研究に関わる数理モデルや学習モデルを用いるものや特殊なマルウェアの抽出情報を利用しているものについて分析し，記述する．

3.2.1 実行ファイルに含まれる文字列の学習に基づくマルウェア検出方法

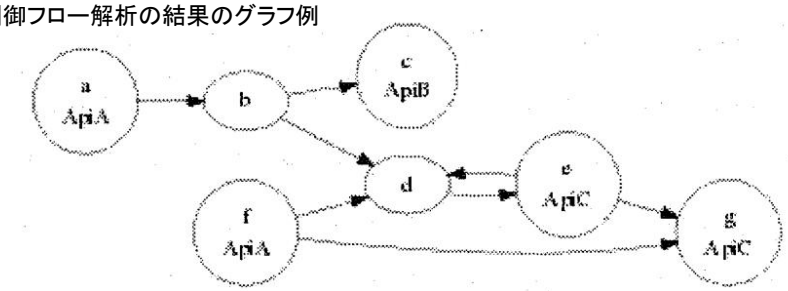
戸部らの研究 [11] では，Support Vector Machine による教師あり学習をコーパスを用いてマルウェアのオブジェクトファイル中の文字列に適用し，マルウェアのパターン認識を行っている．良好な認識精度を持っており，動作も高速である．しかし，オブジェクトファイル中に特徴的な文字列が存在する必要があり，また，マルウェアの動作や機能についても着目していない．



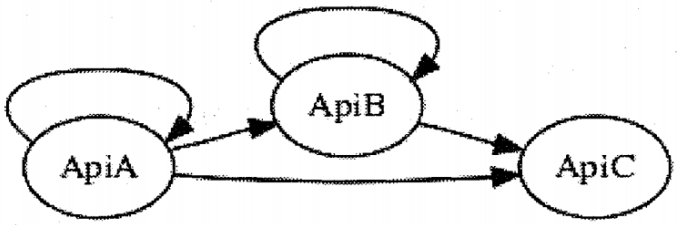
コンピュータウイルスのコード静的解析による特徴抽出と分類について 2007, 岩本

図 3.3: 制御フロー解析

制御フロー解析の結果のグラフ例



上のグラフをAPI推移グラフに変換したグラフ



コンピュータウイルスのコード静的解析による特徴抽出と分類について 2007, 岩本

図 3.4: API 推移解析

3.2.2 Effective and Efficient Malware Detection at the End Host

Kolbitsch らの研究 [12] では，API の流れを記した挙動グラフを用いて，エンドホストでのマルウェアの検知を行っている．岩本らの研究 [10] で用いられた手法の 1 つと同様のもの用いて機械学習により検知を行っている．検知率も高く高速であり，API を基にした機械学習による検知の実行可能を示している．

3.2.3 Malware Detection based on Mining API Calls

Sami らの研究 [13] では，API 呼び出しという情報を，単純に呼び出している種類のみでなく，呼び出しの流れまでを考慮したコール・ツリーのグラフとして抽出する手法を提案している．この手法によって，API に関する情報を，より情報量が多く有効な形で利用することが出来る．しかし，岩村らの研究 [7] などでは，コール・ツリーが同じであっても，ツリーの葉の部分にあたる関数の中身が異なる場合は，やはり手動でアセンブリを見て判断をする必要が出てくるという点で，必ずしも常に有効とは言えないという点が指摘されている．

3.3 まとめ

本章では，静的解析手法によって得られる情報によってマルウェアを分類する研究とマルウェアの静的解析に主眼を置いた研究，機械学習手法による分類の研究の 3 つの種類の研究について紹介し，その手法について述べた．分類結果からマルウェアのファミリーや亜種を特定し，機能を推定出来ることが示されており，適切に分類することの有効性が分かる．そのため，いずれの研究も，様々な手法を用いて分類の高精度化や効率化を図っている．一方で，既知のマルウェアとの比較や所属するファミリーといった情報を基に機能を推定しようとする研究が多い．そのような手法では，新種のマルウェアや新しく機能を追加された亜種のマルウェアに対しては，十分な効果を発揮できない．そのため，分類結果のみによらない機能推定手法を備えた分類手法が必要であると言える．

第4章　動作特定可能性を考慮したマルウェアの分類手法

マルウェアに対して効果的な対策をとるためには，マルウェアを解析することが必要である．しかし，手動により詳細に渡る解析を行うのは時間的・人的コストがかかる．そのため，ソフトウェアによる自動的な解析によって事前に情報を得ることが望まれる．本研究では，第2章で述べたように，ソフトウェアによって，マルウェアに対して動作の概要を示すラベルを付与するとともに，分類を行うことでマルウェアの種（ファミリー）を推定することに焦点を当てている．

本章では，まず，前提としてマルウェアの解析を行う上で重要となるオブジェクトファイルの内部構造を示す．そして，ソフトウェアによって解析を行い，マルウェアにラベルを付与し，分類するという一連の過程で重要となる手法について述べる．具体的には，アンパッキング手法，APIの解析手法，サブルーチンの解析手法，ラベル付与手法，分類手法の4つの手法について述べる．

4.1　前提

マルウェアを解析するためには，オブジェクトファイルがどのような形式となっているかを把握しておくことが重要である．ここでは，手法の説明に先立って，オブジェクトファイルの内部構造について明らかにする．オブジェクトファイルは，先頭から順にヘッダ，セクションヘッダ，数個のセクションによって構成されている．以下に，各部の説明を示す．

4.1.1　ヘッダ

ヘッダは，先頭から順にIMAGE_DOS_HEADER構造体とPEヘッダによって構成されている．図4.1にヘッダの構造を示す．IMAGE_DOS_HEADER構造体については，本研究では重要でないため省略し，以下では，PEヘッダについて詳細を示す．PEヘッダは，IMAGE_NT_HEADERS32構造体によって成立している．この構造体はwinnt.hにて定義されている．以下に，その構造を示す．IMAGE_NT_HEADERS32構造体は，PEシグネチャ，IMAGE_FILE_HEADER構造体，IMAGE_OPTIONAL_HEADER32構造体によって構成されている．各要素についての詳細を示す．

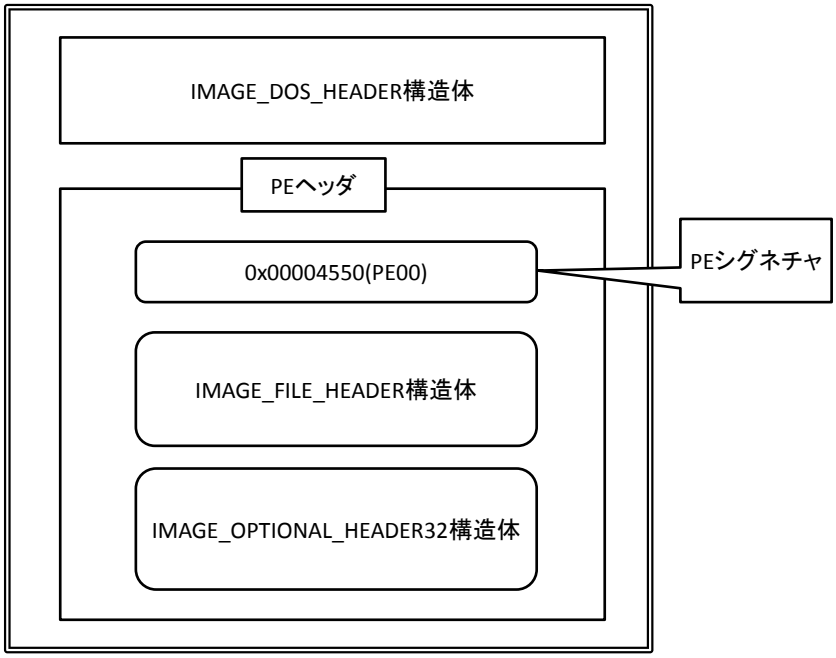


図 4.1: ヘッダの構造

```
typedef struct _IMAGE_NT_HEADERS {
    DOWRD Signature;
    IMAGE_FILE_HEADER FileHeader;
    IMAGE_OPTIONAL_HEADER32| OptionalHeader;
}IMAGE_NT_HEADERS32, *PIMAGE_NT_HEADERS32;
```

PE シグネチャ

PE シグネチャは，PE ヘッダの先頭部分を明示するために存在する 4 バイトのシグネチャである．0x00004550（ASCII コードでは PE00）という値を保持している．

IMAGE_FILE_HEADER 構造体

IMAGE_FILE_HEADER 構造体には，動作想定マシンや作成日時などの，オブジェクトファイルの属性についての情報が格納されている．本研究においては重要でないため，省略する．

IMAGE_OPTIONAL_HEADER32 構造体

IMAGE_OPTIONAL_HEADER32 構造体には，オブジェクトファイルの構造についての情報が格納されている．以下に IMAGE_OPTIONAL_HEADER32 構造体の構成を示すとともに，そのメンバのうち，主要なものについての詳細を記述する．

```
typedef struct _IMAGE_OPTIONAL_HEADER {
    // Standard fields.

    WORD    Magic;
    BYTE    MajorLinkerVersion;
    BYTE    MinorLinkerVersion;
    DWORD   SizeOfCode;
    DWORD   SizeOfInitializedData;
    DWORD   SizeOfUninitializedData;
    DWORD   AddressOfEntryPoint;
    DWORD   BaseOfCode;
    DWORD   BaseOfData;

    // NT additional fields.

    DWORD   ImageBase;
    DWORD   SectionAlignment;
    DWORD   FileAlignment;
    WORD    MajorOperatingSystemVersion;
    WORD    MinorOperatingSystemVersion;
    WORD    MajorImageVersion;
    WORD    MinorImageVersion;
    WORD    MajorSubsystemVersion;
    WORD    MinorSubsystemVersion;
    DWORD   Win32VersionValue;
    DWORD   SizeOfImage;
    DWORD   SizeOfHeaders;
    DWORD   CheckSum;
    WORD    Subsystem;
    WORD    DllCharacteristics;
    DWORD   SizeOfStackReserve;
    DWORD   SizeOfStackCommit;
    DWORD   SizeOfHeapReserve;
    DWORD   SizeOfHeapCommit;
    DOWRD   LoaderFlags;
    DWORD   NumberOfRvaAndSizes;
    IMAGE_DATA_DIRECTORY DataDirectory
 [IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
} IMAGE_OPTIONAL_HEADER32, *PIMAGE_OPTIONAL_HEADER32;
```

- **SizeOfCode**
  SizeOfCode には，コードセクションのサイズが格納される．

- **AddressOfEntryPoint**
  AddressOfEntryPoint には，コードセクション上のプログラム開始位置であるエントリーポイントの値が格納されている．エントリーポイントは，イメージベースからの相対アドレス値で表現される．

- **BaseOfCode**
  BaseOfCode には，メモリ上にロードされたときのコードセクションの位置が格納される．

- **DataDirectory**
  DataDirectory には，データディクショナリが格納される．データディクショナリはオブジェクトファイル中の各領域のアドレスとサイズを示すものであり，IMAGE_DATA_DIRECTORY 構造体によって表される．データディクショナリは，一般に，16 個の要素からなる．したがって，DataDirectory は 16 個の IMAGE_DATA_DIRECTORY 構造体の配列で構成されている．以下に，IMAGE_DATA_DIRECTORY 構造体の構成を示し，表 4.1 にデータディクショナリの各要素を示す．

```
typedef struct _IMAGE_DATA_DIRECTORY {
    DWORD   RVA;
    DWORD   Size;
} IMAGE_DATA_DIRECTORY, *PIMAGE_DATA_DIRECTORY;
```

