博 士 論 文

未知マルウェア対策技術に関する研究

Yasuyuki TANAKA

田中 恭之

情報セキュリティ大学院大学

情報セキュリティ研究科

情報セキュリティ専攻

2018 年 3 月

目次

第 1 章　序論 1

1.1　本研究の背景 1

1.2　マルウェア対策研究分野と貢献技術・本研究の分野 2

第 2 章　統計的手法を用いたマルウェア検出 5

2.1　概要 5

2.2　関連研究 5

2.3　評価検体 7

2.4　提案方式 10

2.5　評価 15

2.6　機械学習への適用 18

2.7　議論 21

2.8　本章のまとめ 22

第 3 章　攻撃コード構成の特徴を用いた ROP コード検出 23

3.1　概要 23

3.2　攻撃コードと防御メカニズム 24

3.3　関連研究 27

3.4　提案方式の着眼点 29

3.5　提案方式 33

3.6　実験 38

3.7　議論 43

3.8　本章のまとめ 46

第 4 章　効果的なブラックリスティングの検討 47

4.1　概要 47

4.2 関連研究 47
4.3 観測環境とデータセット 48
4.4 URL の生存分析 53
4.5 URL の挙動分析 57
4.6 各カテゴリの特性考察 64
4.7 議論 65
4.8 本章のまとめ 68

第 5 章 結論 70

参考文献 72

表目次

2.1 4AV 検体の内訳 8
2.2 未知検体の詳細 9
2.3 正常検体の内訳 9
2.4 パック有無の内訳 10
2.5 ヘッダ情報の内訳 11
2.6 モデルの独立変数と有意確率 13
2.7 各独立変数のオッズ比 14
2.8 アルゴリズム毎の条件分岐数 14
2.9 各検証における検体の組み合わせ 15
2.10 Adobe で用いられる変数 16
2.11 各検証における正答率 16
2.12 各検証における AUC 値 18
2.13 各検証における SVM での正答率 19
2.14 各検証における SVM での FPR 及び FNR 19
2.15 変数毎の処理時間 20
2.16 パラメータ調整後の SVM での正答率及び FPR,FNR 20

3.1 実行権限付与関数例 32
3.2 VirtualProtect 関数の引数 32
3.3 Pre check 条件 36
3.4 Basic check 条件 37
3.5 ROP に用いられる代表的な DLL 例 38
3.6 ROP 検体 40
3.7 実験結果 42
3.8 使われた ROPgadget の例 43

3.9 エンコード詳細 44
3.10 ROPgadgets のユニーク数 44
3.11 同一 ROPgadgets が使われた割合 44
3.12 ROPgadgets による分類 44

4.1 観測環境詳細 50
4.2 ステイタステーブルの例 51
4.3 マルウェアハッシュテーブルの例 51
4.4 IP アドレステーブルの例 51
4.5 AS 名トップ 10 の詳細 52
4.6 3 カテゴリの分類 53
4.7 *IntervalMax* と *IntervalMin* の平均値 55
4.8 *IntervalAve* と *IntervalStd* の平均値 56
4.9 *UniqIPMax* と *UniqIPMin* と *UniqIPAve* の *UniqIPStd* 平均値 59
4.10 *VTsubmit* と *VTscore* の平均値 61
4.11 観測回数と *Unsubmitted rate* の関係 62
4.12 *Netloc* と *Path* と *Query* の平均値 64

図目次

2.1 各検証における ROC グラフ 17

3.1 基本的な攻撃の流れ 25
3.2 ROP コードの動作原理 26
3.3 高度な攻撃の流れ 27
3.4 悪性文書ファイルの例 30
3.5 ROP コードが積まれたスタック 32
3.6 ROP コードの例 34
3.7 全体フローチャート 35
3.8 Final Check 部フローチャート 39

4.1 観測環境 49
4.2 死活監視定義 50
4.3 *Lifetime* と *Activedays* の CDF グラフ 55
4.4 *Stop5* (dark lines) と *Stop10* (light lines) の CDF グラフ 56
4.5 *UniqIP* の CDF グラフ 58
4.6 *IPEntropy* の CDF グラフ 60
4.7 観測回数と *VTScore* 62
4.8 アンチウイルス製品の判定結果 63

第 1 章

序論

1.1 本研究の背景

マルウェアは増加の一途をたどり，McAfee 社の資料 [1] によると，2017 年第二四半期で，6 億個に達しており，また，未知のマルウェアも近年急増している．現状のアンチウイルスソフトが，マルウェアの感染などを検知できるのは 45% 程度でありそれ以外は検知できないとう大手セキュリティ会社幹部の衝撃的な発言に関する報道 [2] からも窺い知れるように従来のシグネチャをベースとしたマルウェア検出手法は限界と考えられる．次に，ゼロデイ脆弱性の問題がある．ゼロデイ脆弱性とはセキュリティ更新プログラム (パッチ) が未公開な状態の脆弱性である．これらのゼロデイ脆弱性は標的型攻撃へ悪用されるため， 社会的問題となっている．標的型攻撃では，業務を装ったメールに攻撃コードを組み込んだ文書ファイルが添付されており，受信したユーザがその文書ファイルを閲覧することで，攻撃コードが実行され，最終的に企業が保有する機密情報の搾取等が行われるのが典型例である．さらに，マルウェアが配布されるインターネット上のサイトが多様化している問題がある．攻撃者の闇市場で流通する攻撃ツールキットの普及により，容易に新しい未知マルウェアが作り出され，マルウェア配布基盤自体も多様化し，一部はビジネスモデル化しており，対策が困難になっている．このような状況を受け，本研究では，未知のマルウェア対策技術の検討を行う．2,3,4 章で示す 3 つの異なるアプローチを用い，アンチウイルスソフトで検出が困難なマルウェアを，できるだけ軽量かつマルウェア感染前に検出することを目指す．

1.2　マルウェア対策研究分野と貢献技術・本研究の分野

マルウェア対策研究分野は，マルウェア自体の特定と，マルウェアに関連する通信の特定の 2 分野に大きく分かれる．マルウェア自体の特定は，マルウェアのファイル自体を正常ファイルと識別する，もしくは，マルウェアを分類することを目的とし，動的解析と静的解析の 2 分野に分かれる．動的解析分野は，マルウェアを実環境で動かした挙動，例えば API コール等の詳細なログを取得できることから，識別よりも分類を目的にした研究，さらにはマルウェアが接続する悪性通信を特定する研究が多い．これらの知見はサンドボックス製品 [3] に反映され，未知のマルウェア検出に大きく貢献している．静的解析分野は，さらに，表層的にファイル自体の特徴を分析する研究分野とプログラムや攻撃コードを分析する 2 種類の分野に分かれる．前者は，特にアンチウイルスや IDS/IPS のシグネチャ技術に応用され，様々な商用製品がリリースされている．本研究の 2 章がこの分野に該当し，ファイル自体のヘッダ情報の特徴を，シグチャではなく，統計的な手法や機械学習を利用して検出を行う．後者の応用例は多くはないが，例えば，Microsoft Enhanced Mitigation Experience Toolkit (Microsoft EMET)[4] のように，プログラムコードの特性，特に攻撃コードの特性を捉えて，緩和策を提供するツールとして提供されている．本研究の 3 章がこの分野に該当し，ROP と呼ばれる特定の攻撃コードの特性を捉えて検出を行う．

次に，マルウェアに関連する通信を特定する研究領域は，マルウェア感染前の通信の特定する分野（入口系対策分野）と，マルウェア感染後の通信の特定をする分野（出口系対策分野）の 2 種類に分かれる．入口系対策分野は，一般ユーザーが Web ブラウザでアクセスするとマルウェア感染するような，インターネット上の悪性なサイト等の特定を行い，それらのサイトをブロックすることで，マルウェア感染を未然に防ぐことを目的とする．本研究の 4 章がこの分野に該当し，長期的に悪性サイトを観測し，効果的にブラックリスティングすることを検討する．出口系対策分野は，マルウェア感染後のマルウェアが行う通信の特定を行い，マルウェア感染者の特定や，情報搾取，DDoS 攻撃等の 2 次攻撃を抑止することを目的とする．これらの技術により特定した悪性な通信先は，ブラックリストとして，Firewall 等に適用する形で用いられる．

本研究では，未知マルウェアや脆弱性を，マルウェア感染や攻撃が拡大する前に，できるだけ軽量な方式で検出することを目指したため，2,3,4 章で取り組む，2 つの静的解析分野及び入口系対策分野の研究分野を選択した．静的解析分野は，2 章で説明する様々な研究の歴史があるが，動的解析と比較すると方式自体は軽量で，また，工夫次第で伸びし

ろがあると考えた．入口系対策分野は，感染を未然に防ぐことは，実際に感染が発生して対処するよりもコスト面で非常に優位であると考えた．今回の研究の対象外とした動的解析分野と出口系対策分野は，サンドボックス関連研究や製品と密接に関わり，最近の研究は，多種環境を考慮したサンドボックス製品の解析結果を利用した研究 [5]，サンドボックス解析を回避するマルウェアに関する研究 [6] にシフトしている．

次に，本研究で取り組んだ 3 つのアプローチを，間接的か直接的かの観点で特徴付ける．直接的な方式の例として，マルウェアのファイル本体の特徴を利用して検出する方式がある．本研究では，2 章がこの方式分野である．準直接的な方式の例として，攻撃プロセスの特徴を利用するもので，共通に使われる攻撃コードを検出することでマルウェア検出を行う方式がある．本研究では，3 章がこの方式分野である．間接的な方式の例としては，多くのマルウェアはインターネット経由で配布されることから，そのような悪性サイトを検出し，そのサイトをブラックリストとしてアクセスさせない方式がある．本研究では，4 章がこの方式分野である．最も間接的な方式の例としては，セキュリティインテリジェンスを利用し，攻撃者をプロファイリングする方式である．本研究では，この方式分野は対象としない．一般に，直接的な方式のほうが，ターゲットとするマルウェアを確実に検出できるが，マルウェアの変化には追従しづらい．一方で，準間接的や間接的な方法は，マルウェアに共通して用いられる攻撃コードの検出や，変化するマルウェアが配布されるサイトのブロックにより，マルウェアの変化に対して有効な方法である．最も間接的な方式では，マルウェア配布が未然に抑止できる可能性がある．直接的，準直接的，間接的方法は，マルウェアの確実な検出とマルウェアの変化においてトレードオフの関係にあることから，これら 3 つを研究することで，網羅的に未知マルウェア対策に繋げられると考えた．

2 章「統計的手法を用いたマルウェア検出」は，直接的な方式であり，ファイルの静的な特徴に対して統計的手法を用いてマルウェア判定を行う．ファイルの多数のヘッダ情報から，判定に有効な情報に絞りモデルを構築し，そのモデルを利用することで軽量にマルウェア判定する．本手法は特に，インターネット上のマルウェアダウンロードサイトから配布されるような一般に広く流通するマルウェアに等に有効である．アンチウイルスソフトが検出できなかったマルウェアを未知マルウェアと定義し，それらを検出可能なことを確認した．実験では，パッキングの有無，アンチウイルスソフトの検知有無を考慮しマルウェアダウンロードサイトから収集した未知マルウェアを含むデータセットで，マルウェアか正常ファイルかの識別精度評価を行った．結果，従来手法に比べ識別性能が高いことを示した．さらに選定した変数を，代表的な機械学習手法に適用し，さらに高精度で識別できることを確認した．

3 章「攻撃コード構成の特徴を用いた ROP コード検出」は，冒頭に述べた準直接的な方式であり，ROP(Return-Oriented Programming) と呼ばれる攻撃コードが，シェルコードの複合ルーチンの外側に配置されると言う攻撃コードを構成する上での特徴を利用して検出を行う．本手法は，2 章での検出対象である一般に広く用いられるマルウェアでなく，標的型攻撃や未知であるゼロデイ脆弱性とともにも用いられる，ある特定のマルウェアに有効である．ROP コードは，ここ数年の脆弱性においてホスト側の防御機構を突破することを目的として，多くの攻撃コードに付加される．特に，昨今の標的型攻撃で用いられる手口で，攻撃コードを埋め込んだ悪性文書ファイルを送付し，被害者がファイルを開くことでマルウェア感染等が引き起こされ，情報の搾取等がなされるケースがある．この ROP コードを静的に検出することで，悪性文書ファイル判定を行う方式を提案し，実際の検体にて評価を行った．

4 章の「効果的なブラックリスティングの検討」は，冒頭に述べた間接的な方式であり，2 章や 3 章で示すマルウェア自体の特徴を捉えて検出するのではなく，マルウェアのダウンロードサイトを，効果的にブラックリスティングし未知マルウェア対策につなげるものである．これにより，マルウェア自体のハッシュ値等の特徴が変化して，未知マルウェアがダウンロードされた場合でも，検出可能である．実際には，約 43,000 個のマルウェアダウンロードサイトを，約 1.5 年にわたり長期観測を行い，幾つかのサイトはとても長い期間にわたりマルウェアダウンロードサイトとして生存を続けること，また幾つかのサイトは消滅と復活を繰り返しながらマルウェアダウンロードサイトとして活動を続けることがわかった．マルウェアの変化に応じて，マルウェアダウンロードサイトの 3 つのカテゴリに分けて解析し，それぞれの特徴に応じた効果的なブラックリスティングを検討した．

なお，本研究で示す 3 つのアプローチは，独立した方式であり，多層防御の観点で併用することで，未知マルウェアに対する感染を未然に防ぐ効果を発揮する．

第 2 章

統計的手法を用いたマルウェア検出

2.1 概要

2 章では，シグネチャ検出では追いつかない現状を受け，統計的方法を用い，既知マルウェアに加えて，アンチウイルスソフトが検出できない未知マルウェア判定を試みる．これは 1.2 章で述べた，静的解析分野の中で，表層的にファイル自体の特徴を分析する研究分野に位置付けられ，直接的なアプローチである．有効と考えられるファイルの静的な情報から特徴量となる変数を機械的な抽出と経験則を合わせて選定し，統計的方法であるロジスティック回帰分析を用いてモデルを構築する方式を提案する．変数選定では，将来的にツールとして実装し高速に使えることを考慮し，変数の数を吟味し，出来る限り有効な変数に絞っている．構築した提案モデルを用い，パッキングの有無やアンチウイルスソフトの検知有無を考慮し収集したデータセットで，マルウェアか正常ファイルかの識別における精度を評価し有効性を示した．さらに選定した変数を，代表的な機械学習手法に適用し高精度で識別できることを示した．

2.2 関連研究

マルウェアを正常ファイルと識別する，もしくは，マルウェアを分類する研究領域は，動的解析と静的解析の 2 種類の分野に分かれる．動的解析分野は，マルウェアを実環境で動かした挙動，例えば API コール等の詳細なログを取得できることから，識別よりも分類を目的にした研究，さらにはマルウェアが接続する悪性通信を特定する研究が多い．さらにここ数年，市販のサンドボックス製品の成熟を受けて，多種環境を考慮したサンドボックス製品の解析結果を利用した研究 [5]，サンドボックス解析を回避するマルウェアに関

する研究 [6] にシフトしている．他方で，静的解析分野はサンドボックスの登場のような大きな節目はないものの，過去から様々なアプローチの研究がされている．本提案方式も静的解析分野に含まれることから，静的解析分野の研究分野を説明する．静的解析分野は主に，(1)n-gram や可読文字列等を用いた識別，(2) 逆アセンブルした結果を用いた識別・分類，(3) 情報エントロピーを用いた識別，(4) ヘッダ情報を用いた識別，に分かれる．

- n-gram や可読文字等を用いた識別

文献 [7] では，検査対象ファイルのバイトシーケンスを n-gram を用い抽出した特徴量を利用する手法がある．最適な n の値として 500 と示されている．また，UNIX strings コマンドによる可読文字を用いる手法 [8]，同じく可読文字列をファイルの先頭バイトに特化して抽出した高速化手法 [9]，また可読文字列と UNIX hexdump コマンドの出力結果を併用する手法 [10] などがある．課題として，パックされた検体の識別性能が挙げられる．

- 逆アセンブルをした結果を用いた識別・分類

パックを解いてあることが前提となるが，ファイルを意味のある機械語と見なし逆アセンブルした結果を利用する．逆アセンブリ結果からオペレーションコードを抽出し n-gram を用いて特徴量を抽出する方法 [11]，最長共通分列を算出する手法 [12] がある．いずれもマルウェアの識別だけでなく，分類を行っている．課題として，計算量に加えて，パッカーは無数に存在し，パッキングをどう解くかが残される．

- 情報エントロピーを用いた識別

文献 [13] では，検査対象ファイルをいくつかの区間に分割し，区間毎の情報エントロピーを計算し，ファイル全体での統計から通常のファイルかマルウェアかを識別する．マルウェアの多くはパッキングされているため高い情報エントロピーを持つという性質を利用している．課題としてパックされていない検体の扱いがある．

- ヘッダ情報を用いた識別

文献 [14] では Windows の実行ファイルである PE ファイルフォーマットのヘッダ情報（以下 PE ヘッダ）とインポートされる DLL(Dynamic Link Library) の情報，約 190 種類の多くの特徴量を用い，パッキングされていても非常に高い精度でマルウェアか否かの判定が可能であることが示されている．PE ヘッダについては，2.4.1 章で解説する．文献 [15] では同じく PE ヘッダの情報から判定を試みるが，PE ヘッダのすべての特徴量を用いるのではなく，寄与度が高い約 7 種類の特徴量に絞り，実用に耐えうる判定精度を出している．また特徴量を減らすことで計算量が少なくてすみ，軽量なツールとして提供もされている．これらの関連研究から，PE ヘッダの情報は，判定対象がパッキングされているか否かの影響をうけにくく，マルウェアか否かの判定に有効な特徴量であることがわか

る．また特徴量を増やせば，単純に精度はあがるが，寄与度が高い特徴量を発見することで，特徴量を減らしても，実用に耐えうる判定精度を出せる可能性があることがわかる．関連研究では，これらの特徴量の意味の考察や，それらを踏まえた選定はされておらず，また，未知マルウェアに対する評価も行われていない．そこで，本稿では，機械的に有効な特徴量を選ぶ手法と，文献 [16, 17] に示されるような，例えばマルウェアのタイムスタンプは意図的に改変しているものが多い等の経験者の知見の双方を用いる．将来的に軽量なツールとして活用できることを目標に，特徴量を削減し，実用に有効な判定精度を目指す．さらに，冒頭で示したように，アンチウイルスソフトで検出できない検体が増えて脅威であることからそれらの検体の検出を試みる．

2.3　評価検体

本検討では，特に，未知マルウェアの入手がポイントとなる．マルウェアの取得環境や未知マルウェアの定義等を述べる．

2.3.1　マルウェア検体取得環境

攻撃に用いられやすい検体やアンチウイルスソフトのシグニチャで検出しない未知検体を，効率的に収集する為に，マルウェアダウンロードサイトに着目し検体収集を行った．文献 [18] によると多くのマルウェアダウンロードサイトは活動と休止を繰り返し，マルウェアの削除や配置が繰り返されることが報告されている．一方で，我々の観測では同一の URL にマルウェアが再配置される際に，短期間のみ別のマルウェア，特に未知マルウェアが配置されることがあることがわかってきており，マルウェアダウンロードサイトを定期的に巡回することで目的の検体を集めることとした．具体的には，Web クローラーを用い，悪性 URL をおおよそ一日に一度の頻度で定期的に巡回し，ダウンロードされたファイルを，ダウンロードの直後に，市販のアンチウイルスソフトで判定する環境を構築し，目的の検体を収集した．巡回対象である悪性 URL は，一般に公開される URL ブラックリスト等を含め様々な情報源から入手した．また攻撃者にアクセスを検知され，悪性サイトが活動休止となるのを避けるため複数拠点からアクセスし送信元 IP アドレスが固定とならない工夫をした．市販アンチウイルスソフトは国内で定評が高い，McAfee 社，Symantec 社，Trend Micro 社，Kaspersky 社の計 4 個を利用し，常にシグネチャは最新の物に更新される環境とした．

2.3.2　マルウェア検体

- 4AV 検体

4AV 検体は，2.3.1 章で示したマルウェア検体取得環境で，Web クローラーにより悪性 URL を巡回し，ダウンロードされた PE ファイルのうち，4 種類すべてのアンチウイルスソフトでマルウェア判定したものである．取得期間は 2014/6/1〜2015/5/1 である．表 2.1 にアンチウイルスソフトのシグネチャから推測したカテゴリ毎の内訳を示す．各検体の SHA1 ハッシュ値はユニークである．

- 未知検体

未知検体は，同様に 2.3.1 章で示したマルウェア検体取得環境で，検体取得時に 4 種類すべてのアンチウイルスソフトでマルウェア判定せず，正常と考えられるファイルを，後日，再度，アンチウイルスソフトでの判定を再度試み，すくなくとも 1 種類でマルウェア判定した検体である．表 2.2 に示す 2 つのデータセットを用意した．収集は複数回試みたが，未知検体がほとんど取得できないケースも多く，データはこの 2 つとした．再判定時期が，後日，再度アンチウイルスソフトでの判定を行った時期である．各検体の SHA1 ハッシュ値はユニークである．アンチウイルスソフトのシグネチャ名からカテゴリの推測が困難なものが多く存在したため，4AV 検体で行ったカテゴリ分類は実施できなかった．ただ，Trojan や Downloader に該当すると推測されるものは多く見られたが，Virus に該当するものは皆無であった．これは近年 Trojan 系の亜種が大量に作成・流通し，Virus にくらべて Trojan 系のシグネチャが多くリリースされる背景と一致すると考えられる．

表 2.1　4AV 検体の内訳

カテゴリ	個数
BackDoor	127
Downloader	779
Dropper	965
PWS	197
Trojan	1316
Virus	1219
合計	4603

表 2.2　未知検体の詳細

名称	収集時期	再判定時期	個数
未知検体 1	2014/12/01〜10	2015/5/29	1158
未知検体 2	2015/03/18〜28	2015/7/23	1166
合計			2324

2.3.3　正常検体

- Windows 検体

Windows 検体は，Microsoft Windows7 に導入時にインストールされるファイルである．Microsoft 社からリリースされているため，パッキングされておらず，セキュリティの署名が適切に存在する等，マルウェアとの判別がしやすいことが考えられるため，次に示すその他正常検体を準備した．

- その他正常検体

その他正常検体は，インターネット等から入手したフリーソフトで，念のためアンチウイルスソフトでマルウェア判定されないことを確認したものである．これらの正常検体を表 2.3 に示す．

表 2.3　正常検体の内訳

名称	個数
Windows 検体	2452
その他正常検体	2602
合計	5054

2.3.4　パッキング

マルウェアの多くはパッカーによりパッキングされてることが知られる．パッカーとは，マルウェア等の実行ファイルを圧縮し，その圧縮ファイルと展開のためのコードを組み合わせるツールである．パッカーを用いて作成されたファイルが実行されると，まず，展開コードが圧縮ファイルを展開し元の実効ファイル本体を復元後，実行される．後の評価で用いるため，マルウェア検体及び正常検体について，パッカー判定ツールである

PEiD[19] 及び linux の file コマンドの出力を用い，各検体がパッキングされているか否かを判定し，内訳を表 2.4 に示した．

表 2.4　パック有無の内訳

名称	パック無し	パックあり
4AV 検体	2443	2160
未知検体 1	894	264
未知検体 2	986	180
Windows 検体	2452	0
その他正常検体	2226	376

2.4　提案方式

未知マルウェアの検出を目標として，ヘッダ情報に着目し，機械的な抽出と経験則を組み合わせて有効な特徴量を選定し提案モデルを構築する．将来的に軽量なツールとして活用できることを目標に，特徴量を削減し，実用に有効な判定精度を目指す．

2.4.1　ヘッダ情報の抽出

Windows の実行ファイルは，PE ファイルフォーマット [20] と呼ばれる形式をとる．PE ファイルフォーマットは，ローダがプログラムをロードするのに必要なデータが格納される DOS 及び PE ヘッダと，プログラムコードや各種データが格納されるセクションで構成される．本研究では特に PE ヘッダ情報に着目する．PE ヘッダ情報の抽出には，PE ファイルパーサーである pefile[21] を用いて行った．抽出したヘッダ情報は合計 118 個であった．表 2.5 に内訳としてジャンル毎の個数を示す．特に Resource セクションはマルウェア識別に有意な特徴が現れやすいと考え ResourceType 毎に分けることとした．次に，4AV 検体及び正常検体，合計 9657 検体をランダムに半分に分け，一方をヘッダ情報抽出検体セット，他方を評価用セットとした．ヘッダ情報抽出検体セットは 4828 個，評価用セットは 4829 個である．ヘッダ情報抽出セットを対象として，この 118 個のヘッダ情報を抽出した．4AV 検体は，その定義の通り，市販アンチウイルスソフト 4 社すべてがマルウェア判定したものであり，マルウェアに共通する特徴量が得られると考え，またこれらの特徴量は未知マルウェア判定に有効ではないかと推測した．

表 2.5　ヘッダ情報の内訳

ジャンル	個数
FileHeader	7
OptionalHeader	30
OptionalHeaderDirectorySize	16
OptionalHeaderDirectoryRVA	16
SectionTable	28
ResourceType	21
合計	118

2.4.2　有効なヘッダ情報の絞り込み

有効な変数の絞り込み手法として，ステップワイズ法 [22] がある．ステップワイズ法を試みたが，計算量が大きくなり有効な結果が得られなかったため，本研究では，Lasso(Least absolute shrinkage and selection operator) 推定法 [23] を用いた．Lasso 推定法は Tibshirani によって提唱された方法で，正則化を行うことで，影響しないパラメータをゼロに推定してくれる方式である．ほとんどの値がゼロになる解は疎な解と呼ばれ，実装上，非ゼロのパラメータ要素だけを保持すれば良いため，メモリ効率が良くなり有効である．本研究では，将来的にツールとして高速に使えることを考慮したため，効率よくモデル推定，変数選択が行えることを見通して同手法を採用した．実際には Lasso の実装である R のライブラリ glmnet[24] を用い，ペナルティの値を変化させ，各変数の係数を確認し変数候補を絞った．変数間の相関が高いと多重共線性が発生するので，変数間の相関係数も参考にした．文献 [25] 等の実験結果から，改変されたタイムスタンプ，エントリーポイントが示すセクションの名前が通常と異なる，TLS(thread local storage) [20] エントリの存在，Resource エントリの使われ方，ゼロバイトのセクションの存在，IAT(Import Address Table)[20] に頻繁に登録される関数，エントロピーが極端に高いもしくは低いセクションの存在，バージョン情報の使われ方等，PE ヘッダにおいて，マルウェアと正常ファイルで差異が出やすいと考えられる経験則がある．これらの経験則も加味して最終的な変数選択を行った．118 個のうち以下 10 個に絞った．以下にその定義を示す．