4.1.2 セクションヘッダ

セクションヘッダは，オブジェクトファイルが持つセクションの数と同数の IMAGE_SECTION_HEADER 構造体によって構成されている．IMAGE_SECTION_HEADER 構造体には，各セクションの特性についての情報が指定されている．図 4.2 にセクションヘッダの構造を示す．

4.1.3 セクション

各セクションは，セクションヘッダの IMAGE_SECTION_HEADER 構造体に指定された特性の通りに配置されている．各セクションの配置順序は，対応する IMAGE_SECTION_HEADER 構造体の配置順序と同様になる．図 4.3 にセクションの構造を示す．また，各セクションはそれぞれ特徴的なデータ構造を持つ．ここでは，本研究と関係の深いインポートセク

表 4.1: データディクショナリの要素

要素名	内容	備考
Export Table	エクスポートテーブルのアドレスとサイズ	
Import Table	インポートテーブルのアドレスとサイズ	本論文で特に用いる
Resource Table	リソーステーブルのアドレスとサイズ	
Exception Table	例外テーブルのアドレスとサイズ	
Certificate Table	属性認証テーブルのアドレスとサイズ	
Base Relocation Table	ベース再配置テーブルのアドレスとサイズ	
Debug	デバッグデータの開始アドレスとサイズ	
Architecture	アーキテクチャ固有データのアドレスとサイズ	
Global Ptr	グローバルポインタレジスタの RVA	
TLS Table	スレッドローカルストレージテーブルのアドレスとサイズ	
Load Config Table	ロードコンフィグレーションテーブルのアドレスとサイズ	
Bound Import	バウンドインポートテーブルのアドレスとサイズ	
IAT	インポートアドレステーブルのアドレスとサイズ	本論文で特に用いる
Delay Import Descriptor	遅延インポート記述子のアドレスとサイズ	
Reserved		確保領域

ションの構造について記述する．ここで紹介する構造を基にして，後述の API 解析を行う．インポートセクションには，ローダによってオブジェクトファイルのイメージがメモリ上にロードされた際に，外部 DLL の動的リンクの準備を行うために必要な情報が格

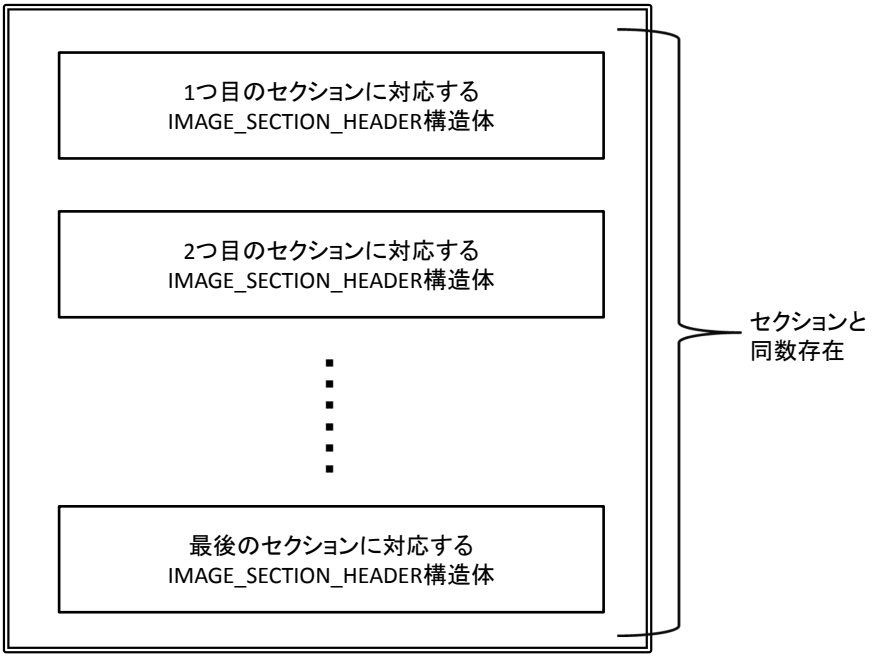


図 4.2: セクションヘッダの構造

納されている．インポートセクションは，先頭から順にリンクする DLL の個数と同数の IMAGE_IMPORT_DESCRIPTOR 構造体と 20 バイトの NULL 領域，リンクする DLL の個数と同数の DLL 名，ルックアップテーブル，ヒント名，IAT によって構成されている．以下に，各部の詳細を示す．

- **IMAGE_IMPORT_DESCRIPTOR 構造体群**
  インポートセクションの先頭には，複数の IMAGE_IMPORT_DESCRIPTOR 構造体が配置されている．この構造体は動的リンクする DLL の個数だけ存在する．その後には 0x00 によって埋め尽くされた IMAGE_IMPORT_DESCRIPTOR 構造体と同じサイズ（20 バイト）の NULL 領域が配置される．以下に IMAGE_IMPORT_DESCRIPTOR 構造体の構成を示し，以下にそのメンバのうち，主要なものを示す．

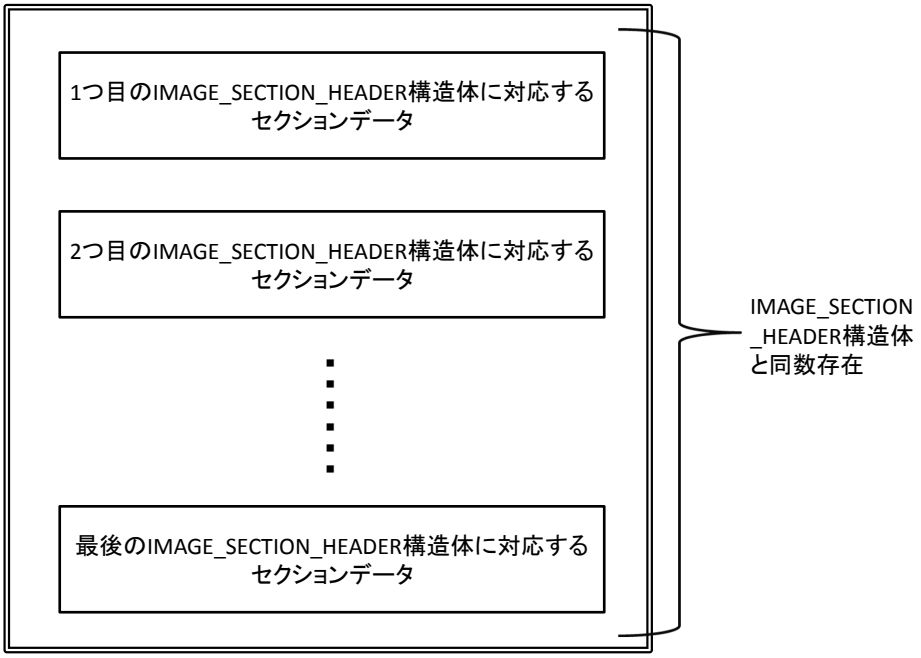


図 4.3: セクションの構造

```
typedef struct _IMAGE_IMPORT_DESCRIPTOR {
    DWORD   OriginalFirstThunk;
        // RVA to original unbound IAT
        // (PIMAGE_THUNK_DATA)
    DWORD   TimeDateStamp;
    DWORD   ForwarderChain;
        // -1 if no forwarders
    DWORD   Name;
    DWORD   FirstThunk;
        // RVA to IAT
        // (if bound this IAT has actual addresses)
} IMAGE_IMPORT_DESCRIPTOR;
typedef IMAGE_IMPORT_DESCRIPTOR UNALIGNED
                                *PIMAGE_IMPORT_DESCRIPTOR;
```

– **OriginalFirstThunk**
OriginalFirstThunk には，後述するルックアップテーブル内にある DLL の IMAGE

_THUN
K_DATA32 共用体の配列の先頭要素を指す RVA が格納される．IMAGE_THUNK_DATA の配列が示すのは，おおむね IMAGE_IMPORT_BY_NAME 構造体への RVA である．また，OriginalFirstThunk の最上位ビットが立っている場合のみ，残りの 31 ビットがインポートしている API の序数を示す．以下に IMAGE_THUNK_DATA32 共用体の構成を示す．

- **TimeDataStamp**
  TimeDateStamp には，動的リンク時のタイムスタンプが格納される．オブジェクトファイルの状態では動的リンクがされていないため，NULL が格納される．
- **ForwarderChain**
  ForwarderChain には，エクスポート転送を許可する際には-1 が，許可しないときは 0 が格納される．
- **Name**
  Name には，DLL 名を指す RVA が格納される．
- **FirstThunk**
  FirstThunk には，DLL のインポートテーブルが羅列されている領域の先頭アドレスが RVA で格納されている．

```
typedef struct _IMAGE_THUNK_DATA32 {
    union {
        DWORD ForwarderString;      // PBYTE
        DWORD Function;             // PDWORD
        DWORD Ordinal;
        DWORD AddressOfData;        // PIMAGE_IMPORT_BY_NAME
    } u1;
} IMAGE_THUNK_DATA32;
typedef IMAGE_THUNK_DATA32 *PIMAGE_THUNK_DATA32;
```

- DLL 名
  この領域には，DLL の名前が 1 つにつき 16 バイトの領域に格納されている．DLL 名が 16 バイトに満たない場合，名前を格納した残りの領域は NULL で埋められる．前述の通り，動的リンクする DLL の個数分だけ存在する．
- ルックアップテーブル
  ルックアップテーブルには，IMAGE_THUNK_DATA32 共用体が列挙されている．この共用体には，後述するヒントテーブル内にある，関数名を保持する IMAGE_IMPORT_BY_NAME 構造体への RVA が格納されている．まず，1 つ目の DLL 内に存在する API 関数の