- 変数 1：FH_Characteristics

ファイルヘッダの Characteristics は，該当ファイルが，実行可能か否か，シンボル情報が含まれているか否か，再配置情報が含まれているか否か，デバック情報が含まれているか否か，システムファイルか否か等，多数の情報が含まれている．経験則からも，マルウェアと正常ファイルで差異が出やすい変数と考えられる．

- 変数 2：FH_TimeDateStamp

ファイルヘッダの TimeDataStamp は UNIX 時間で示されるタイムスタンプで，マルウェアの場合意図的に改ざんする等，経験則からも，正常ファイルと差異が出やすい変数と考えられる．

- 変数 3：FileAlignment

オプショナルヘッダの FileAlignment は，ディスク上で各セクションが配置されるアライメントを示し，各セクションの開始アドレスは FileAlignment の倍数になる．

- 変数 4：MajorSubsystemVersion　及び　変数 5：MinorSubsystemVersion

オプショナルヘッダの MajorSubsystemVersion 及び MinorSubsystemVersion 動作する WindowsOS のバージョンを示す [26]．既存の経験則でも本変数は相関が高いことがわかっている．両変数は関連研究の Adobe[15] でも加算して 1 つの変数とする形で，特徴量として用いている．

- 変数 6：Subsystem

オプショナルヘッダの Subsystem は，CUI アプリケーション，GUI アプリケーション，ドライバ等のネイティブケーション等を示す．

- 変数 7：Debug_Size

デバック情報が格納されるデバックセクション領域の大きさを特徴量とする．デバッグセクション領域の大きさは，Adobe でも特徴量として用いている．

- 変数 8：RT_Version　及び　変数 9：RT_Dialog　及び　変数 10：RT_Group_Cursor

いずれもリソースセクションに関する変数である．Version はバージョン情報，Dialog はダイアログ情報，Group_Cursor はハードウェアに依存したカーソルリソースに関する情報である [26]．特に Version 情報については，経験則からも，正常ファイルの場合，作成者や所属会社の情報やコピーライト情報が記録されている一方で，マルウェアの場合，これらの情報が改変や削除されるケースもあり，差異が出やすい変数と考えられる．

次に，この 10 変数をロジスティック回帰分析に用いてモデルを構築する．ロジスティック回帰分析は，独立変数を複数個とる回帰分析である重回帰分析の一種であり，重回帰分析が従属変数として量的変数を取ることと比較して，従属変数として質的変数を取ることができる特徴がある．マルウェアか否かを判定する場合，2 値の質的変数を出力ととればよい．また特徴量である各変数の有意確率やオッズ比等を得ることが出来るため変数選択

をする上で有効である．オッズ比は，その変数が，質的変数であれば 0 である場合に比べて 1 の場合，何倍，マルウェアであると判定されやすいかを示し，量的変数であれば，1 増えた場合，何倍，マルウェアであると判定されやすいかを示す．したがって，オッズ比の大小も，変数選定の上での参考にすることが出来る．

構築したモデルの各独立変数の係数と有意確率を表 2.6 にオッズ比を表 2.7 に示す．他の変数と比較して FH_TimeDataStamp は有意確率が低く，オッズ比でも大きな効果は発揮していないが，前述した変数の意味からの考察で意味が有ると判断し加えている．オッズ比は FH_TimeDataStamp を除くすべての変数で 1 未満となっている．例えば，変数 DEBUG_Size のオッズ比は，表 2.7 から約 0.8697 であるが，これは DEBUG_Size の値が 1 大きくなると，0.8697 倍，マルウェアである確率が高くなることを示す．言い換えると DEBUG_Size は，大きいほど，正常検体である確率が高いと言える．つまり，オッズ比が 1 未満とは，各変数が小さくなればマルウェアである確率が高まることを示している．

表 2.6　モデルの独立変数と有意確率

独立変数	偏回帰係数	有意確率
切片	2.64E+01	<2e-16
FH_Characteristics	-7.04E-05	<2e-16
FileAlignment	-6.93E-01	<2e-16
MajorSubsystemVersion	-3.16E+00	<2e-16
MinorSubsystemVersion	-8.98E-01	1.21E-15
Subsystem	-5.43E+00	<2e-16
DEBUG_Size	-1.40E-01	<2e-16
RT_Version	-7.91E-01	<2e-16
FH_TimeDateStamp	-4.06E-09	0.0023
RT_Dialog	-8.62E-02	1.12E-05
RT_Group_Cursor	-1.19E-01	4.30E-05

2.4.3　変数を絞り込む意義

変数を絞るメリットを評価する．我々も将来的に R 等の統計・機械学習プラットフォームに依存しないツールとして軽量に実装・動作することを目標としているため，評価に

表 2.7　各独立変数のオッズ比

独立変数	オッズ比
FH_Characteristics	1.00E+00
FileAlignment	9.99E-01
MajorSubsystemVersion	4.67E-02
MinorSubsystemVersion	4.07E-01
Subsystem	4.39E-03
DEBUG_Size	8.70E-01
RT_VERSION	4.53E-01
FH_TimeDateStamp	1.00E+00
RT_Dialog	9.17E-01
RT_Group_Cursor	8.88E-01

は同様の思想の Adobe[15] で提供される言語 python で実装されているツール [27] を用いた．同ツールでは，4 種類の統計・機械学習アルゴリズムが実装されている．各アルゴリズムそれぞれにおいて，1 変数が関係する条件分岐処理がいくつ存在するかカウントした．1 変数あたりの分岐処理の数の平均値を表 2.8 に示す．アルゴリズムによって条件分岐の数は異なるが，特に，アルゴリズム PART の場合，1 変数あたりの分岐の数は 25 となり，仮に変数の値が 118 個の場合，条件分岐の数は 2950 個となり，変数の数 10 個の場合の条件分岐の数 250 個と比較すると大きい値となり，コードの肥大化と処理効率の低下に繋がる．そのため特にツールとしての用途の場合，判定精度の大きな劣化のない範囲での有効な変数の吟味と選定は重要である．

表 2.8　アルゴリズム毎の条件分岐数

アルゴリズム	条件分岐数平均値
J48	2
J48Graft	4
PART	25
Ridor	14

2.5 評価

2.5.1 概要

各評価で用いる検体の組み合わせを表 2.9 に示す．ここで 4AV 評価検体及び正常評価検体は，2.4.1 章で示した，他方のセットである評価用セット合計 4829 個である．つまり，提案方式の汎化能力を適切に評価するため，評価用として，ヘッダ情報抽出に用いていない検体を使用する．検証 1 ではパッキングを意識せず，4AV 検体と正常検体すべてを対象としてマルウェアか否かの識別を目的とする．検証 2・3 は，検証 1 検体について，パッキングの影響を加味した観点で，パッキングあり，パッキング無し検体での識別精度を評価する．最後に，検証 4・5 では，未知検体に対する識別精度を評価する．

表 2.9　各検証における検体の組み合わせ

名称	マルウェア	正常
検証 1	4AV 評価検体すべて	正常評価検体すべて
検証 2	4AV 評価検体パック有	正常評価検体パック有
検証 3	4AV 評価検体パック無	正常評価検体パック無
検証 4	未知検体 1 すべて	正常評価検体すべて
検証 5	未知検体 2 すべて	正常評価検体すべて

評価環境は R を用い，判定にはロジスティック回帰を用いた．またそれぞれの検証で Adobe[16] で用いられる変数と比較して行った．表 2.10 に Adobe で用いられる変数を示す．

2.5.2 結果

表 2.11 に各検証における正答率を表す．それぞれの検証観点は表 2.9 に対応する．ここで，正答率は以下のように定義する．

$$\text{正答率} = \frac{TruePositive\,\text{数} + TrueNegative\,\text{数}}{\text{サンプル総数}}$$

正答率の評価の精度を高める為に，10-fold cross-validation を行った．具体的には，サンプル総数をランダムに 10 分割し，そのうち 1 つをテストデータ，残りを訓練データと

表 2.10　Adobe で用いられる変数

独立変数
FH_NumberOfSections
ImageVersion
EXPORT_Size
RESOURCE_Size
DEBUG_Size
IMPORT_Rva
VirtualSize2

して学習と予測を行う．次に，テストデータと訓練データを入れ替えて合計 10 回実施し，10 回の平均値を正答率とする．

表 2.11　各検証における正答率

名称	提案方式	Adobe
検証 1	0.9258	0.8519
検証 2	0.948	0.8681
検証 3	0.9393	0.859
検証 4	0.9002	0.8174
検証 5	0.9342	0.8563

さらに分類器の識別精度を考察するために Receiver Operating Characteristics (ROC) グラフを描いた．提案変数を実線, Adobe での変数を破線として, 図 2.1 に示す．ROC グラフは，縦軸に True Positive Rate(以下 TPR)，横軸に False Positive Rate(以下 FPR) をとり，識別における閾値を変化させ，各閾値における TPR と FPR の値をプロットしたものである [29]．TPR と FPR は以下と定義する．

$$TPR = \frac{TruePositive\,数}{TruePositive\,数 + FalseNegative\,数}$$

$$FPR = \frac{FalsePositive\,数}{FalsePositive\,数 + TrueNegative\,数}$$

また ROC グラフの曲線下の面積を示す AUC(Area under the curve) の値を算出して表 2.12 に示した．ROC グラフで (FPR,TPR)=(0,0) から始まり，(FPR,TPR)=(0,1)

を通り，(FPR,TPR)=(1,1) に進む曲線が理想的な分類器を示し，そのとき，AUC の値は 1 となる．

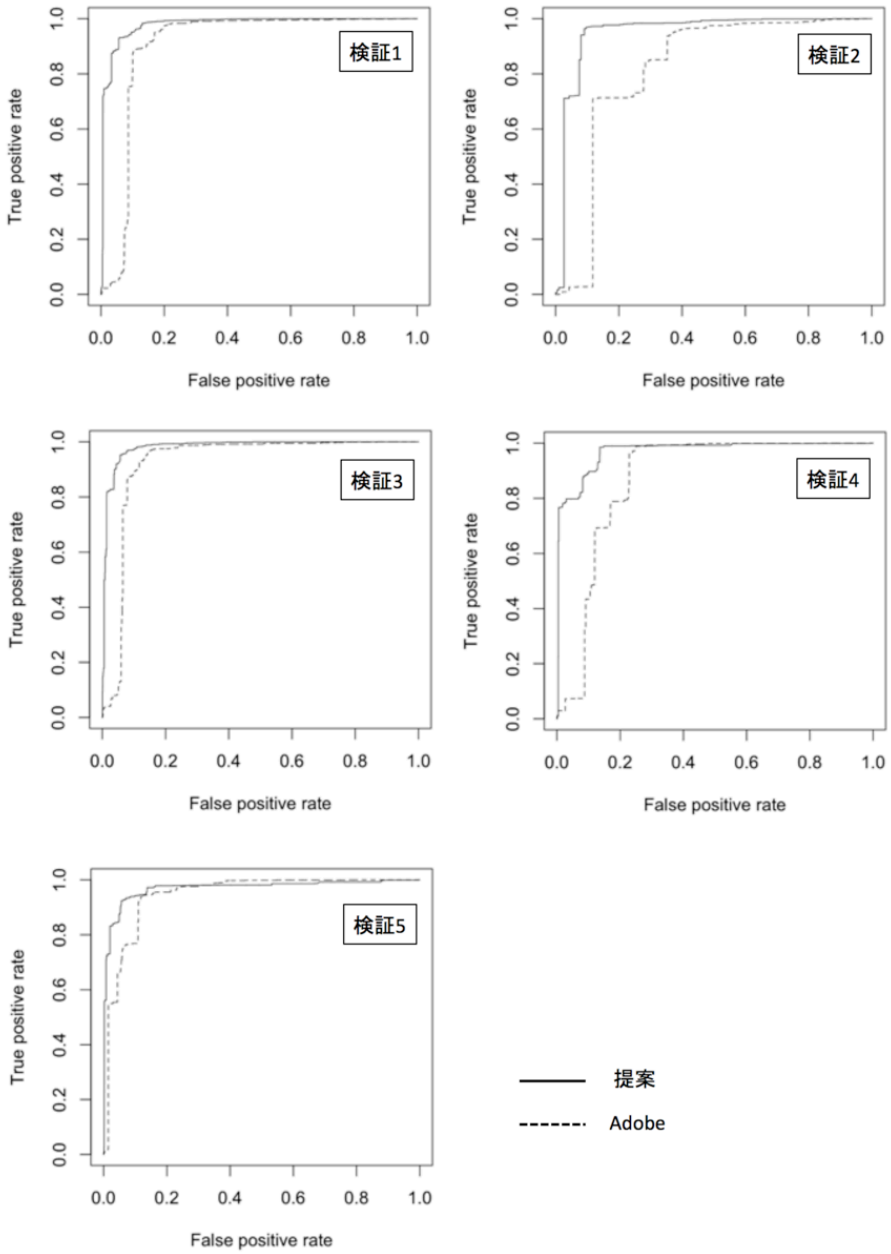

図 2.1　各検証における ROC グラフ

表 2.12　各検証における AUC 値

名称	提案	Adobe
検証 1	0.9792	0.9036
検証 2	0.9507	0.8165
検証 3	0.9818	0.9252
検証 4	0.9699	0.8698
検証 5	0.9701	0.9451

2.5.3　考察

表 2.11 より，正答率は，検証 1〜5 のすべての検証で，提案方式は Adobe を上回った．検証 2 及び 3 のパッキング有無の観点は，正答率からみると提案方式と Adobe 共に検証 2 のパッキングがある方が，検証 3 よりも正答率が高い結果となった．しかし図 1 の ROC グラフ：検証 2 は提案方式と Adobe 共に，他と比べて理想的な ROC グラフからは外れており，また，表 2.12 の検証 2 の AUC 値を見ると，検証 2 は他と比べて低くなっていることから識別器の性能としては他の検証と比べると低い結果となり，パッキング検体の識別は他と比べると困難であることがわかる．検証 4 及び 5 の未知検体について，検証 4 が一番正答率が低くなった．検証 4 は Adobe でも 0.8174 と全体で一番低くなった．一部の未知検体の判定は困難であることがわかる．しかしながら，ROC グラフ及び AUC 値から，Adobe では検証 4 はグラフが理想から外れてきており AUC 値も 0.8 台に下がっているが，提案方式は検証 4，検証 5 共に 0.970 付近の値となっていて識別値の性能は良いことがわかる．

2.6　機械学習への適用

本章では，今回選定した 10 変数の条件は同等として，識別器をロジスティック回帰から，機械学習で用いられるサポートベクターマシン（以下 SVM）で判定した結果を示す．SVM では各変数の有意確率やオッズ比等は得られず 2.4.2 章で示した各独立変数の貢献度のような考察には向かない．しかしながら，実用的には各変数の貢献度の算出よりも，モデル全体として良く，その結果，指標である正答率が高いほうが有用である．そこで識別性能には定評がある SVM に 10 変数を適用し，評価実験を行った． R のライブラリ e1071[28] を用い，カーネルはガウシアンカーネルを用いた．同ライブラリでは，同カー

ネルの他に，線形内積カーネル，多項式カーネル，シグモイドカーネルを選択できる．これらの 4 カーネルで比較実験を行った結果，一番高い精度が確認できたため，同カーネルを用いることとした．10-fold cross-validation を行って評価した正答率を表 2.13 に示す．この段階ではパラメータチューニングは行っていない．つまり R のライブラリ e1071[28] のデフォルト値である gamma=1，cost=1 の結果である．検証 1〜検証 5 のすべての検証で Adobe より高い正答率となった．表 2.14 に FPR 及び False Negative Rate(以下 FNR) を示す．FPR については両者とも検証 2 が一番悪い値となった．2.5.3 章での結果同様，パッキング検体は他と比べると困難であると考えられる．特に Adobe では 0.64 を越え過検知が懸念される結果となった．検証 4 の未知検体 1 では，両方式とも FNR が 0.07 を越え，見逃しの可能性が大きくなった．検証 1〜5 全体で，Adobe の FPR は 0.1 を越え全体的に過検知よりの方式であることがわかる．

表 2.13　各検証における SVM での正答率

名称	提案	Adobe
検証 1	0.952	0.8826
検証 2	0.9582	0.8864
検証 3	0.9596	0.8865
検証 4	0.9573	0.8841
検証 5	0.9581	0.8832

表 2.14　各検証における SVM での FPR 及び FNR

名称	提案		Adobe	
	FPR	FNR	FPR	FNR
検証 1	0.03669	0.059	0.1719	0.05341
検証 2	0.06626	0.02591	0.6471	0.01389
検証 3	0.02456	0.06446	0.1532	0.02821
検証 4	0.02433	0.07686	0.132	0.07513
検証 5	0.02912	0.04288	0.138	0.06089

次に，2.4.3 章で示した，変数を絞り込む意義について，さらに定量的に評価するため，10 変数の場合と 118 変数の場合で処理時間を計測した．対象は評価用セット 4829 個である．評価用セットをランダムに半分に分け，片方で，モデルを作成，もう一方で予測

し，それぞれ時間を計測する．次に，作成と予測のデータを入れ替え，同様に時間を計測し両者の平均値を取った．結果を表 2.15 に示す．単位は秒である．10 変数と比べて 118 変数の方が，モデル作成について約 26 倍，予測について約 21 倍の処理時間がかかることが確認できた．尚，表 2.13 における評価と同条件での 118 変数での正答率は 0.9601 となり，表 2.13 の検証 1，提案と比較して，0.0081 上回ったが大幅な向上とはならず 10 変数の有効性が確認できた．尚，測定環境は以下の通りである．

- ハードウェア：MacBook Pro
- プロセッサ：3GHz Intel Core i7
- メモリ：8GB
- OS：OS X 10.9.5
- ソフトウェア：R version 3.2.2, ライブラリ e1071

表 2.15　変数毎の処理時間

10 変数		118 変数	
モデル作成	予測	モデル作成	予測
0.4724	0.03680	12.18	0.7746

表 2.16　パラメータ調整後の SVM での正答率及び FPR,FNR

名称	正答率	FPR	FNR
検証 1	0.9701	0.02323	0.03731
検証 2	0.966	0.05236	0.03062
検証 3	0.9692	0.02198	0.04773
検証 4	0.9814	0.02048	0.02363
検証 5	0.976	0.01307	0.0471

SVM はパラメータ調整をすることで正答率があがるため，グリッドサーチを用い調整した．過学習を防ぐため，テストデータをパラメータ調整に含めないようにして行った．具体的には次の通りである．サンプル総数をランダムに 10 分割し，そのうち 1 つをテストデータ，残りを訓練データとして，訓練データのみを対象に R のライブラリ e1071 の tune.svm() 関数を用い最良なパラメータ gamma 及び cost を算出する．その値を使用し，モデルを作成し，テストデータに対し正答率を算出する．次に，テストデータと訓練

データを入れ替えて，前述したパラメータ算出と，テストデータでの評価を合計 10 回実施し，10 回の平均値を正答率とする．FPR 及び FNR も同様に算出した．パラメータ調整後の正答率及び FPR，FNR の値を表 2.16 に示す．正答率はすべての検証で 0.965 を超える値となった．特に検証 4 の未知検体について 0.98 を超える値，検証 5 の未知検体については 0.975 を超える値を取り，本稿で示した 10 変数の有効性が確認できた．FPR，FNR は検証 1〜5 で最大でも 0.05 台にとどまっており，同様に有効性が確認できた．

2.7　議論

本稿では未知マルウェア検出を焦点に方式を提案し評価を行ったが，以下の課題があると考えられる．

- 特徴量の回避を狙う検体

本稿では，ツールとして高速に判定できることを目標に，識別に有効な特徴量を 10 個に絞り込み，用意したデータセットに対し識別が可能なことを示した．しかしながら，マルウェア作成者がこの特徴量を知れば，回避可能な検体を作ることが可能であると考えられる．

- 未知マルウェアの定義と入手方法

本稿では，アンチウイルスソフトで，後日，検出可能な検体を未知マルウェアと定義した．しかし時間が長期間経過してもアンチウイルスでは検知できない検体も存在すると考えられるため，サンドボックス製品の併用等が考えられる．また，今回の検討の中で Web クローラーを用いることで未知マルウェアを得られる知見が得られ，それに基づき検体収集を行ったが，常に安定量得られる訳ではないので，さらに入手方法は検討する必要がある．

- マルウェアの分類

本稿では，マルウェアか正常ファイルかの 2 値判定を試みた．特徴量をさらに工夫することによって，マルウェアの分類ができる可能性がある．しかし特に最近の未知マルウェアは複数のアンチウイルスベンダの判定名も異なり，また，判定名からカテゴリが推測できないものが大半であり，正解データを得ることが困難である．

- 実装に用いるアルゴリズムの評価・選定

本稿では，ロジスティック回帰及び SVM を用い 10 変数を用いた識別精度について，また，SVM を用い処理時間について評価したが，ツールとして軽量に実装・動作する最終な目標を実現するためには，実装面も考慮した機械学習アルゴリズムの選定が必要になる．

2.8 本章のまとめ

シグネチャ検出では追いつかない現状を受け，統計的方法を用いマルウェア判定を行い，結果を示した．これは 1.2 章で述べた，静的解析分野の中で，表層的にファイル自体の特徴を分析する研究分野に位置付けられ，直接的なアプローチである．有効と考えられるファイルの静的な情報から特徴量となる変数を機械的な抽出と経験則を合わせて選定し，統計的方法であるロジスティック回帰分析を用いてモデルを構築する方式を提案した．変数選定では，将来的にツールとして実装し高速に使えることを考慮し，変数の数を吟味し，出来る限り有効な変数に絞った．構築した提案モデルを用い，パッキングの有無やアンチウイルスソフトの検知有無を考慮し収集したデータセットで，マルウェアか正常ファイルかの識別における精度を評価し有効性を示した．さらに選定した変数を，代表的な機械学習手法に適用し高精度で識別できることを示した．

第 3 章

攻撃コード構成の特徴を用いた ROP コード検出

3.1 概要

ゼロデイ脆弱性とはセキュリティ更新プログラム（パッチ）が未公開な状態の脆弱性である．シマンテック社によるとゼロデイ脆弱性の発生件数は，2011 年の 8 件，2012 年の 14 件に対して 2013 年は 23 件と近年増加傾向にある [29]．これらのゼロデイ脆弱性は標的型攻撃へ悪用されるため，社会的問題となっている [30]．標的型攻撃では，業務を装ったメールに攻撃コードを組み込んだ文書ファイルが添付されており，受信したユーザがその文書ファイルを閲覧することで，攻撃コードが実行され，最終的に企業が保有する機密情報の搾取等が行われるのが典型例である．

ユーザ環境に脆弱性がある場合，それを利用して，文書ファイルを閲覧しただけで攻撃コードを実行されてしまう．このため，ユーザ環境のセキュリティアップデートを適切に行い，パッチレベルを最新化しておけばこのような攻撃の被害は受けない．しかしゼロデイ脆弱性の場合，ユーザ環境はパッチが提供されておらず無防備な状態であり，仮に脆弱性自体が公表されていない場合，危険があることすら分からない状態となってしまう．

3 章では，対策が非常に困難であるゼロデイ攻撃対策を視野に入れ，標的型攻撃で多用される悪性文書ファイルを静的解析により安全かつ効率的に検出する手法を提案する．これは 1.2 章で述べた，静的解析分野の中で，プログラムコードを分析する研究分野に位置付けられ，準直接的なアプローチである．提案方式を実装し他のツールと比較した実験結果を通して有効性と課題を示す．本提案手法では，文書ファイルに埋め込まれた ROP 攻撃コードの検出により悪性文書ファイルを検出する．ROP 攻撃コードは，ROP

(Return-Oriented Programming) [31] 技法によってホスト側の防御機構を突破することを目的とする攻撃コードである．Microsoft 社の資料によると，ここ数年に発生した新しい脆弱性に対する攻撃において 95% 以上の割合で用いられ [32]，この中にはゼロデイ攻撃も含まれる．このため，ROP 攻撃を早期に検出することでゼロデイ対策につなげられると考えた．本手法では，実際の攻撃コードを分析した得られた特性：(1) ROP 攻撃コードの目的が共通している点，(2) シェルコード自体の暗号化対象とならない点，を利用して静的に ROP 攻撃コードの検出を行う．なお，本稿では，これ以降，この攻撃目的で用いられる ROP 攻撃コードを単に ROP コードと標記する．

本提案手法の利用シーンは次の 2 つである．1 つ目は，文書ファイル型検体解析の現場における解析者支援である．一般に，文書ファイル型検体の解析には，動的解析に加え詳細な機能を知るための人手による静的解析が行われる．動的解析では安全面の考慮に加えパッチレベル等環境を変化させ攻撃が発動する環境を探す必要がある．一方，静的解析では人手による解析コストが高いことから解析部分を絞る必要がある．さらに ROP を含む検体の場合，動的解析では環境のバリエーションが増加し，静的解析ではメモリ状態の解析等が加わることにより，両解析方式とも解析が困難となる．本提案手法により，対象検体に ROP が含まれているか否か，ROP が含まれている場合そのオフセット値や，ROPgadget のアドレス等の情報を得ることができる．これらの情報を参考にし，解析者は，さらに詳細な解析に進むことができる．2 つ目は，企業内の CSIRT 等の組織で，対象検体にゼロデイ攻撃の可能性がないかチェックをしたい場合である．3.6.5 章で示すように，同一の ROP コードが異なる脆弱性を攻撃するために用いられる傾向があるため，未知の脆弱性を突くゼロデイの悪性文書ファイルの早期検出に効果が見込める．

本提案手法の対象とする環境は, OS は 32 ビット Windows, 文書ファイルは, Microsoft Word (拡張子 doc), Microsoft Excel (拡張子 xls), Microsoft PowerPoint (拡張子 ppt), Rich Text Format（拡張子 rtf）である．

3.2　攻撃コードと防御メカニズム

3.2.1　悪性文書の基本的な攻撃の流れ

悪性文書ファイルとは，ファイル内に文書データ，攻撃コード，マルウェアを含む文書ファイルである．悪性文書ファイルは閲覧ソフトウェアの脆弱性を突いて感染するが，exe 形式のマルウェアとは異なり実行形式ではないためユーザは不安を感じずに開いてしまうことが多い [33]．図 3.1 のように，ユーザが悪性文書ファイルを開くと，閲覧ソフト