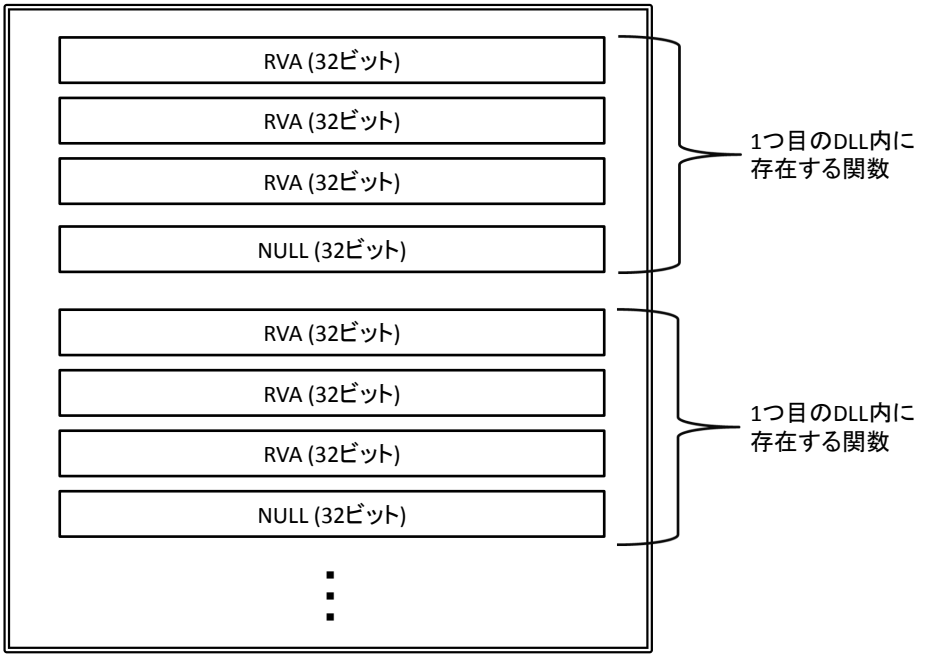


図 4.4: ルックアップテーブルの構造

関数名への RVA が続き，終端に 4 バイト分の NULL 領域を挟んで，2 つ目の DLL についてが同様に続く．このようにして，全ての DLL 内にある API 関数について記述されるまで続いている．ルックアップテーブルは，IMAGE_IMPORT_BY_NAME 図 4.4 にルックアップテーブルの構造を示す．

- ヒントテーブル
  ヒントテーブルには，IMAGE_IMPORT_BY_NAME 構造体の配列で構成されている．この構造体には，API 名が格納される．0 が格納される DWORD 値とそれに続く可変長の文字列の集合によって構成されている．文字列は NULL 文字で終わる．各々の API 名の開始位置が偶数に合わせられるため，必要に応じて文字列の終端の NULL 文字が 2 バイト分になる．以下に，IMAGE_IMPORT_BY_NAME 構造体の構成を示す．

```
typedef struct _IMAGE_IMPORT_BY_NAME {
    WORD    Hint;
    BYTE    Name[1];
} IMAGE_IMPORT_BY_NAME, *PIMAGE_IMPORT_BY_NAME;
```

- インポートアドレステーブル
  インポートアドレステーブルには，ルックアップテーブルと同様に，ヒントテーブ

ル内にある関数名の先頭アドレスが RVA で格納されている．相違点として，動的リンク時に実際の API 関数の実行アドレスに置き換えられることが挙げられる．

4.2 解析準備

第 2 章で述べたように，マルウェアを静的解析手法によって解析する前には，解析する上でのいくつかの障害を取り除く必要がある．本節では，パッキングされたマルウェアのアンパックと Import Address Table（以下，IAT）の再構築を行う手法について述べる．

4.2.1 アンパック

マルウェアは通常，パッキングによって，そのオブジェクトコードのほとんどの領域が解読出来ない形で圧縮されている．これは，静的解析手法によって解析を進めていく際に，バイナリを基にした解析が出来なくなるために大きな障害となる．ここでは，パッキングされたマルウェアをどのようにしてアンパックするかについて述べる．マルウェアのアンパックは，隔離された仮想環境を構築し，その仮想環境内でデバッガを用いてマルウェアを実際に実行して，メモリ上に展開された実行ファイルのメモリダンプを取得することで行う．パッキングされたマルウェアは，実行する際にアンパックされるため，これによってマルウェアの正味のオブジェクトコードを得ることが出来る．

4.2.2 Import Address Table の再構築

第 2 章で述べた手法によってアンパックを行った場合，取得されるのはネイティブコードではなく，ローダによってメモリ上に展開されたコードのダンプとなる．プログラムがローダに読み込まれる際には，IAT には Import Name Table（OriginalFirstThunk 配列の以下，INT）と同様の情報が書き込まれている．一方，ローダによって読み込まれた後は，IAT にはローダによって解決された実効アドレスが格納される．IAT に実効アドレスが格納されていると，ローダによって読み込んだ際と同様の環境でしかインポートされている API を特定することが出来ず，解析の障害となる．そのため，メモリダンプを取得することによってアンパックをした後には，IAT の再構築を行う必要がある．IAT の再構築とは，ローダによって解決される前の値を格納することであり，IAT の `IMAGE_IMPORT_DESCRIPTOR` 構造体の各々が，対応する INT の `IMAGE_IMPORT_BY_NAME` の値と同様になるようにする．

4.3 API 解析手法

API に基づいた解析をしていくためには，マルウェアがインポートしている API についての情報が必要となる．本研究では，静的解析手法によって API の情報を取得し，それを主に用いて後述の分類やラベル付けを行う．そのため，ここでは，オブジェクトファ

イルを参照することによって API を取得する手法を明らかにする．マルウェアの API や DLL についての情報を得るためには，第一に，その情報が格納されている領域を特定する必要がある．前述の通り，それらの情報が格納されているのは，インポートセクションである．まず，インポートセクションの位置を特定する手法を述べる．そして，インポートセクション内の必要な API の情報が格納されている位置を探索して抽出する手法を述べる．

4.3.1 インポートセクションの特定

API を特定するためには，インポートセクションの位置を特定する必要がある．まず，PE シグネチャを探索し，PE シグネチャからの相対位置を考慮することで，PE ヘッダ中の IMAGE_DATA_DIRECTORY 構造体のうち，ImportTableAddress の値を取得する．これはインポートセクションの先頭アドレスを指すため，これによってインポートセクションの位置を特定することが出来る．

4.3.2 API 情報の抽出

API 情報を抽出するには，まず前節で特定したインポートセクションの先頭から IMAGE_IMPORT_DESCRIPTOR 構造体の OriginalFirstThunk メンバをたどり，IMAGE_THUNK_DATA 構造体を参照する．そしてさらに，格納されている RVA から，IMAGE_IMPORT_BY_NAME 構造体を参照する．前述の通り，この構造体には API 名が格納されているため，インポートされている API の情報として利用出来る． また，CALL 命令によって呼び出されている API 情報を取得する場合は，CALL アドレスをたどる．CALL アドレスが指すのは，呼び出す API に関する IAT での領域である．再構築された IAT では IMAGE_THUNK_DATA 構造体が格納されているため，そのアドレスを参照すれば，API 名が格納されている IMAGE_IMPORT_BY_NAME 構造体から API 名を得ることが出来る．

4.4 ラベル付与手法

本研究では，第 1.3 節で定義したラベルを利用することで，マルウェアの動作の概要を記述することを 1 つの目的としている．ここでは，ラベルを付与する手法を述べる．まず，ラベルを付与する過程で用いる API-ラベル対応データベースについて，定義を行う．そして，ラベルを付与する手法の詳細を述べる．

4.4.1 API-ラベル対応データベース

ラベルを付与する際に用いる API-ラベル対応データベースについて，定義を行う．第 1.3 節で述べたように，ラベルとはマルウェアの利用している API の動作を抽象化することによって，マルウェアの動作の傾向を示すためのものである．API-ラベル対応データ

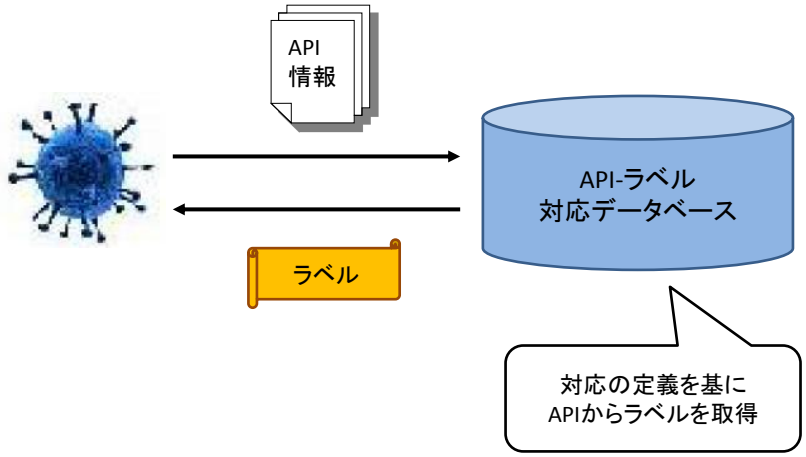


図 4.5: ラベル付与の過程

ベースとは，この抽象化のために用いられるデータ群である．同じ動作傾向を持つ API 群とその動作傾向を代表的に示すラベルの対応を API-ラベル対応データベースに記述しておき，このデータベースを参照することによって API からラベルを作成する．例えば，RegCreateKeyEx や RegOpenKeyEx，RegSaveKey などの API はレジストリを操作するという動作傾向を持つため，レジストリというラベルとの対応をデータベースに記述しておく．

4.4.2 ラベルの付与

前項で定義した API-ラベル対応データベースを用いて，ラベルを付与する手法を述べる．まず，第 4.3.2 節で述べた手法によってマルウェアの利用している API の種類の一覧を取得する．次に，API-ラベル対応データベースを基に，API からラベルを取得し，その種類と数を集計する．一定数以上の数を持つラベルをそのマルウェアの持つ動作の特徴として，マルウェアに付与する．ラベルを付与する過程を図 4.5 に示す．

4.5 サブルーチン解析手法

多くのプログラムにおいては，プログラムコードはサブルーチンによって機能単位で分割されている．これはマルウェアにおいても同様であるため，サブルーチンがどのような

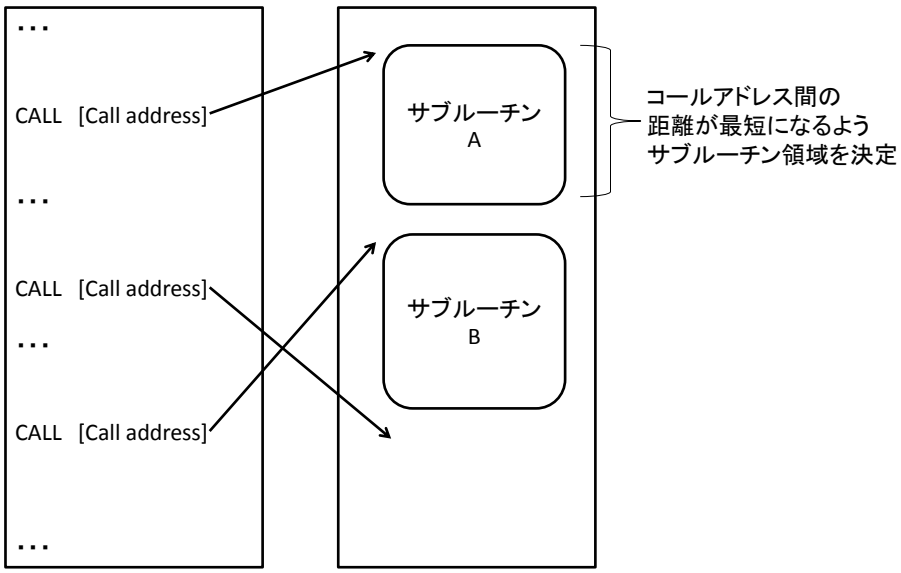


図 4.6: サブルーチン領域の特定手法

機能を持っているかを解析することはマルウェアがどのような動作をするかを知る上で重要であると言える．本節では，サブルーチン内で呼び出されている API を基にして，サブルーチンの動作の概要を取得する．まず，サブルーチン領域を特定する手法を示し，解析の際に用いる動作-実際にサブルーチンの動作を解析する手法について述べる．