の脆弱性を攻撃するエクスプロイトコードが作動する．エクスプロイトコードは，閲覧ソフトの制御権をコントロールできるようにするまでの役割を担う．制御権が取られると，シェルコードが実行される．シェルコードは，主に，文書ファイル内に埋め込まれた実行形式のマルウェアを取り出して実行する．

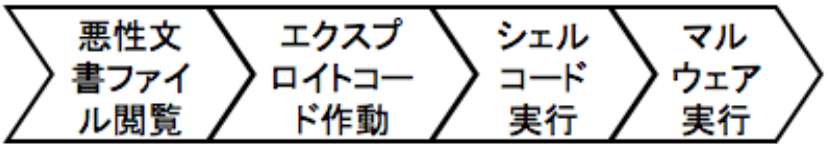

図 3.1　基本的な攻撃の流れ

3.2.2　防御メカニズム：DEP

DEP（Data Execution Prevention）とは，データ領域内でのコード実行を阻止するもので，Microsoft Windows では WindowsXP SP2 で導入された．エクスプロイトコードによりキックされるシェルコードは，メモリ上のプログラム領域ではなく，スタックやヒープと呼ばれるデータ領域内に配置される．DEP 有効下では，データ領域での実行権がなくシェルコードは実行できず攻撃者は攻撃に失敗する．

3.2.3　DEP を回避する ROP 等 Code-Reuse 攻撃

DEP を回避するため Return-to-libc という技法が用いられるようになった．これはメモリ中に存在する API 関数を直接コールする手法で，API 関数への引数を適切にスタックに積んでコールすることで API 関数を動作させ目的を達成することができる．さらにこの技法を進めた ROP（Return Oriented Programming）[31] 技法が最近主流となっている．ROP 技法を用いたコードの一例を図 3.2 に示す．ROPgadget とは，実行可能領域にある ret 命令で終了する数バイトのコード断片である．この例では，最終的にアドレス 0x50505050 に値 0xDEADBEEF を格納することを実現している．攻撃者は目的が達成できるように適切にスタックを積んでおく（①）．スタックの最上位の値にリターンするように調整しておき（②），最初に gadget No1 が実行される（③）．レジスタ EAX には意図した値（0x50505050）が格納され（④），ret 命令でスタックを参照し（⑤），次の gadget No2 にリターンする形で gadget No2 が実行される（⑥）．このよう

にして，最終的に gadget No3 が実行されることで目的が達成できる．ROP に関連する技法として JOP（Jump Oriented Programming）[34] が提案されている．ret の代わりに jmp を用いるものだが，gadget をコントロールする Dispatcher を用意し，jmp 先をこの Dispatcher にするように工夫している．また SOP（String Oriented Programming）[35] はフォーマットストリング脆弱性を利用する技法である．なお ROP，JOP，SOP は Code-Reuse 攻撃とも呼ばれる．

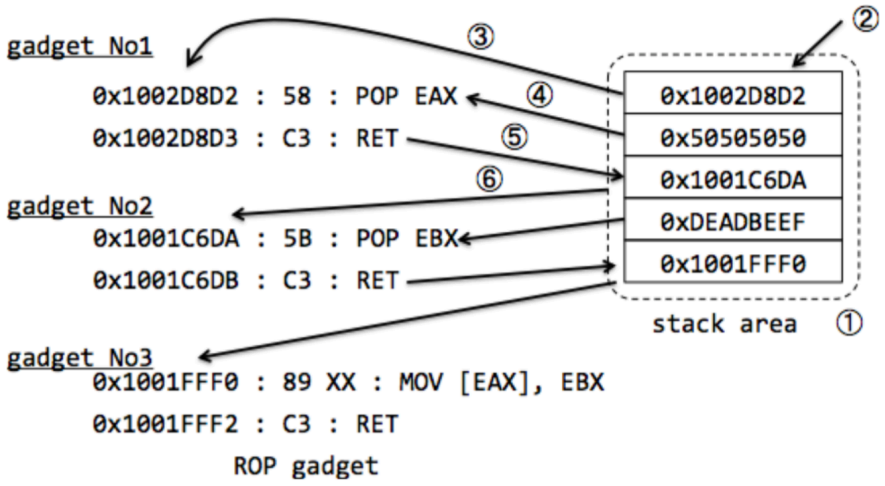

図 3.2　ROP コードの動作原理

3.2.4　防御メカニズム：ASLR

ASLR（Address Space Layout Randomization）はアドレス空間を OS 起動時にランダム化するもので，WindowsVista から導入されている．ASLR は，攻撃者が Return-to-libc や ROP の手法により API 関数やスタック・ヒープの固定な既知アドレスを利用することに対抗した方式である．

3.2.5　ASLR の回避

攻撃者は当該 OS が 32 bitOS である場合は，ランダム化されたアドレス空間をスキャンして必要なアドレスを見つけ出すことによって，現実時間・現実試行回数内で ASLR を回避可能である．また DLL によっては ASLR によるランダム化が行われないものが存在し，これを悪用した手法が最近の攻撃の主流となっている．さらに，ランダム化が行われている DLL しか得られない場合でも，脆弱性を利用して特定の DLL のベースアドレ

スを得ることで動的に ROP コードを組み立てる手法も登場している [36]．ここで ASLR が適用されていない DLL を ASLR 非対応 DLL，ASLR が適用されておりロード時にアドレスがランダム化される DLL を ASLR 対応 DLL と呼ぶことにする．

3.2.6　防御メカニズムを回避する攻撃コード

DEP（3.2.2 章）や ASLR（3.2.4 章）といった防御メカニズムを回避するために，攻撃者は ROP 技法（3.2.3 章）を用いる．また IDS やアンチウイルス等の検出メカニズムを回避するために，攻撃者はシェルコードを暗号化する．この ROP コードや暗号化シェルコードが埋め込まれた悪性文書ファイルの動作順序を図 3.3 に示す．図 3.1 と比べると ROP コード実行と復号コード実行が増えている．実行順序に着目すると，必ず ROP コードが実行された後で，暗号化されたシェルコードを復号するための復号コードが動作する．ROP コードは DEP を回避して復号コードに実行権限を与える目的に使われるためである．すなわち ROP コード部は復号コード以降に含めることができず，シェルコードの暗号化の対象にならない．本提案手法ではこの点に着目している．

図 3.3　高度な攻撃の流れ

3.3　関連研究

3.3.1　悪性文書ファイル検出

悪性文書ファイルの検出に関する先行研究を示す．いずれもファイルを実行せずに検出する静的解析手法を用いたものである．三村らは悪性文書ファイルに埋め込まれた実行ファイル（マルウェア本体）を自動抽出する Handy Scissors を提案している [37]．複数のエンコード方式への対応や総当たり方式による鍵の探索により実行ファイルの抽出を行う．大坪らの o-checker は Microsoft 文書ファイルのサイズや構造に関する情報を検査することで悪性文書ファイルを検知する手法を提案している [38]．両手法ともにアンチウイルスソフトよりも高い検出性能を持っていると報告されている．鍵の探索機能を持つツールとして OfficeMalScanner がある [39]．鍵の探索やエンコード方式のバリエーションは

HandyScissors に劣るが OfficeMalScanner の優位点はシェルコード検査機能を持っている点である．いずれの方式も ROP コード検出の観点はなく，我々の提案手法とは異なる．

3.3.2　ROP コードの動的解析

次に ROP コードの検出分野での先行研究を示す．いずれもホスト側に実装し動的に検出する手法であり，静的に ROP コードを検出する先行研究はない．ROP を防ぐには関数が呼ばれてリターンした場合に適切に呼び元のアドレスの次のアドレスに戻っているかをチェックすればよい．これは従来から提案されている方式で，Davi らの ROPdefender では，shadow stack という領域を本来の stack 領域とは別にもうける．この領域を用いリターンアドレスを記録し適切にリターンしているかをチェックする．この方式で ROP の検出は可能となるが，shadow stack の管理のオーバヘッドが大きくなり性能に影響が出る [40].

Pappas らの kBouncer では，最近の Intel プロセッサで提供される機能である LBR（Last Branch Recording）を用いて ROP コードを検出を実現している [41]．LBR には過去 16 回分と限られはするが直近の分岐情報が記録されている．これの利点は LBR は CPU が提供するレジスタであるため，この記録にかかる性能劣化はゼロと見なすことができる点である．この LBR の履歴を活用し ROP コードを検出する．具体的には，攻撃者は最終的に API コールやシステムコールを目的とするのでその時点で過去の分岐履歴をみて正常なコードか ROP コードかを判定する．LBR の制約から履歴は 16 個に限られるが，Pappas らの調査結果では，API コール，システムコールを目的とした ROP gadget は通常 10 個程度であるため問題ないと結論づけている．さらに Cheng らは，ROPecker は，LBR に加え，さらにスライディングウインドウやコンディションフィルタリング等を用い kBouncer を改良している [42].

一方で，動的解析である kBouncer や ROPecker の限界も示されている．Carlini らは，これらの方式が，ret 命令の戻りアドレスの 1 つ手前の命令が call 命令かどうかをチェックしている点に着目し，そのような ROP チェックを回避できる gadget をみつける等，代表的な 3 つの手法を提案し，動的解析検出を回避できることを示した [43]．また Goktas らは，kBouncer や ROPecker が ROP 判定閾値である個々の ROP gadget のサイズや ROP gadget のチェイン数を経験則により固定的に決めていることに着目し，その閾値を超えない ROP コードを構成することにより検出の回避を成功させている [44].

3.3.3 ROP コードの静的解析

Tanaka らは，攻撃コードとして ROP 技法が用いられる際の，ROP コードの目的や動作順序に着目し，ROP 技法で用いられるアドレスを特徴文字列として蓄積しておき，パターンマッチによる検出を行うことを提案している．攻撃ツール Metasploit[45] に限定した評価ではあるが約 84% の ROP 攻撃を検出可能であった [46].

検体詳細解析の現場での，MS-Office 文書ファイルの動的解析では安全面の考慮に加えパッチレベル等環境を変化させ攻撃が発動する環境を探す必要がある．さらに ROP を含む検体の場合，動的解析では環境のバリエーションが増加し，さらに解析が困難となる．文献 [40, 41, 42] の ROP 防御機構環境を用いた場合でも解析コストは同様に高い．これは ROP がメモリ状態に強く依存することから，正確に ROP が動作するスタックやヒープの状態を再現する必要があるためである．このような正確なメモリ状態構築には，解析対象の文書ファイルだけではできないケースも多く，他の適切な DLL 等の読み込みが必要になる等解析を困難なものとする．

3.4 提案方式の着眼点

本提案では，ホスト側の防御機構を突破する ROP コードを静的に特定することで，悪性文書か否かの判定を可能とする手法を提案する．本提案では図 3.3 に示すように ROP コード部はシェルコードの暗号化ルーチンの外部に置かれることに着目する．ROP コードは復号コード自体に実行権限の付与する役割を持つため ROP コードはシェルコードの暗号化ロジックの内部に入らない．このため ROP コード部はシェルコードのようにコード実行によって暗号化を解く必要がなく静的な解析が可能である．一般に静的解析は攻撃コードを実行しないため安全に解析できる利点があり，文献 [37, 38, 39] の手法が提案されているが，ROP コードに着目した静的解析の提案例はない．

3.4.1 各コード部の考察

3.2.6 章および図 3.3 で示した各攻撃コードが組み込まれた悪性文書ファイルの一例を図 3.4 に示す．

これは，3.6.2 章に示すような実際の悪性文書ファイルを調査して得られた代表例である．いずれかのコードを検出できれば悪性文書ファイルと特定できるためコード部分ごとに考察を行う．

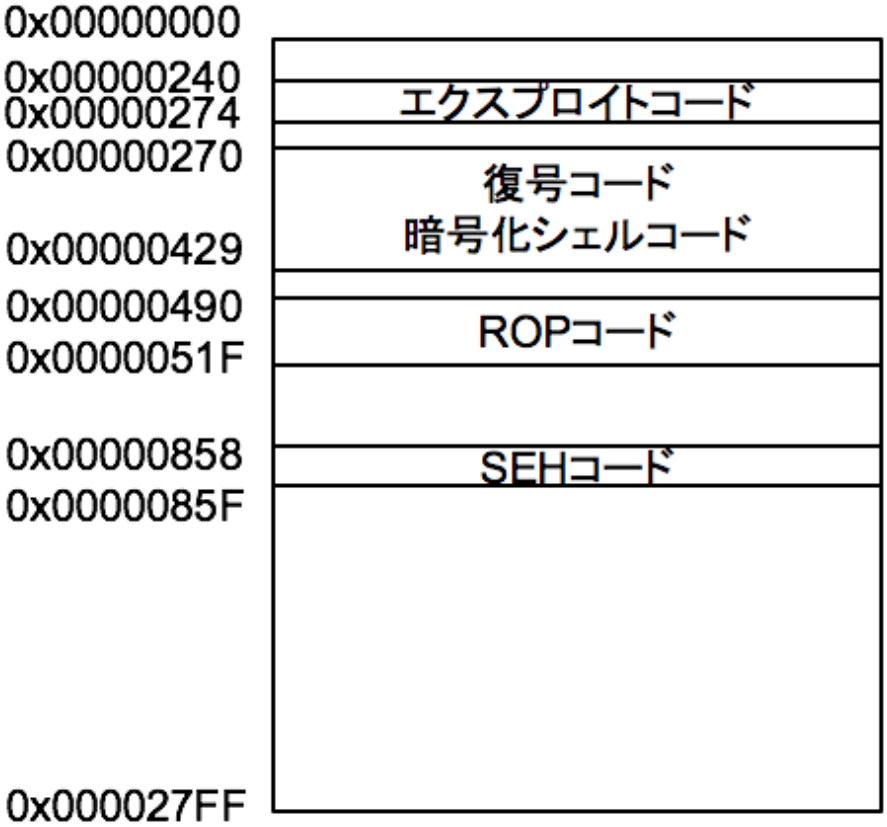

図 3.4　悪性文書ファイルの例

- エクスプロイトコード部

脆弱性を発動させるために特徴が現れやすく暗号化は難しいため特定がしやすいが，ゼロデイのように未知の脆弱性の場合は特定ができない．

- ROP コード部

DEP 回避を目的とする．後続する復号コード部の外に配置され，復号コードに実行権を付与する．復号コード部の外に配置されることからシェルコードの暗号化対象とはならない．3.4.2 章に示す方法で一般に流通する攻撃コードを調査したところサイズは平均 100 Byte 程度であった．

- SEH コード部

SEH コードは Windows の例外処理機構である SEH（Structured Exception Handling）を悪用するコードであり安定して後続のコードを実行されること目的とする．SEH コードは ROP コード部と同様に復号コードの外側に配置され，シェルコードの暗号化の対象にはならない．調査結果，コードサイズは平均 10 Byte 程度であった．

- 復号コード部

暗号化シェルコードの復号を目的とする．XOR 等の論理演算を用いる単純なコードの場合は，数十 Byte 程度であった．ポリモーフィックコードを生成するエンコーダが用いられた場合，さらにサイズが大きくなった．

- シェルコード部

シェルコードはいくつかの共通した特徴を持つ [47]．その中でも多くのコードは API 関数アドレスの自己解決を行うために PEB（Process Environment Block）を参照するため判定が可能である．しかし容易に暗号化や難読化が行われ，特に複数回暗号化を行う手法であるマルチエンコーディングが行われると検出が困難となる．

上記の考察より，脆弱性に依存せず，暗号化や難読化にも影響されない部分は，ROP コード部，SEH コード部，復号コード部である．SEH コード部や論理演算ベースの復号コード部はサイズが小さく特徴をとらえるのは容易ではない．エンコーダベースの復号コード部はそのサイズから特徴をとらえられる可能性はあるが，本提案では ROP コード部を対象とした．

3.4.2　ROP コードの内部構成

ROP コードは 3.2.3 章で示したように実行権限のあるメモリ領域のコード部分をつないで利用することで DEP を回避して任意のコードを実行可能とする技法である．一般に流通する攻撃コードを，次に示す方法で調査した結果，攻撃コードとして ROP 手法が用いられる場合，共通した目的があることが判明した．調査方法は，最初に，Metasploit の攻撃コードから，Windows プラットフォーム用のすべての攻撃コードのうち，ROP が用いられるものを抽出し（CVE 番号ユニークで約 70 個あった）目視による解析を行った．ここで Windows プラットフォームすべてを対象としたので MicrosoftOffice 以外の脆弱性も含まれる．次にこの攻撃コードの CVE 番号から exploit-db[48] の検索機能を使い，exploit-db に登録のある Metasploit 以外の攻撃コードを入手し同様に解析を行った．また，インターネットから入手できる悪性文書検体や独自に入手した悪性文書検体で ROP が含まれるものも解析対象とした．この調査から，ROP コードの目的は，ROP を利用して自由度の高い任意のシェルコードを書くのではなく，後続するコード領域に実行権限付与するのみであるという事実を発見した．

Windows では関数をコールし実行権限を付与する．実行権限を付与可能な関数例を表 3.1 に列挙する．またその 1 つである VirtualProtect 関数の引数について表 3.2 に示す．次に VirtualProtect 関数を用いた ROP コードがスタックに積まれた状態例を図 3.5

に示す．

表 3.1　実行権限付与関数例

実行権限付与関数例		
VirtualProtect	VirtualAlloc	NtSetInformationProcess
HeapCreate	SetProcessDEPPolicy	WriteProcessMemory

表 3.2　VirtualProtect 関数の引数

名前	説明
lpAddress	アクセス権限を変えたいページ領域のアドレス
dwSize	領域のサイズ
flNewProtect	アクセス権限．実行権を付与するには 0x40(PAGE_EXECUTE_READWRITE)

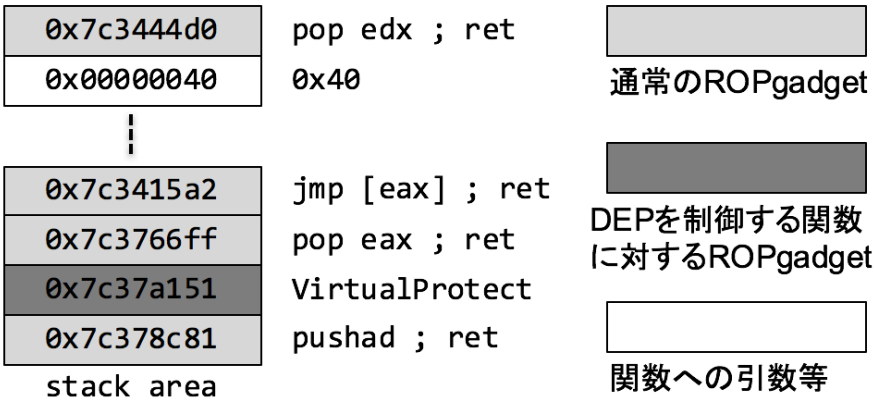

図 3.5　ROP コードが積まれたスタック

図 3.5 で DEP を制御する表 3.1 に示す関数のアドレス（図では VirtualProtect 関数）に関する ROPgadget を濃いグレー，これらの関数に適切な引数等を準備するのに用いられる ROPgadget を通常の ROPgadget として薄いグレー，関数等への引数を白で示す．攻撃者はスタックにこの 2 種類の ROPgadget および引数等を適切に積んでおき，ROP コードを実行し，シェルコード配置エリアのメモリ領域に実行権限を付与し DEP を回避する．DEP が回避されると復号コードの実行等の処理が開始する．

攻撃者は安定して攻撃を成功させるためにアドレスが固定値として存在する ROPgadget を用いようとする．3.2.4 章で示した ASLR が機能している場合，固定値として用い

ることが困難となるが 3.2.5 章で示した回避方法がある．攻撃コードを調査したところ，ROPgadget のアドレス数は限定的であり，3.3.3 章で示した Tanaka らの従来方式 [46] は，この点を利用して検出を行う．

3.5 提案方式

3.5.1 非シグネチャ依存方式

3.3.3 章で示した従来方式 [46] は，攻撃者が用いる既知の ROPgadget のアドレスを特徴文字列として蓄積しておき検出する方式であり，未知のアドレスは検出できない．ただ攻撃ツール Metasploit の攻撃コードを利用した攻撃も多数観測されており脅威であることから [49]，Metasploit 攻撃コード検出に効果が示されている従来方式 [46] でも一定の効果はあると考えられる．しかし，未知の ASLR 非対応 DLL のアドレスを用いる攻撃手法，または，ASLR 対応 DLL でも，特定の DLL のベースアドレスを動的に攻撃コード内で取得して用いる攻撃手法 [36] に対してはあらかじめシグネチャを作れないため検出できない．特に文献 [36] の手法は ASLR 非対応 DLL を必要としないため攻撃者の立場からすると有効である．ASLR 非対応 DLL は悪用されることが知られるようになると，ASLR 対応した修正プログラムがリリースされるためである．このような新たな攻撃手法に対抗するため，提案方式では，従来方式のように特定の DLL 等の固定アドレスのシグネチャに依存しない方法で ROP コードの検出を目指す．

提案方式を説明するために，ASLR 非対応 DLL から作られた ROP コードを図 3.6 に示す．これは Metasploit から取得した JRE1.6 アプリケーションである msvcr71.dll から作られた実際の ROP コードである．msvcr71.dll は ASLR 非対応 DLL のためベースアドレスはつねに固定で，メモリ空間 0x7C340000 から 0x7C396000 にロードされる．攻撃者はこの空間内から ROPgadget を探してそれらを組み合わせて ROP コードを作成する．このため，作られる ROPgadget はすべて上位バイトが 0x7C3 となる．この ROPgadget が連結された固まりである ROP コードは図 3.6 のようになる．したがって図のグレーで示すように上位バイトが共通して 4 バイト周期で現れるという特徴がある．図の各バイト列はリトルエンディアンで配置されることに注意する．未知の ASLR 非対応 DLL や ASLR 対応 DLL から作られた ROP コードの場合，ベースアドレスは推測できず，対応したシグネチャは作成できないが，特定のメモリ空間に DLL はロードされるため，図のグレーで示すような 4 バイト周期で同一文字列が現れるという特徴は共通する．つまり提案手法の要点は ROPgadget は特定の DLL のアドレス空間から作られ

ASLR 対応か非対応かにかかわらず，上位バイトが等しい ROPgadget が固まりとなって ROP コードとして文書ファイル内に含まれることに着目し検出を行う．この周期的に現れる特徴をベースに経験則から次節以降で示す複数の条件や閾値を決定し，シグネチャに依存しない方式で ROP コードを判定する．

```
664C367C664C367CDC4E347C10020000
051E357CE3363357CFFFFFFFF55523347C
8E21357C37593347C00000040B11E357C
B9C5367C671E397C582E347C02D2347C
F4F8347CA2153347CDC4E347C51A1377C
818C377C305C347C
```

図 3.6　ROP コードの例

3.5.2　提案方式：全体フローチャート

図 3.7 に提案方式の全体フローチャートを示す．判定対象のファイルは文書ファイルである doc 形式，xls 形式，ppt 形式，rtf 形式である．バイナリ形式でファイルオープンする．rtf 形式の場合，攻撃コードは ASCII 文字エンコードされているためデコードを行う．ここで 1 ワードを 4 バイトと定義する．図 3.7 の (※ 1) で 1 ワードを読み込み，後述する PRE check 部で判定を行い，No であれば，offset に 1 バイト加算し，再度，図 3.7 (※ 1) で 1 ワードを取り出し同様の判定を行う．PRE check 部で Yes となった場合，図 3.7 (※ 2) で offset から 25 ワード取り出し，BASIC check 部で判定を行う．No の場合は offset に 1 バイト加算し同様に頭から処理を行う．Yes の場合，次の FINAL check 部での判定を行い，Yes であれば ROP コードが見つかったとして ROP コード部のオフセット値を出力して終了，No であれば，同様に頭から処理を行う．ここで 25 ワードの値は，実際の攻撃コードを観察し得た知見から決定した．

3.5.3　提案方式：PRE check 部

PRE check 部では先頭 1 ワードが ROP コードであるかどうか判定を行う．表 3.3 で示す条件のいずれかに該当する場合 ROP でないと判断し図 3.7 で No に進む．表 3.3 の条件 P1 は，悪用される DLL はユーザ空間にロードされる経験則から，ROPgadget と

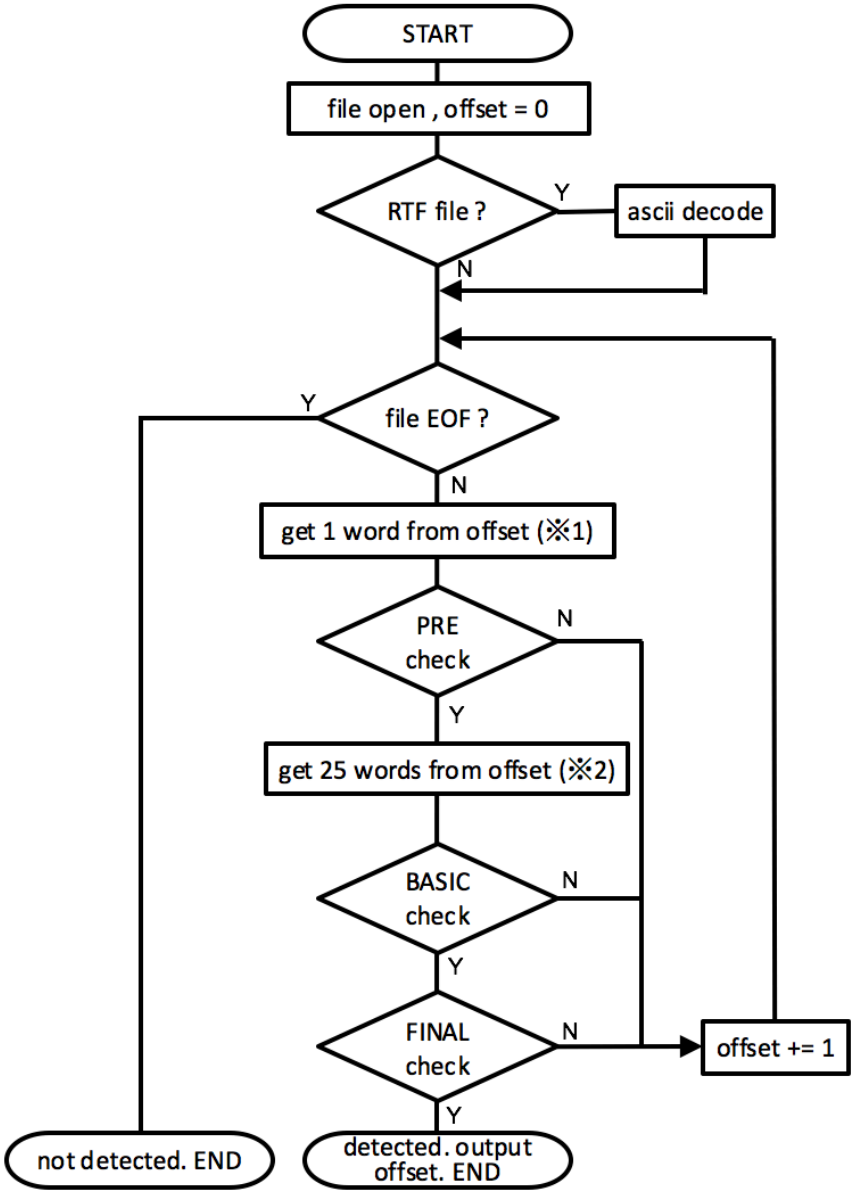

図 3.7　全体フローチャート

して利用されるメモリ空間は 0x00000000 から 0x7FFFFFFF の範囲となり，この範囲外であれば ROP コードと判断しない．条件 P2 は，本方式の制約事項となるが，上位 1 バイトが 0x00 で始まるメモリ空間の ROP コードの判定は，誤検出を誘発することから行わないこととした．つまり 0x00000000 から 0x00FFFFFF のメモリ空間で ROP コードを組み立てられた場合，本提案手法では検出できない．これは MS-Office 文書内に 0x00 で始まる ROP コードに類似したコードが存在するためである．MS-Office 文章は CFB