4.5.1 サブルーチン領域の特定

サブルーチンの解析を行うためには，サブルーチンのオブジェクトコード上の開始位置と終了位置を知る必要がある．ここでは，サブルーチンの開始位置と終了位置を探索し，サブルーチンの領域を特定する手法を述べる．まず，コード領域をエントリーポイントから逆アセンブルしていく．near CALL の命令が現れたら，その場所からサブルーチンの呼び出し先として参照しているアドレスをサブルーチンの開始位置とする．そして，そのサブルーチンの開始位置から逆アセンブルしていき，最初に RETN 命令が現れた場所をサブルーチンの終了位置とする．これによって，サブルーチンの開始位置と終了位置を知ることで，各サブルーチン領域を特定する．前述のサブルーチン領域を特定する手法を，図 4.6 に示す．

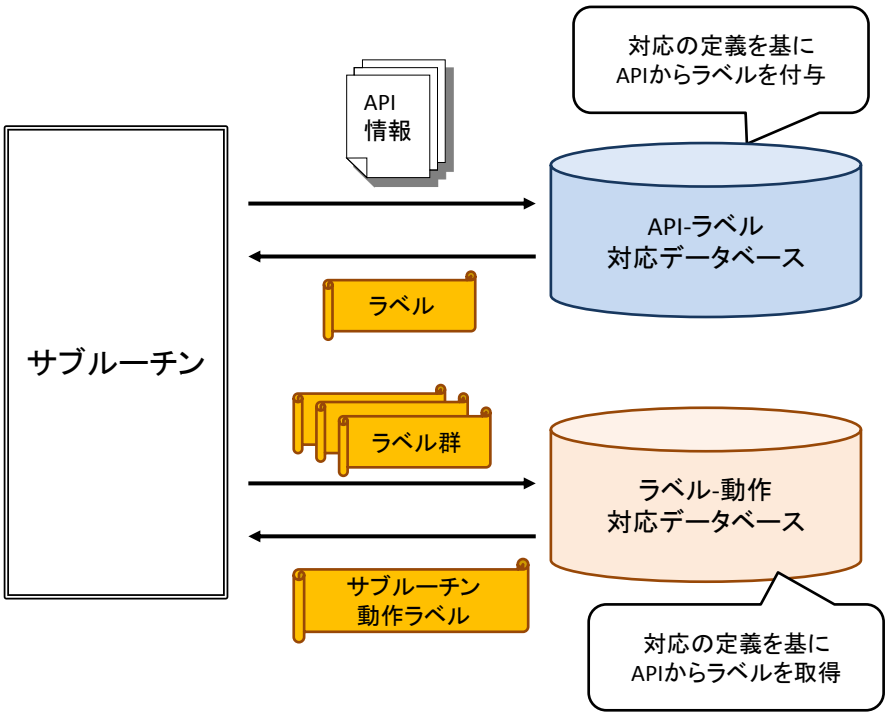


図 4.7: サブルーチンの動作解析の過程

4.5.2 ラベル-動作対応データベース

サブルーチンの動作を解析する際に用いるラベル-動作対応データベースについて，定義を行う．ラベル-動作対応データベースとは，サブルーチン単位で取得されたラベルの組み合わせを基に，サブルーチンの動作を示すサブルーチン動作ラベルを付与するために用いられるデータ群である．サブルーチンに付与されたラベルの種類の組み合わせとその組み合わせを持つサブルーチンが示す傾向のある動作の対応をラベル-動作対応データベースに記述しておき，このデータベースを参照することによって，サブルーチンに付与されたラベルからサブルーチン動作ラベルを作成する．

4.5.3 サブルーチン動作ラベルによるサブルーチンの動作の解析

前項で定義したラベル-動作対応データベースを用いて，ラベルを付与する手法を述べる．まず，サブルーチンから抽出した API に基づいて API-ラベル対応データベースによりラベル付けを行う．次に，ラベル-動作対応データベースを基に，付与されたラベルの種類の組み合わせからサブルーチン動作ラベルを取得し，マルウェアの動作の 1 つとして付与する．サブルーチン動作ラベルを付与し，動作を特定する過程を図 4.7 に示す．

4.6 分類手法

前述の通り，分類することによってマルウェアの所属するファミリーを推定することは，手動で解析を行う前段階での情報として，非常に有用であり，本研究の目的の 1 つである．ここでは，分類を行う手法を述べる．まず，分類に用いる機械学習手法である Support Vector Machine（以下，SVM）について示す．その上で，マルウェアの分類をする上で，どのように SVM を利用するかについて記述する．

4.6.1 Support Vector Machine を用いたマルウェアの分類手法

本研究において，SVM を実際にマルウェアの分類に用いる手法を述べる．まず，分類に用いる分類器を選定する必要がある．分類先となるマルウェアのファミリーは言うまでもなく複数であるため，SVM を利用したマルチクラス分類器を用いる必要がある．また，マルウェアのファミリーの数を考慮すると，1 対 1 方式は計算効率の点から，有効ではないと考えられる．そのため，1 対他方式のマルチクラス分類器（以下，multi-class SVC）を採用する必要がある．本研究では，1 対他方式の multi-class nu-SVC を用いる．分類手法については，以下の通りである．まず，既知のマルウェアを教師信号として学習させることで，分類の準備を行う．準備の第一段階として，学習データとなる検体から API の種類と呼び出し回数の情報を抽出する．次に，分類先となるマルウェアのファミリーに対してクラス ID を割り振る．そして，クラスごとに，そのクラスに分類するか否かを判定するための SVM を用意し，API の情報を特徴ベクトル，クラス ID を分類先として学習させる．これらの学習済みの SVM を，学習量の多いものから並べ，順に判定させていくようにして multi-class nu-SVC を構築する．したがって，SVM の並べられた順にその SVM に対応するクラスに分類されるかどうかを判定し，分類されないと判定された場合は次の SVM に同様にして判定させるというように，順次 1 対他方式の判定を行っていくことでマルチクラスの分類を実現する．最後までどの SVM に対応するクラスにも分類されなかった場合は，その他のファミリーに属するというように判断する．実際に分類する際には，分類対象となる検体から API の情報を抽出して学習済みの分類器に入力することで，分類先のクラス ID を出力として得るという形で行う．

4.6.2 入力データのデータ構造

前項で述べた通り，本研究では API の情報を特徴ベクトルとして用いる．ここでは，マルウェアの利用している API の情報からどのように特徴ベクトルを構成するかを述べる．特徴ベクトルの各成分には，API の種類ごとの呼び出し回数を割り当てる．あるマルウェア検体において，ある種類の API が 5 回呼び出されていたとすると，その API に対応する成分の値が 5 となり，別の種類の API が一度も呼び出されていない場合は，その API に対応する成分の値は 0 となる．したがって，特徴ベクトルの次元は 1 検体内での使用の有無に関わらず，API の種類の総数となる．

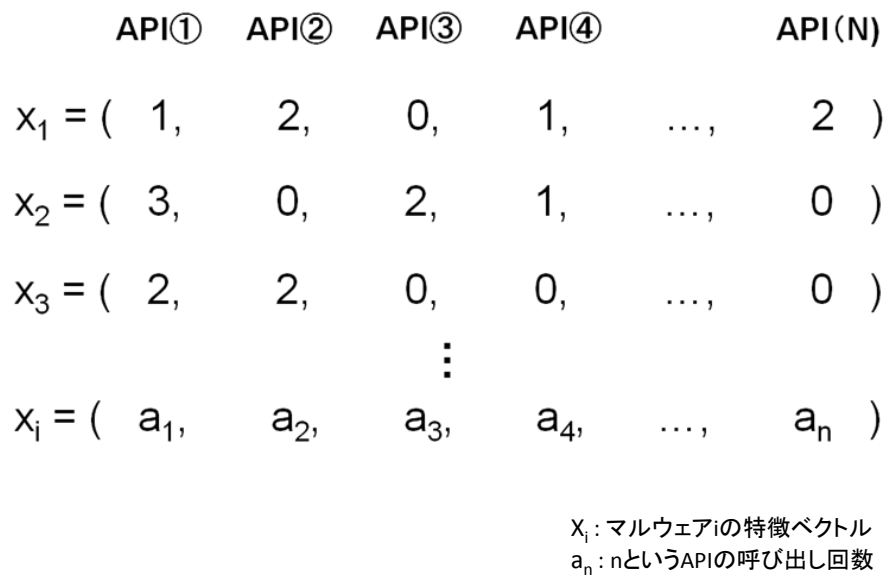


図 4.8: 入力データのデータ構造

4.7 まとめ

本章では，マルウェアを解析・分類について述べる上で前提となるオブジェクトファイルの構造と解析に必要な準備をそれぞれ示し，解析および分類する手法について述べた．まず，API を解析する手法について述べ，サブルーチンの解析とラベルの付与，分類の手法について記述した．第 5 章では，本章で紹介した手法を基にして実装した，マルウェアを実際に分類するシステムについて述べる．

第5章　実装

第4章で述べた手法を用いて，解析からラベルの付与，分類までの一連の流れを自動で行うシステムを実装した．本章では，今回利用した実装環境を示し，そのシステム構成の詳細について述べる．

5.1　実装環境

本システムの実装環境を表5.1に示す．

5.2　実装したシステムの構成

ここでは，実装したシステムのシステム構成について述べる．本システムは解析準備部，情報抽出部，サブルーチン解析部，ラベル付与部，分類部の5つの部分からなる．本システムの設計を図5.1に，概要図を図5.2に示す．また，以下に，各部の詳細について述べる．

5.2.1　解析準備部

解析準備部では，第4.2節で述べた解析を行う上で取り除く必要がある2つの障害を除去し，解析可能な状態にする．この部分は，アンパックおよびIATの再構築のためにデバッガが利用可能である必要がある．そのため，VM上に用意したWindows Vista SP2 (32-bit)上で作業を行う．一連の流れの実装について，以下に示す．まず，アンパックするために，デバッガであるOllyDbg[14]を用いて解析対象となるマルウェアのオブジェクトファイルをメモリ上に展開し，オリジナルエントリーポイント（以下，OEP）と推定される場所まで実行する．OEPの探索は，OEPの直前にPOPAD命令が見られるという特徴を利用した，PUSHAD命令およびPOPAD命令を基準にする方法で行う．そして，OllyDbgのプラグインであるOllyDump[15]を利用してメモリダンプを取得する．次に，IATの再構築を行うために，同じくOllyDbgのプラグインであるImportStudio[16]を実行し，解析準備部の目的である，本来のIATを持ったオブジェクトファイルを得る．これらの一連の流れは，OllyDbgのプラグインであるOllyScript[17]を用いて自動化されている．また，複数のマルウェアのオブジェクトファイルに対してこれらの解析準備を連続的に行えるよう，Pythonのスクリプトによって自動化した．解析準備部の動作の一連の流れを，図5.3に示す．

表 5.1: 実装環境

要素名	属性	利用環境
OS	実機	Ubuntu 10.10 (32-bit)
	VM	Windows Vista SP2 (32-bit)
言語		C, Python, OllyScript
コンパイラ	C	gcc4.6
	Python	Python2.7
	OllyScript	OllyScript0.94
ライブラリ	逆アセンブルエンジン	HDE32(HackerDisassemblerEngine-32)
	RDBMS	SQLite3
	学習器	libSVM
デバッガ	OllyDbg	
デバッガプラグイン	Dump プラグイン	OllyDump
	スクリプトプラグイン	OllyScript
	IAT 再構築プラグイン	ImportStudio