形式 [50] であり，その仕様から一部の 0x00 を含む固定長フィールドの繰返しが影響すると推測される．この類似部分を判定する他の条件が見つけられるのが理想ではあるが，この制約空間（0x00000000 から 0x00FFFFFF）は ROP 技法が用いられるユーザ空間全体（0x00000000 から 0x7FFFFFFF）のなかの 1/128 の割合であるため，今回は制約事項としても本提案手法の利用価値への影響は少ないと判断し本研究の対象外とした．条件 P3 は，下位 2 バイトがともにゼロとなる場合で，関数のベースアドレスとなる値を ROPgadget の起点するケースは皆無と考え可能性なしとする．

表 3.3　Pre check 条件

条件 No	条件	例
P1	値が 0x7FFFFFFF より大きい	0x953B235F
P2	値が 0x01000000 より小さい	0x0001F235
P3	値が 0x10000 で割り切れる	0x1B420000

3.5.4　提案方式：BASIC check 部

BASIC check 部では，攻撃コードを観察した知見から，該当 25 ワードが，ROP コードの可能性があるかの判断を行う．表 3.4 に示す条件のいずれかに該当する場合 ROP の可能性なしとして図 3.7 で No に進む．いずれも ROP コードを組み立てるうえでの障害となる条件であり除外して問題ないと考えられる．

表 3.4　Basic check 条件

条件 No	条件	閾値	例
B1	1 バイト毎に同一の値が閾値以上出現	15	0x0B340B7A, 0x0B880B55
B2	各ワードの上位 2 バイト目が 0x00 となるワードが閾値以上出現	12	0x3D001283, 0x2200DEAD
B3	各ワードの下位 1 バイト目が同一となるワードが閾値以上出現	12	0x4D12836D, 0x458E546D
B4	各ワードの下位 1 バイト目が単調増加するワードが閾値以上出現	6	0x340B2E01, 0x9B6EB702, 0x831E6603
B5	各ワードの下位 1 バイト目が 0x00 となるワードが閾値以上出現	6	0x3D321200, 0x22DE4500

3.5.5　提案方式：FINAL check 部

FINAL check 部では，該当 25 ワードが ROP コードと断定できるかどうかの最終判断を行う．図 3.8 にフローチャートを示す．先頭 1 ワードと後続する各ワードとの差分を diff 配列に格納する．次に図 3.8 (F1) で diff 配列で値がゼロのものや同じ値のものは ROP の組み立てに非現実であるため，ROP コードではないと判断し，diff 配列から除外する．アドレスの差分が同じ ROPgadget を複数用いて ROP コードを組み立てることは，CTF [*1]等の競技において問題作成者が意図的に制約として課すケースがある．しかし，ROPgadget が得られるメモリ空間が広い必要があり，現実世界では利用可能なメモリ空間は狭いため，実攻撃コードとして経験上目にすることがなく，ROP コードの可能性は低いとして除外している．次に図 3.8 (F2) で diff の値が値 0x100000 以上のもの，つまり個々の ROPgadget のアドレスが離れすぎているものは ROP として非現実であるとして除外する．この閾値とした値については，ROP に用いられる代表的な DLL を調査した結果を表 3.5 に示し，この値を参考に決定した．図 3.8 (F3) では，各 ROPgadget のアドレスが逆にあまりにも近い場合は ROP を組み立てるのが困難になり非現実であるので，閾値 10 個より大きい場合 ROP ではないと判断する．具体例として，F3 条件に該

[*1] CTF（Capture the flag）：コンピュータセキュリティ技術を競う競技

当する ROP コードの例は，0x12345611，0x12345633，0x123456ab，等で下位 1 バイトのみ変化する異なる ROPgadget が 10 個以上用いられる ROP コードである．5.1 節で説明したように，攻撃者は特定の固定アドレスのメモリ空間（図 3.6 の例で，msvcr71.dll が読み込まれる 0x7C340000 から 0x7C3960000）をできる限り活用して ROPgadget を見つける．このとき，たとえば 0x7C340000 から 0x7C3400FF の中からのみ ROPgadget を見つけ DEP を解除するコードを組み立てるのはほぼ不可能であると経験則から判断し，この条件を入れた．最後に図 3.8 (F4) で，残った diff 配列の要素数が閾値 8 個より大きい場合 ROP コードと判定する．つまり，F4 条件は最終的に，これまでの処理で ROPgadget と考えることができるワードが閾値個以上のこった場合，該当コード部分が ROP コードと最終判定している．これらの閾値も経験則によって決定した．

表 3.5　ROP に用いられる代表的な DLL 例

名前	開始アドレス	終了アドレス	サイズ
msvcr71.dll	0x7c340000	0x7c396000	0x56000
hxds.dll	0x51bd0000	0x51ca7000	0xd7000
Msvcrt.dll	0x77c10000	0x77c68000	0x58000

3.6　実験

3.6.1　ツール試作

提案方式についてツールを試作し実験を行った．ツールの仕様は以下のとおりである．

- OS：Mac OS X v10.9 Mavericks
- 言語：Python 2.7.5
- 対応ファイル形式：doc, xls, ppt, rtf

3.6.2　実験検体

手動解析で ROP コードが埋め込まれていることが確認できた検体を準備した．内訳は，表 3.6 に示すインターネット [45, 51, 48, 52] で入手できた in the wild 検体含む約 150 個の悪性文書ファイルから ROP を含むもの 6 検体，独自ルートで確認した悪性文書ファイルから 8 検体である．悪性文書ファイル自体の流通が少なく，また，ROP コード

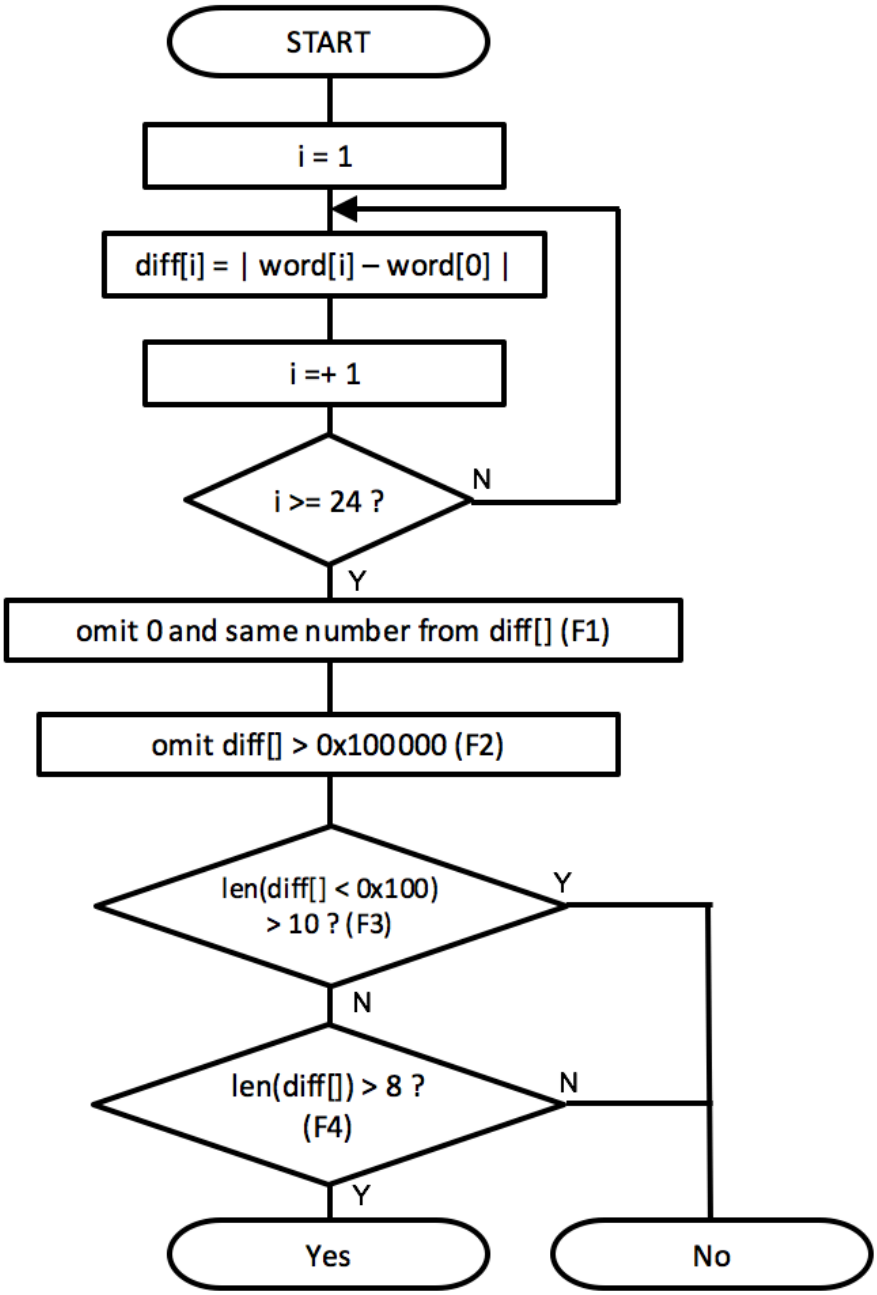

図 3.8　Final Check 部フローチャート

が埋め込まれている検体は比較的新しい検体に限られることから，これ以上の検体母数を増やすことは困難であった．また一部のファイルではファイル名を MD5 ハッシュ値の先頭 5 文字のみの標記としている．これらの検体はあらかじめ解析を行い CVE 番号が判明したものは標記した．これは異なる CVE 番号でも類似した ROP コードが用いられる場

合があるのではないかという仮説を検証するために行った.

表 3.6　ROP 検体

検体 No	ファイル名	CVE	入手元	in the wild
1	msf11_006.doc	CVE-2010-3970	Metasploit	
2	44.doc	CVE-2010-3970	contagio	✓
3	msf12_027.doc	CVE-2012-0158	Metasploit	
4	cve_.doc	CVE-2010-3333	exploit-db	✓
5	msf14_017.doc	CVE-2014-1761	Metasploit	
6	e378_.bin	CVE-2014-1761	malwr	✓
7	d309e_	CVE-2012-0158	独自	✓
8	c2b68_	CVE-2012-0158	独自	✓
9	cf548_	CVE-2014-1761	独自	✓
10	006fc_	CVE-2012-0158	独自	✓
11	47a92_	不明	独自	✓
12	6653a_	CVE-2012-0158	独自	✓
13	b61ee_	不明	独自	✓
14	eda67_	CVE-2012-0158	独自	✓

3.6.3　False Positive 検査

正常なファイルを誤判定しないことの確認として，正常なファイルを，doc 形式：15 ファイル，rtf 形式：15 ファイル，xls 形式：10 ファイル，ppt 形式：10 ファイルの合計 50 ファイルを主にインターネットからダウンロードし正常検体として用意した．試作ツールで検査を行い ROP コード判定されないことを確認した.

3.6.4　実験結果

提案方式を実装した試作ツールの結果のほかに，関連研究である 3.1 節の o-checker, OfficeMalScanner，およびアンチウイルスソフトである ClamAV での検出結果を表 3.7 に示した.