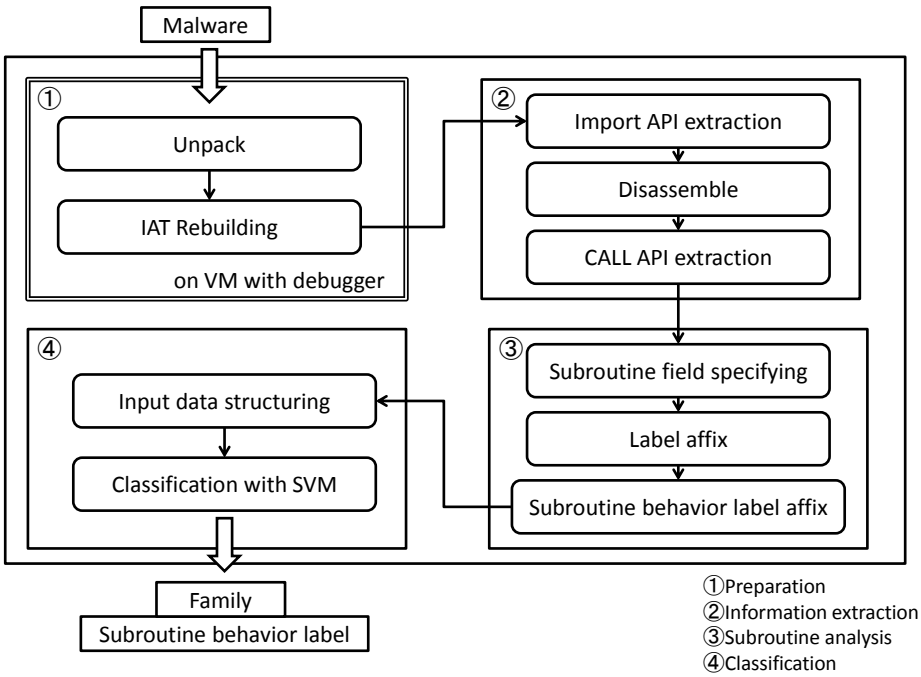


図 5.1: システムの設計

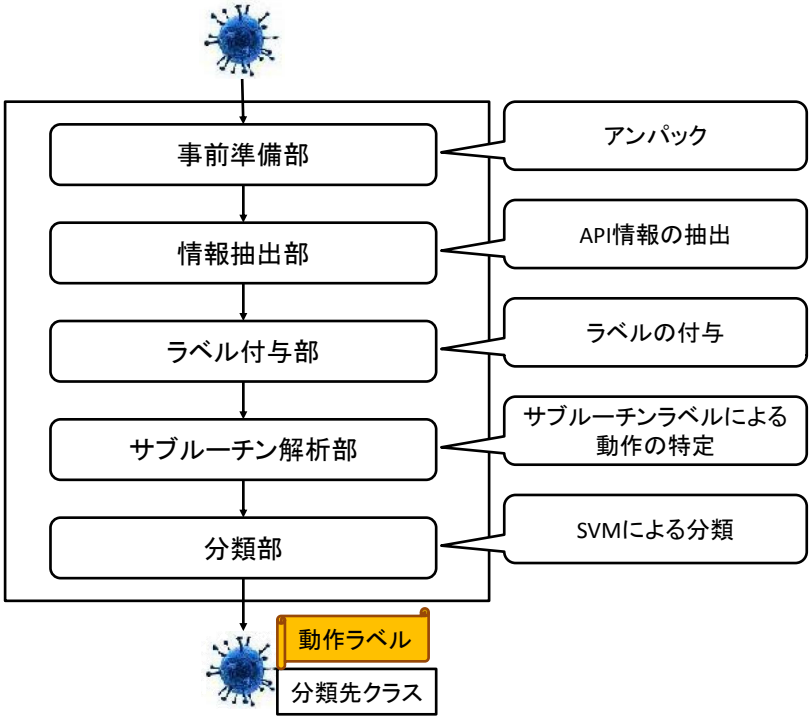


図 5.2: システム概要図

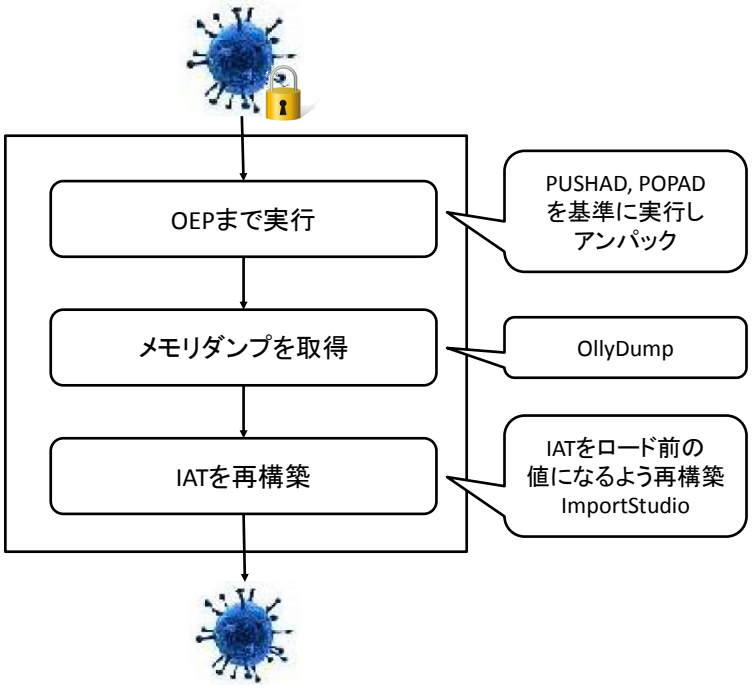


図 5.3: 解析準備部の概要図

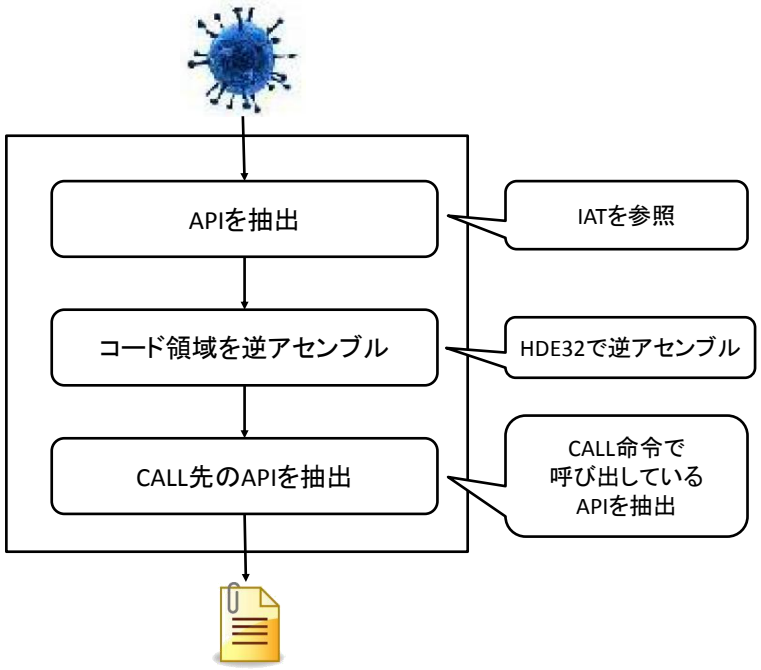


図 5.4: 情報抽出部の概要図

5.2.2 情報抽出部

情報抽出部では，マルウェアの API についての情報を解析し，抽出する．C 言語によって実装したプログラムによってファイルを読み込み，4.3 で述べた通りに参照していき，最終的に IAT を参照することで，インポートされている各 API の情報を抽出する．また，CALL されている各 API についての情報も同様に抽出する．情報抽出部の動作の一連の流れを，図 5.4 に示す．

5.2.3 ラベル付与部

ラベル付与部では，4.4 で述べた手法に基づいてマルウェアに対してラベルを付与する．API-ラベル対応データベースの作成は，MSDN ライブラリ [18] の Web サイトの関数リファレンスに記述された機能ごとのカテゴライズを利用した．ラベル付与部の動作の一連の流れを，図 5.5 に示す．

5.2.4 サブルーチン解析部

サブルーチン解析部では，第 4.5 章で述べた手法に基づいてサブルーチンに動作ラベルを付与することで，サブルーチンの動作を明らかにする．ラベル-動作対応データベースの作成は，既知検体に対するサブルーチンごとのラベル付与結果とウイルス情報検索

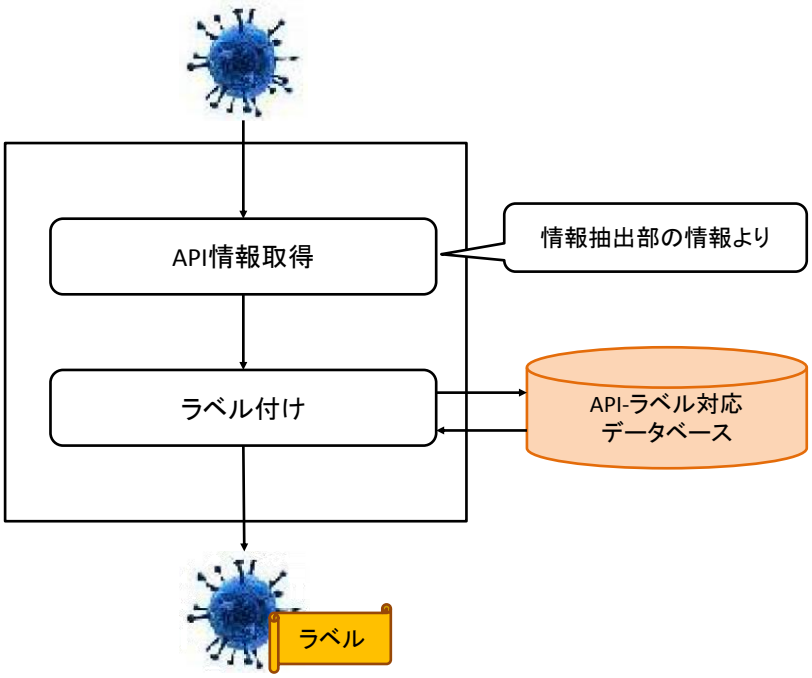


図 5.5: ラベル付与部の概要図

iPedia[19] に掲載されている動作情報の対応から行った．サブルーチン解析部の動作の一連の流れを，図 5.6 に示す．

5.2.5 分類部

分類部では，第 4.6 節で述べた手法に基づいて SVM によりマルウェアの分類を行う．SVM の実装については，C および C++による SVM のライブラリである，libSVM を利用して行った．あらかじめ，学習用の検体より第 5.2.2 項で抽出した API 情報から第 4.6.2 項で示した入力データのデータ構造で入力データを構成し，学習モデルのビルドを行うようにする．そして，分類対象の検体に対しては，抽出した API 情報を基に同様に入力データを構成パターン認識を行う．

5.3 まとめ

本章では，第 4 章で述べた手法に基づいて実装したシステムの構成について述べた．また，システムを構成する事前準備部，情報抽出部，ラベル付与部，サブルーチン解析部，分類部の 5 つの部分について，詳細を明らかにした．

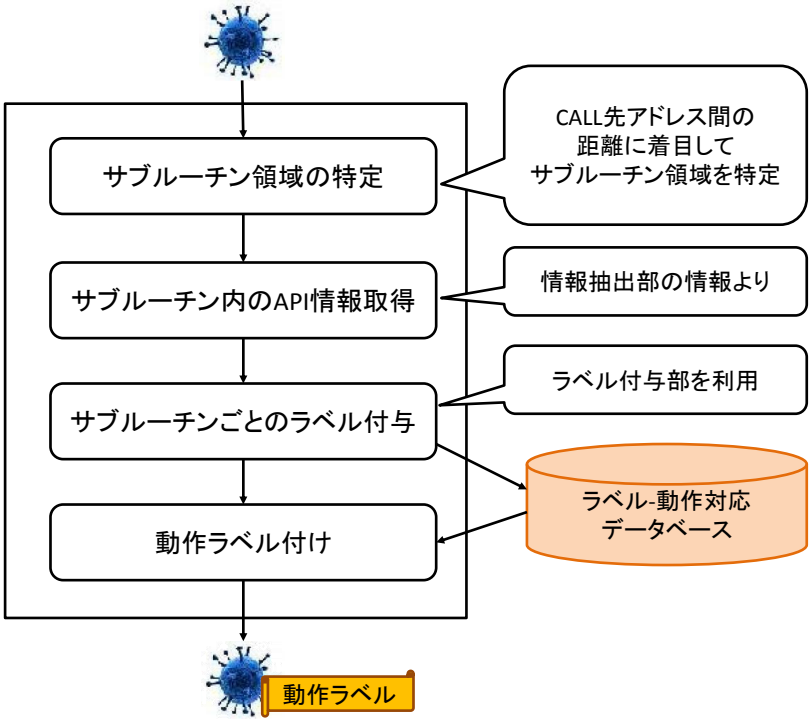


図 5.6: サブルーチン解析部の概要図

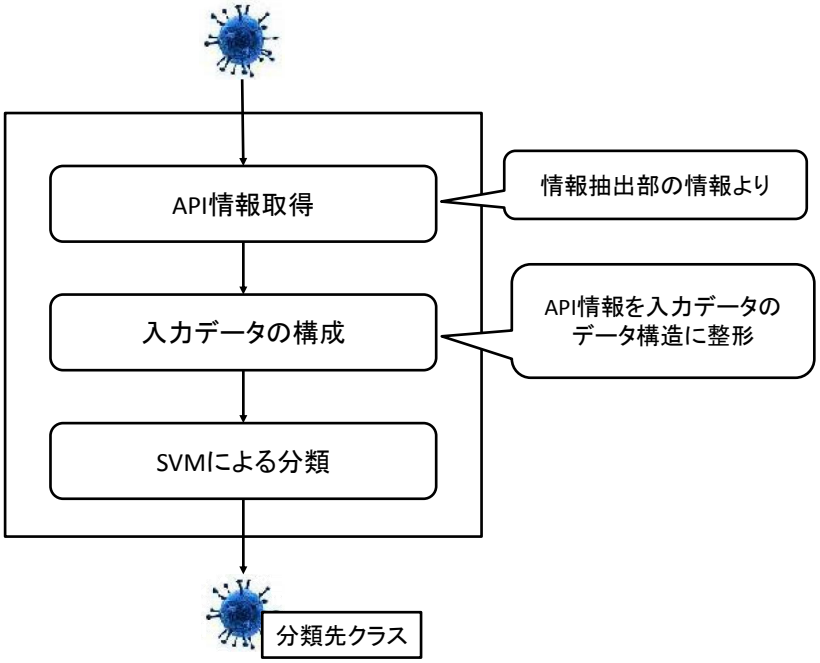


図 5.7: 分類部の概要図

第6章　実験と評価

本章では，提案手法を用いて実装したシステムの有意性を検証する実験とその結果について述べる．また，得られた結果を評価し，議論を展開する．

6.1　実験

ここでは，本システムを用いて行った2種類の実験について述べる．1つめは分類によるファミリー特定実験である．この実験では，本システムを用いて実際に分類を行い，ファミリーを正しく特定出来るかを検証した．2つめはラベルによるマルウェアの動作特定実験である．この実験では，本システムを用いてラベルを付与し，それによってマルウェアの動作を特定することが出来るかを検証した．それぞれの実験についての詳細を，以下に示す．

6.1.1　分類によるファミリー特定実験

本システムを用いて実際にマルウェア検体を分類し，分類結果により正しいファミリーを特定出来るかを検証する実験を行った．この実験の概要と実験環境を提示し，その結果を記述する．

実験の概要

まず，分類対象となる検体をVirusTotal[20]を用いたサーフィス解析することで，あらかじめファミリーを特定した．これは，システムにより得られた結果の正しさを確認するためのものである．ファミリーは独立行政法人情報処理推進機構（以下，IPA）がコンピュータウイルス届出状況の報告などに用いるものを利用した．そして，本システムを用いてそれらの検体を実際に分類し，その結果とあらかじめ特定しておいたファミリーとを比較し，正しいファミリーに分類されている検体数とその全分類対象検体における割合を取得した．それらを本実験の実験結果として提示する．

実験環境

分類実験の実験環境について述べる．表6.1に実験環境をまとめて示す．

表 6.1: 分類実験の実験環境

要素名	利用環境	備考
OS	Ubuntu 10.10	実機 (32-bit)
	Windows Vista SP2	VM (32-bit)
学習対象	2000 検体	研究室の保有するマルウェア収集サーバより無作為抽出
分類対象	1000 検体	研究室の保有するマルウェア収集サーバより上記の学習対象と重複のないよう無作為抽出

表 6.2: 各ファミリーに分類された検体数

検体名 (IPA)	分類された検体数	ファミリー一致数	一致率
W32/Virut	532	485	91.1%
W32/IRCbot	183	109	59.5%
W32/Autorun	52	24	46.1 %
W32/Mydoom	49	33	67.3 %
W32/Downad	45	19	42.2 %
W32/Sasser	42	11	26.1 %
W32/Waledac	7	1	14.4 %
W32/Sality	5	0	0 %
その他	85		
全体	1000	682	68.2 %

実験結果

前項の実験環境にてマルウェアの分類およびファミリーの特定を行った結果を述べる．各ファミリーに分類された検体数を表 6.2 に示す．分類対象である 1000 検体のうち，正しく分類出来た検体数は 682 検体であった．これにより，今回の分類実験においてマルウェアを正しく分類出来た割合は 68.2 %であることが分かる．この結果についての評価と議論は第 6.2 節に後述する．

6.1.2 ラベルによる動作特定実験

本システムを用いてマルウェア検体に対して動作ラベルを付与し，それによってマルウェアの動作の概要を知ることが出来るかを検証する実験を行った．この実験の概要と実験環境を提示し，結果を記述する．

表 6.3: ラベルによる動作確認実験の実験環境

要素名	利用環境	備考
OS	Ubuntu 10.10	実機 (32-bit)
	Windows Vista SP2	VM (32-bit)
実験対象	10 検体	研究室の保有するマルウェア収集サーバより特徴的な動作を持つ検体を抽出
	W32/Antinny	
	W32/Autorun	
	W32/Bugbear	
	W32/Fujacks	
	W32/Mimail	
	W32/MSBlaster	
	W32/Mydoom	
	W32/Sasser	
	W32/Virut	
	W32/Zotob	

実験の概要

まず，実験対象となる検体の選定を行った．マルウェア解析を行う Web サービスである VirusTotal[20] によって検知名を取得し，McAfee ウイルス検索 [21] およびウイルス情報 iPedia[19] を用いて機能を確認していくことで，特徴的な機能を持つマルウェアを実験対象として選定した．これらの実験対象の検体に対して，本システムを用いてラベルを付与し，そのラベルを基に検体の動作を確認出来るかどうかを確認した．具体的には，各々の検体について，既知の動作に対して正しく特定することが出来た動作の数と全動作数のうち特定することが出来た動作数の割合を取得した．さらに，全検体のうち，一定以上の動作を特定することが出来た検体の割合を算出した．それらを本実験の実験結果として提示する．

実験環境

ラベルによる動作確認実験の実験環境について述べる．表 6.3 に実験環境をまとめて示す．

実験結果

前項の実験環境にてラベルによるマルウェアの動作特定実験を行った結果を述べる．実験対象とした 10 検体について，全動作数と特定出来た動作数，特定率を表 6.4 に示す．対

表 6.4: ラベルによる特定出来た動作数

検体名 (IPA)	全動作数	特定出来た動作数	特定率
W32/Antinny	10	10	100 %
W32/Autorun	10	9	90 %
W32/Bugbear	8	8	100 %
W32/Fujacks	4	4	100 %
W32/Mimail	8	7	87.5 %
W32/MSBlaster	7	7	100 %
W32/Nimda	4	4	100 %
W32/Sasser	6	5	83.3 %
W32/Virut	13	11	100 %
W32/Zotob	5	5	100 %
全体	75	70	93.3 %

象とした 10 検体のうち，ラベルによって全ての動作を特定することが出来た検体の割合は 60 %であった．また，全検体の持つ動作のうち特定出来たものの割合は 93.3 %であった．これらの結果についての評価と議論は第 6.2 節に後述する．

6.2 評価

上記の 2 つの実験の結果について評価を行い，議論を展開する．

6.2.1 分類によるファミリーの特定

第 6.1.1 節で行った分類実験の結果を通して，本システムによるマルウェアの分類の正当性と，分類によるファミリーの特定の有効性について述べる．

本システムによって正しく分類を出来た割合は，第 6.1.1 項で述べた通り，68.2 %であった．限られた中で，一定の割合を確保して分類し，ファミリーを特定出来ることを示した．分類結果から，検体数が多く，十分に学習出来ている分類先のファミリーに属する検体については，高い精度で分類出来ることが確認された．一方，学習データの量が十分でないファミリーについては，十分とは言えない結果であった．これには，2 つの要因が考えられる．1 つ目は，純粋に学習量が少ないために効果的な分類が行えなかったことが挙げられる．2 つ目は，学習量が多いものから判定を行っていくように学習器を配置したため，先に分類したものの誤認識が後の分類に影響を与えていると考えられることが挙げられる．以下に，想定している例を挙げる．今回の実験で最初に配置した SVM では，Win32/Virut であるかどうかの判定を行っている．この最初の分類において，本来は後半

に配置されている Win32/Sasser に分類されるべき検体を Win32/Virut に分類してしまうと，Win32/Sasser かどうかの判定を行う機会が減少してしまう．このような場合が，結果に影響を与えていると想定される．

また，この結果はあくまで第 6.1.1 項で述べた実験環境で行った場合のものであり，この実験環境の持つ特徴が分類精度を低下させる要因となり結果に影響を与えていることも考えられる．1 つ目は研究室の保有するマルウェア検体の偏りである．研究室の保有するマルウェア収集サーバは受動型・能動型ハニーポッドを含む複数の手法によって収集を行っている．しかし，例えばメールによる感染が主流の検体は，これらの収集環境のみでは効率的に収集することが難しい．このように，感染経路に対する収集手法の適正の有無によって，マルウェア検体のファミリーごとの検体数に偏りが生じていることが考えられる．事実，研究室の持つ全マルウェア検体のうち，メールによる感染活動を行うマルウェアの一種である W32/Netsky に属するのは 3 検体のみであった．本システムでは，機械学習によって分類を行っているため，高い精度で分類を行うためには，十分な学習が必要不可欠である．したがって，無作為抽出の結果，学習データ群に含まれず分類対象群に含まれているような検体があれば，それは分類精度の低下に繋がる．マルウェア検体のファミリーごとの検体数に偏りがある環境下では，分類精度も低下すると考えられるのである．すなわち，事前に十分に学習させる用意が出来る場合，今回の実験と比較して，さらに高い精度で分類を行うことも可能であると言える．また，2 つ目は全検体の数である．機械学習では一般に，学習データを増やし十分に学習させた学習器を用いることで精度を向上させることが出来る．したがって，学習データ群となる検体が十分に確保されている環境では，現状よりも優れた精度で分類することが出来ると言える．以上のような事柄を踏まえると，環境を整えて運用することで，本システムは十分に正当な分類を行うことが出来ると期待される．

また，分類の結果によってマルウェア検体の属するファミリーを特定するという試みの有効性について以下に述べる．本システムは，ファミリーごとの分類を行うため，分類の正しさがファミリーの特定の正しさと等しい．したがって，分類実験の結果が 68.2 %であることから，ファミリーを正しく特定出来る割合も 68.2 %であり，かつ，前述の通りの向上が見込まれる．ファミリーの特定の正確性を 68.2 %とすると，解析者は未知のマルウェアを解析する際に，およそ 7 割の検体について事前に情報を得ることが出来ることになる．これらのマルウェアについては解析をする際に特に留意するべき点を知っている状態で解析を進めることが出来るため，解析を効率的に進めることが出来ると考えられる．したがって，本システムによってファミリーを特定する有効性はあると言える．

6.2.2 ラベルによる動作特定可能性

第 6.1.2 項で行ったラベル付与による動作特定実験の結果を通して，本システムによるマルウェアの動作の特定可能性とその有効性について述べる．

本システムによって特定出来たマルウェアの動作は，第 6.1.2 項で述べた通り，6 検体についてはマルウェア検体の持つ全ての動作であり，他の 4 検体についてはマルウェア検体の持つ全ての動作のうち 1 つあるいは 2 つの動作を除いた全てのものであった．全体とし

て，全検体の持つ 75 の動作のうち，70 の動作を特定することが出来た．全体の特定率は 93.3%であった．この数値から，本システムによる動作特定可能性が確認出来た．今回挙げた 10 検体について，ファイルやレジストリの作成，削除，改変などといった操作は全て特定することが出来た．これらはマルウェアの動作の基本とも言えるものであり，多くのマルウェアが行うものである．その他の動作として，ファイルの検索やミューテックスオブジェクトの作成，ネットワーク通信，プロセスの操作などは特定することが出来た．一方で，バックドアの作成やファイルサーバとして振る舞うようになるといった動作については，特定することが出来なかった．これらについては，動作-ラベル対応データベースに適切に対応付けを定義することが出来なかったことが理由として挙げられる．したがって，これらの動作についてデータベースに定義することで，検知出来るようになる可能性が考えられる．このことより，実際に環境を整備して運用する際には，高い精度での動作特定が期待出来る．

また，動作を特定することの有効性について以下に述べる．これによって，解析者は未知のマルウェアを解析する際に，動作の概要についての情報を事前に得ることが出来ることになる．これらのマルウェアについては，動作の詳細に留意して解析を進めれば良いため，解析を効率的に進めることが出来る．例えば，未知のマルウェアの解析の際に，本システムによってレジストリの操作を行うという情報が事前に提供されている場合，解析者はどのレジストリキーが操作されているかを追跡しさえすれば良いことになる．これによって，事前情報がない場合と比較して，迅速に解析を進めることが出来る．したがって，本システムによって動作を特定する有効性はあると考えられる．

6.3 まとめ

本章では，まず，本論文で提案した手法に基づいて実装したマルウェアの解析・分類システムを用いた実験について述べ，その結果を示した．そして，それらの結果を示し，システムの有効性について議論を展開した．実験結果からファミリーの特定および動作の特定の両者について一定の正当性が示された．それを基に本システムを用いて解析に際して事前に情報を得ることが解析に与える影響を確認し，本システムの有効性を述べた．

第7章　考察

本章では，第 6 章の実験と評価を受けて，提案した手法についての考察を進める．そして，本研究手法をどのように利用することが想定されるかについて議論する．さらに，改善可能な点や解決すべき課題を挙げ，解決のための方法や方向性について議論することで，本研究をさらに発展させることを目指す．

7.1　提案手法の考察

本研究で提案した 2 つの手法について，包括的な考察を与える．

7.1.1　分類によるファミリーの特定についての考察

第 6.1.1 項での実験より，提案した手法によってマルウェアを分類することで，一定の精度で未知のマルウェアの属するファミリーを特定出来ることが分かる．また，本手法は事前に十分な学習が必要となるために既知のマルウェアが用意出来る運用環境を整えなければならない一方，十分な検体数を確保できているマルウェアについては 90%を超える精度でファミリーを特定出来ており，一定の成果が期待出来る．また，分類に API を利用するという特徴から，その結果にはマルウェアの動作が強く影響を与えている．したがって，ファミリーを特定した結果から，マルウェアの動作についての類似関係を知ることが出来る．

7.1.2　ラベルによる動作特定についての考察

ラベルという API の情報を抽象化した概念を用いた手法で，マルウェアの動作の概要を提示出来ることが第 6.1.2 項での実験によって確認出来た．現在，マルウェアの動作の情報を静的解析によって取得する既存の研究としては，第 3 章で述べたように，マルウェアを分類した結果から推定するものや，マルウェア間の類似度を基にした関係性から推定するものがある．しかし，これらの研究で用いられている手法では，ごく少ない新しい機能が追加されている亜種や，どのマルウェアとも類似していない新種に対しては有効でない可能性がある．動作の推定結果がマルウェアの類似度にのみ依存しているためである．これに対し，本手法では API の情報を基にして動作を特定しているため，前述のような状況においても，問題なく動作を特定し情報として提供することが期待出来る．このよう

な利点があることから，既存の研究とは異なる特性を持つ新しい手法を提案することが出来たと言える．

7.2 想定される利用例

本研究の成果がどのような状況での利用が想定されるかについて述べる．本研究の手法によって得られる情報は，主として，解析者，対策立案者，研究者の三者にとって有用であると考えられる．

解析者は，新しく出現したマルウェアを迅速に解析する必要がある．しかし，静的解析手法による解析は第 2.5 節で述べた通り，事前に情報がなくオブジェクトコードを詳細に渡って解析する必要がある場合，非常に時間がかかる．また，動的解析手法による解析においても，監視するべき動作が特定出来ていない状態で解析する場合は，レジストリやファイル，ネットワーク通信などを始めとする，多くの情報を監視して判断する必要がある．これに対して，本研究で提案した手法によるシステムを活用することで，事前にマルウェアのファミリーと動作の概要を知ることが出来る．これによって，静的解析手法による解析においては，特に気をつけてして解析するべき部分を限定することが出来る．また，動的解析手法による解析においては，監視するべき事項を絞ることが出来る．したがって，本研究の成果をこのように利用することで，解析者は解析作業を効率化することが出来る．

対策立案者は，マルウェアの脅威に対して効果的な対策をとる必要がある．そのためには，マルウェアの動作を把握していることが必須であると言える．未知のマルウェアの脅威が発生した際に，対策立案者は本研究の提案手法によるシステムを通して，その動作を知り，対策を考案する際に活かすことが出来る．また，マルウェアのファミリーについての情報も得られるため，同じファミリーに属する既知の亜種に対する対策を水平的に利用出来るかどうかを検討する材料とすることが出来る．このようにして，対策立案者は本研究の成果を用いることで，有効な対策を効率的に考案することが出来る．

研究者は，研究対象としてマルウェアを利用する機会が多いと考えられる．実験などの際に，研究内容に合わせたマルウェア検体を選定することは，研究を的確に進める上で重要である．本研究で提案した手法を用いることで，研究者は実験に必要な動作の要件を満たしたマルウェアや，適切なファミリーのマルウェアを選択することが出来る．したがって，本研究の成果によって，研究者が実験環境を準備する負担を減少させることや，マルウェアを解析する技術を持たない隣接分野の研究者であっても適切なマルウェアを選択出来るようにすることが期待出来る．

7.3 解決すべき課題

本研究で提案した 2 つの手法について，解決すべき課題を挙げ，手法の洗練を目指す．

7.3.1 分類によるファミリーの特定についての課題

マルウェアの分類によるファミリー特定手法の課題を挙げる．最も注力するべき課題は，分類の精度をより向上させることを考慮する必要がある．本手法では，分類の精度が特定したファミリーの正しさに直接的な影響を与える．正しいファミリーを情報として提供することが，解析や対策の立案の円滑化に繋がるため，分類精度を向上させることは，本手法の有効性の向上と相関性があると言える．

7.3.2 ラベルによる動作特定についての課題

ラベルによる動作特定手法の課題を挙げる．第一に，特定出来る動作の種類についての課題がある．第 6.1.2 項での実験では，ファイルやレジストリ，プロセスの操作を始めとする，マルウェアの基本的な動作を特定することが出来た．しかし，特定出来る動作の種類には，ラベル-動作対応データベースに定義されているものという限界が存在する．これは，手法の特性上避けることが出来ない問題であるが，事前にラベル-動作対応データベースを整備しておくことで，さらに精度を向上させることは可能であると考えられる．次に，特定出来る動作の粒度についての課題がある．ラベルで特定出来るのは動作の概要のみであり，動作の対象など，詳細な情報を得ることは出来ない．例えば，マルウェアの中にはウイルス対策ソフトウェアのプロセスを停止させるものなどがあるが，プロセスを停止させることは特定することが出来ても，どのようなプロセスを停止させるようにプログラムされているのかは，本手法のみでは知ることが出来ない．したがって，現在よりも多くの情報を得るためには，動作対象などを特定する手法を構築し，併用することが必要であると考えられる．

7.4 まとめ

本章では，本論文で提案した手法について考察を進めるとともに，今後解決すべき課題を展開した．運用にあたっては，これらを考慮することで本論文で行った実験での結果よりも優れた成果を挙げることが出来ると期待される．また，課題を基にして手法の改善を目指し研究を進めることで，より効果的な解析・分類手法を構築することが出来る．

第8章　結論

本章では，本論文の全体についてをまとめ，本論文によって達成されたことについて述べる．そして，本論文の発展を実現するために今後の展望を述べる．

8.1　まとめ

本論文の目的は，増加や複雑化を続けるマルウェアに対して，より効果的な対策を効率的にとっていくことを可能とするために，マルウェアについての事前情報を提供することである．さらに，その実現のためにマルウェアを静的解析手法を用いてソフトウェアによって自動で解析することで，動作の概要を特定するとともに，分類することによってマルウェアのファミリーを判別し，亜種や同種などといったマルウェア間の関係性を明らかにすることでマルウェアの情報を得ることが出来るようにすることであると言える．

そこで，本論文では，上記のような目的を達成するために2つの手法を提示した．1つ目は機械学習手法であるSVMを用いてマルウェアのAPI呼び出しを基に分類することで，マルウェアのファミリーを特定する手法であり，2つ目はラベルを用いてマルウェアの動作を特定する手法である．マルウェアの分類することによってファミリーを特定する手法では，まず，既知のマルウェアをファミリーごとにまとめたクラスを作成する．それに対して静的解析手法によって抽出したAPI呼び出しの情報を基に，機械学習であるSVMでマルウェアを分類する．ラベルを用いて動作を特定する手法では，まず，ラベルと呼び出しているAPIを対応付けしたデータベースおよびラベルの組合せと動作を対応付けしたデータベースを作成しておく．これらのデータベースを参照することで，静的解析手法によって抽出したAPIを基にラベルを付与していくとともにラベルの組合せから動作を推定する．これらの2つの手法を提案した．

これらの手法を基にしてマルウェアを解析・分類し，動作の特定およびファミリーの特定を行うシステムを実装した．そして，これを用いて研究室の保有するマルウェア検体を対象とした解析および分類の実験を行い，各々の手法の有効性を検証した．分類によってマルウェアのファミリーを正しく特定することが出来た割合は68.2%であった．また，ラベルを用いてマルウェアの動作を特定出来た割合は93.3%であった．これらの結果から，本システムを学習データの量などを考慮し適切に運用することで，効果的にファミリーの特定および動作を特定し得ることが示された．そして，本システムによる実験による経験を基に，2つの手法に対して考察を与えるとともに，現状で考えられる解決すべき課題を挙げ，今後の発展のための道を示した．

本論文によって，対策の立案や解析に先立って，マルウェアの属するファミリーや動作

の概要を事前情報として提供することが可能となった．これにより，解析者や対策立案者は，提供された事前情報に基づいて解析および対策の立案を進めることが出来るため，効果的かつ効率的に作業を進めることが出来るようになる可能性を提示した．

8.2 今後の展望

本論文全体のまとめを受けて，今後の展望を述べる．

提案した手法についての今後の展開としては，2 つの発展が考えられる．1 つ目は，手法を洗練することによる，分類および解析の精度の向上である．2 つ目は，他の解析手法との連携による自動解析の効率化である．前述の通り，提案手法のみでシステムを構築した場合，動作の概要を把握することは出来ても，動作の対象となるレジストリキーや通信先など，詳細な情報を取得することは現状では出来ない．これについて，API 呼び出しの際の引数を取得出来るネイティブ API フックを利用した解析手法との連携や，本システムによって監視項目を絞った上での動的解析手法による解析など，他の解析手法と組み合わせた手法によって，より詳細な情報を効率的に得られることが考えられる．上述の 2 つの方向性について，前者は提供する情報の信頼性の向上へ繋がり，後者は提供する情報量の増加に繋がると期待出来る．

さらに，提案手法を用いたシステムについての今後の展開として，その運用体制について示す．本システムは，学習データとなる検体数やラベルに関する対応付けを定義したデータベースなどの情報量を増やすことによって，より精度の高い分類を行うことが出来る．そこで，大学を始めとするマルウェアを収集して研究や解析を行っている研究機関に運用してもらい，情報を集約して共有することを提案する．これによって，本システムによる解析および分類の精度を高めるとともに，マルウェアについての情報を必要とする機関に広く提供するという形で本システムを活用することが期待出来る．

謝辞

本論文の作成にあたり，ご指導頂いた慶應義塾大学環境情報学部学部長 村井 純博士，同学部教授 徳田 英幸博士，同学部教授 中村 修博士，同学部准教授 楠本 博之博士，同学部准教授 高汐 一紀博士，同学部准教授 三次 仁博士，同学部准教授 植原 啓介博士，同学部専任講師 中澤 仁博士，同学部准教授 Rodney D.Van Meter III 博士，同学部教授武田 圭史博士，同大学 DMC 機構専任講師 斉藤 賢爾博士，同大学政策・メディア研究科特別研究講師 佐藤 雅明博士に感謝致します．特に武田圭史博士は，研究で行き詰まる私に対して，時には厳しく，時には丁寧に，非常に根気強く指導していただきました．常に新しいアイディアと研究手法で私を導いていただき，何度も私に新しい視点や手本を見せて下さりました．本論文の作成も，氏の粘り強いご指導なくしては成し得ないものでした．本当にありがとうございました．

本研究を進める上で，最初期からアドバイスを頂いた村井研卒業生の水谷 正慶博士に感謝致します．本研究を始めるきっかけとなったのも氏であり，本研究の礎は，氏の適切な助言によって成り立っています．本当に感謝いたします．

研究室生活を通してご指導頂いた慶應義塾大学大学院政策・メディア研究科修士課程 上原 雄貴氏，重松 邦彦氏，村井研究室卒業生の福岡 英哲氏，梅田 昂翔氏に感謝致します．特に，上原 雄貴氏，重松 邦彦氏に感謝いたします．両氏には，研究のみならず，研究室生活に伴う様々なことを教えていただきました．今まで私が研究を続けることが出来たのは，両氏のご指導ご鞭撻の賜物であることに疑いはありません．本当に感謝致します．

研究室で苦楽を共にした相見 眞男氏，Vu Xuan Duong 氏，関根 冬輝氏，Pham Van Hung 氏，山本 知典氏，吉原 洋樹氏に感謝致します．彼らと一緒に研究をすることでお互いを刺激しあい，より質の高い議論や研究をすることができました．特に山本 知典氏は研究室に所属する以前からの友人であり，同じ学問を志す同級生として，時には目標として，時にはお互いに切磋琢磨する相手として，研究室の内外を問わない付き合いをしてくれました．本研究室へと誘う形で私が所属を決めるきっかけとなったのも氏であり，彼なくしては私の研究生活そのものが送れませんでした．本当に感謝致します．

研究室で共に生活を送った有馬 怜文氏，大矢 崇央氏，鴻野 弘明氏，中島 明日香氏，由井 卓哉氏，吉原 大道氏，露木 航平氏，中安 恒樹氏，吉田 裕氏に感謝致します．彼らと過ごすことが出来たお陰で，たとえ苦しい時でも，研究室で遅くまで活動を続けることが出来ました．本当に感謝致します．

最後に，大学入学からの 4 年間だけでなく 22 年間をあらゆる面で支えていただいた父，碓井 利一郎，母，碓井 和枝と私の家族に心から感謝致します．

参考文献

[1] ラルフ・ベンツミュラー&サブリナ・バーケンコフ. G data マルウェアレポート 2010 年下半期. Technical report, G Data セキュリティラボ, Bochum, Aug 2011. http://www.gdata.co.jp/files/2011H1MalRep.pdf.

[2] McAfee Labs. Mcafee 脅威レポート 2011 年第 2 四半期. Technical report, McAfee Labs, Santa Clara, Aug 2011. http://www.mcafee.com/japan/media/mcafeeb2b/international/japan/pdf/threatreport/threatreport11q2.pdf.

[3] McAfee Labs. Mcafee 脅威レポート 2011 年第 3 四半期. Technical report, McAfee Labs, Santa Clara, Aug 2011. http://www.mcafee.com/japan/media/mcafeeb2b/international/japan/pdf/threatreport/threatreport11q3.pdf.

[4] ラルフ・ベンツミュラー&サブリナ・バーケンコフ. G data マルウェアレポート 2010 年下半期. Technical report, G Data セキュリティラボ, Bochum, Feb 2010. http://www.gdata.co.jp/files/2010lasthalfMalwareReport.pdf.

[5] McAfee Labs. Mcafee 脅威レポート 2010 年第 4 四半期. Technical report, McAfee Labs, Santa Clara, Feb 2010. http://www.mcafee.com/japan/media/mcafeeb2b/international/japan/pdf/threatreport/threatreport10q4.pdf.

[6] Kaspersky Lab. Kaspersky マンスリーレポート. Technical report, McAfee Labs, Santa Clara, Aug 2011. http://www.mcafee.com/japan/media/mcafeeb2b/international/japan/pdf/threatreport/threatreport11q2.pdf.

[7] 岩村誠, 伊藤光恭, and 村岡洋一. 機械語命令列の類似性に基づく自動マルウェア分類システム. **情報処理学会論文誌**, 51(9):1622–1632, Sep 2010.

[8] 安本幸希, 森井昌克, and 中尾康二. マルウェアコードの類似度判定にる機能推定. **電子情報通信学会技術研究報告**. *ISEC,* **情報セキュリティ**, 107(345):31–36, Nov 2007.

[9] 勝手壮馬, 安本幸希, 伊沢亮一, 森井昌克, and 中尾康二. マルウェアの分類方法とその応用に関する考察. **電子情報通信学会技術研究報告**. *ICSS,* **情報通信システムセキュリティ***: IEICE technical report*, 109(86):103–108, Jun 2009.

[10] 岩本一樹 and 和崎克己. コンピュータウイルスのコード静的解析による特徴抽出と分類について. **電子情報通信学会技術研究報告**. *ISEC,* **情報セキュリティ**, 107(397):107–113, Dec 2007.

[11] 戸部和洋, 森達哉, 千葉大紀, 下田晃弘, and 後藤滋樹. 実行ファイルに含まれる文字列の学習に基づくマルウェア検出方法. In **マルウェア対策人材育成ワークショップ（*MWS2010*）論文集**, number 2, pages 777–782, Oct 2010.

[12] Clemens Kolbitsch, Paolo Milani Comparetti, Christopher Kruegel, Engin Kirda, Xiaoyong Zhou, and XiaoFeng Wang. Effective and efficient malware detection at the end host. In *SSYM'09 Proceedings of the 18th conference on USENIX security symposium*, pages 351–356, Aug 2009.

[13] Ashkan Sami, Babak Yadegari, Hossein Rahimi, Naser Peiravian, Sattar Hashemi, and Ali Hamze. Malware detection based on mining api calls. In *SAC '10 Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1020–1025, Mar 2010.

[14] Oleh Yuschuk. Ollydbg. `http://www.ollydbg.de/`.

[15] Gigapede. Ollydump. `http://www.openrce.org/downloads/details/108/OllyDump`.

[16] tpericin. Importstudio. `http://www.reversinglabs.com/download/ImportStudio_1.0.1.rar`.

[17] SHaG. Ollyscript. `http://www.openrce.org/downloads/details/106/OllyScript`.

[18] Microsoft. Msdn ライブラリ windows 開発. `http://msdn.microsoft.com/ja-jp/library/gg250485(v=vs.85).aspx`.

[19] 独立行政法人情報処理推進機構. ウイルス情報 ipedia. `http://seclab.ipa.go.jp/zha-virusdb/web/Top.php`.

[20] Hispasec Sistemas. Virustotal. `https://www.virustotal.com/`.

[21] McAfee. Mcafee ウイルス検索. `http://www.mcafee.com/japan/security/vlibrary.asp`.