試作ツールは No5，6，9 を除く検体について，ROP コード判定を行い，ROP コードのオフセット値を出力した．また 3.5.5 章の F4 条件で残ったワードを参照することで

ROPgadget のアドレス情報を得ることができた．例として，検体 No3 および No4 でから得られた ROPgadget のアドレス情報を 3.6.5 章の図 3.8 に示す．オフセット値が正しいかは各検体をバイナリエディタで閲覧し確認した．次に ROPgadget のアドレス情報が正しいかの確認は動的解析が必要となる．実際にはデバッガを用い，各検体を動作させ，得たアドレス情報が，メモリ領域上で 3.2.3 章の図 3.2 で示した ROP コードを構成する意味のある値であり確かに ROPgadget であることを確認した．さらにこれらのコードを観察したところ，3.4.2 章の図 3.5 で示した特定の DLL アドレス空間から構成される DEP を解除する目的とした ROP コードであることが分かった．このことからも特定 DLL アドレス空間に着目した 5 章で示す提案方式の有効性が確認できた．

一方で，検体 No5，6，9 は試作ツールで判定できなかった．これらの検体は CVE-2014-1761 で脆弱性を発動させるためにエンコードしておく必要があり図 3.6 に示すようなバイトコードとは形式が異なるため検出できなかった [53]．エンコード工程の詳細を表 3.9 に示す．工程 1 のデータがオリジナルデータで工程 7 がエンコード後のデータである．エンコード対象は，攻撃コードすべて（エクスプロイトコード，ROP コード，復号コード，シェルコード）で，エンコードなしでは，攻撃者が意図する脆弱性が発動しない．このようにエンコードとは暗号化や難読化と異なり一定のルールに基づいて型変換することを示す．なお，このエンコード下でも ROP コード部はシェルコードの暗号化ロジックの外に配置されるというルールは成り立っている．

o-checker は，No1，2，3，5，7，10，12，14 が検出できなかった．この理由として，ファイル構造に問題がない形で各種攻撃コードが埋め込まれていたからであった．これは大坪らも CVE-2010-3333 として言及している [54]．本実験では CVE-2010-3970，CVE-2012-0158 の検体が検出不可となったが，これは検体の作りに依存すると考えられる．一方，試作ツールで検出できなかった No6，9 を o-checker はファイル構造の特徴を使って検出することができた．

OfficeMalScanner は，No4，6，8，9 についてシェルコード検出機能により検出できた．検出できなかった No1 の検体は，metasploit のシェルコードエンコードオプションで bloxor を指定して作成したものである．これを外したところ OfficeMalScanner のシェルコード検出機能で FS レジスタ (0x30) へのアクセスコードでとらえることができた．これは多くのシェルコードで見られる特徴であり暗号化されていないコードをとらえるのに有効な手法である．ただし最近のシェルコードは暗号化がされていることからこの手法での検出には限界がある．

最後に ClamAV については，No6，9 の比較的新しい脆弱性を検出できたにもかかわらず，他の古い脆弱性の検出はできないという結果になった．パターンマッチング方式での

検出は難しいことが推測される．

表 3.7　実験結果

検体 No	試作ツール	o-checker	OfficeMalScanner	ClamAV
1	✓	NA	NA	NA
2	✓	NA	NA	NA
3	✓	NA	NA	✓
4	✓	✓	✓	✓
5	NA	NA	NA	NA
6	NA	✓	✓	✓
7	✓	NA	NA	NA
8	✓	✓	✓	NA
9	NA	✓	✓	✓
10	✓	NA	NA	✓
11	✓	✓	NA	NA
12	✓	NA	NA	NA
13	✓	✓	NA	NA
14	✓	NA	NA	NA

3.6.5　ROP コードの解析と汎用性考察

提案手法の有効性をさらに考察するために，試作ツールで ROP 検出可能であった検体の ROP コードの解析を行った．表 3.8 に検体 No3 および No4 で用いられた ROPgadget を示す．それぞれユニークな ROPgadget は 14 個使われており, 両者で共通しているものをグレーで示す．表 3.10 に各検体で用いられた ROPgadget のユニークな個数を示した．表 3.11 に検体どうしで共通の ROPgadget が使われた割合を示す．分母の数字は比較 2 検体合計のユニーク ROPgadget 数であり，分子は比較 2 検体で共通した ROPgadget 数である．たとえば，表 3.8 の検体 No3 と No4 で分母は 16，分子は 12 となり，表 3.11 には 12/16 と標記される．表 3.11 より No1 と No2，No3 と No10，No7 と No12 と No14 は用いられた ROPgadget が完全に一致していることが分かる．また No3 と No4，No4 と No10，No11 と No13 は，それぞれ 12/16（75%），12/16（75%），11/14（約 79%）と高い割合で同一の ROPgadget が用いられていることが分かる．ここで，表 3.6 より

表 3.8　使われた ROPgadget の例

検体 No3	検体 No4
0x3f2233cc	0x3f2233cc
0x3f2b745e	0x3f2b745e
0x3f2cb9e0	0x3f2cb9e0
0x3f2d59df	0x3f2d59df
0x3f2f18cc	0x3f2f18cc
0x3f389ca5	0x3f389ca5
0x3f39795e	0x3f39795e
0x3f39afcf	0x3f39afcf
0x3f39cb44	0x3f39cb44
0x3f39cb46	0x3f39cb46
0x3f3b3dc0	0x3f3b3dc0
0x3f3b3dc4	0x3f3b3dc4
0x3f10115c	0x3f398267
0x3f2cb9e1	0x3f3a16de
計 14 個	計 14 個

No3 は CVE-2012-0158 脆弱性を，No4 は CVE-2010-3333 脆弱性を利用するもので，用いられる脆弱性は異なるが，用いられる ROPgadget は高い割合で一致していることが分かる．これは，冒頭で述べたように，ゼロデイ脆弱性で，その脆弱性に対するシグネチャが存在せず対処ができない場合でも，異なる脆弱性でも共通して使われる可能性が高い ROP コードを検出することで，ゼロデイ対策になる可能性を示している．

表 3.12 に，表 3.11 での ROPgadget の類似度を参考にして，各検体の ROP コードを分類した．結果，5 つのタイプに分類できた．これは，3.6.2 章で示したように ROP 検体の多数収集は困難である状況下で限られたサンプル数ではあったが，本提案手法は，異なる 5 つのタイプの ROP コードを検出できたことを示しており，有効性が確認できた．

3.7　議論

本提案方式の特徴は以下である．

- シグネチャに依存せず ROP コードを検出

表 3.9　エンコード詳細

工程	データ	説明
1	0x12345678	オリジナルデータ
2	0x1234 0x5678	上位・下位バイト分離
3	0x5678 0x1234	上位・下位バイト反転
4	22136 4660	10 進数変換
5	-18604	0x1000 から工程 4 の値を引く
6	18040 564	絶対値取得
7	?\u-18040 ?\u-564	“ ?\u-“ を先頭に付加

表 3.10　ROPgadgets のユニーク数

No	1	2	3	4	5	6	7	8	9	10	11	12	13	14
数	10	10	14	14	NA	NA	13	11	NA	14	13	13	12	13

表 3.11　同一 ROPgadgets が使われた割合

No	2	3	4	5	6	7	8	9	10	11	12	13	14
1	10/10	0/24	0/24	NA	NA	0/23	0/21	NA	0/24	0/23	0/23	0/22	0/23
2	-	0/24	0/24	NA	NA	0/23	0/21	NA	0/24	0/23	0/23	0/22	0/23
3	-	-	12/16	NA	NA	0/27	0/25	NA	14/14	0/27	0/27	0/26	0/27
4	-	-	-	NA	NA	0/27	0/25	NA	12/16	0/27	0/27	0/26	0/27
5	-	-	-	-	NA	NA	NA	NA	NA	NA	NA	NA	NA
6	-	-	-	-	-	NA	NA	NA	NA	NA	NA	NA	NA
7	-	-	-	-	-	-	0/24	NA	0/27	2/24	13/13	1/24	13/13
8	-	-	-	-	-	-	-	NA	0/25	1/23	0/24	1/22	0/24
9	-	-	-	-	-	-	-	-	NA	NA	NA	NA	NA
10	-	-	-	-	-	-	-	-	-	0/27	0/27	0/26	0/27
11	-	-	-	-	-	-	-	-	-	-	2/24	11/14	2/24
12	-	-	-	-	-	-	-	-	-	-	-	1/24	13/13
13	-	-	-	-	-	-	-	-	-	-	-	-	1/24

表 3.12　ROPgadgets による分類

No	1	2	3	4	5	6	7	8	9	10	11	12	13	14
タイプ	Ⓐ	Ⓐ	Ⓑ	Ⓑ	NA	NA	Ⓒ	Ⓓ	NA	Ⓑ	Ⓔ	Ⓒ	Ⓔ	Ⓒ

本方式により，高度化する攻撃手法に対抗するためシグネチャに依存せず ROP コードを検出し，ROP コードのオフセット値や ROPgadget のアドレス情報を得ることができる．ROP コード部の目的は DEP の解除でありシェルコードの暗号化ロジックの外側に配置される点に着目し検出する．ROP コードは新しい脆弱性において多用される点 [32],

および，同一 ROP コードが複数の脆弱性を攻撃するために用いられる傾向がある点から，未知の脆弱性を突くゼロデイ検体検出に効果が見込める．

- ツールによる静的解析検出

本方式は，他の関連研究では行われていない静的解析による ROP コード検出を行う．一般に，文書ファイル型検体の解析には，動的解析に加え詳細な機能を知るための人手による静的解析が行われる．本方式はツールとして動作し，解析者に有益な出力情報を提供する．

- 既存方式との併用

本提案方式の検出方法は既存方式と独立しているため組み合わせて用いることができ，攻撃が高度化するなか多層防御の観点でも貢献できる．

一方，本提案方式の課題として以下がある．

- エンコードへの対応

CVE-2014-1761（検体 No5，6，9）では ROP コードがエンコードされていたため，本提案方式で検出ができなかった．ただ脆弱性を発動させるためのエンコードは難読化とは異なり，一定の制約に基づいて型変換されるため，検出できる可能性はあるが，アルゴリズムは図 3.7 とは異なるものを考える必要がある．

- 64 bit 対応

今回は windows32 bit 環境を想定したが，類似アルゴリズムで 64 bit 対応も可能と考えられる．ただ，実際に 64 bit の ROP コードの流通を確認できていない．

- 未知 DLL 名の判定

今回は実装していないが，既知の DLL から作られた ROP の場合は DLL 名を判定することが方式上，可能である．しかしながら未知 DLL の場合は，アドレスは分かっても DLL 名は判定できない．DLL 名を判定するためには，候補 DLL を読み込み動的解析を繰り返す等の解析が必要であり，その自動化も課題である．

- 制約事項の解消

今回上位 1 バイトが 0x00 から始まる ROP コードの判定は誤検出を誘発する可能性があることから行わないこととし，本方式の制約事項とした．これを判定できる条件，たとえば，今回は利用しなかったが DEP 制御関数への引数等の特徴文字列等を併用する，もしくは，3.4.1 章で示した復号コード部に現れる特徴文字列を併用する等により制約事項を解消できる可能性がある．

- 閾値を回避する検体

本提案方式では入手可能な ROP 検体の解析結果に基づく経験則から閾値を選定してい

る．意図的に閾値を回避するような検体の場合検出できない．

- 実験検体数の増加

悪性文書ファイルの検体は個人・企業の秘密情報が含まれるため一般に流通しにくい点も研究を進めるうえでの課題である．

3.8　本章のまとめ

3 章では，対策が非常に困難であるゼロデイ攻撃対策を視野に入れ，標的型攻撃で多用される悪性文書ファイルを静的解析により安全かつ効率的に検出する手法を提案した．本提案手法では，文書ファイルに埋め込まれた ROP コードの検出により悪性文書ファイルを検出する．これは 1.2 章で述べた，静的解析分野の中で，プログラムコードを分析する研究分野に位置付けられ，準直接的なアプローチである．ゼロデイを含む攻撃で昨今多用される ROP コードの目的が共通している点やシェルコード自体の暗号化対象とならない点を利用し検出を行う．提案方式を実装し他のツールと比較した実験結果と課題を示した．

第 4 章

効果的なブラックリスティングの検討

4.1 概要

インターネット利用の増加に伴い，その悪用は悩ましい問題である．ドライブバイダウンロード攻撃は未だに深刻であり，エクスプロイト・アズ・ア・サービス・エコシステムでは，マルウェアダウンロードサイトは，攻撃者にとって特に重要な役割を担う．4 章では，約 43,000 個のマルウェアダウンロード URL を約 1.5 年の長期間にわたり観測した結果を分析した．一部のマルウェアダウンロードサイトは長期間にわたり生存を続け，また消失と復活を繰り返す特徴があることを明らかにした．特に，マルウェアの変化に着目し，3 つのカテゴリを定義し解析を行った．*unchanged* カテゴリの 10% は 500 日以上生存を続けること，*changed occasionally* カテゴリの 10% は 15 回以上消失と復活を繰り返すことを明らかにした．さらに IP アドレスの変化等の他の解析も行った．3 つのカテゴリごとに攻撃者のリソースやオペレーションが異なることを明らかにした．解析に基づき，各カテゴリの特徴と効果的なブラックリスティング対策について議論する．本章は，1.2 章で述べた，マルウェアに関連する通信を特定する研究分野の中で，マルウェア感染前の対策として利用される入口系に位置付けられ，間接的なアプローチである．

4.2 関連研究

これまでにマルウェアの感染源となるドライブバイダウンロード攻撃を分析した研究は数多くある．Grier らはドライブバイダウンロード攻撃における，エクスプロイト・ア

ズ・ア・サービス・モデルの脅威を警鐘し，多くの主要なマルウェアがドライブバイダウンロード攻撃によって配布されることを示した [55]．Provos らは，10 ヶ月にわたりドライブバイダウンロード攻撃を分析し，特に，Google 検索エンジンのサーチ結果のおよそ 1.3% に少なくとも 1 つの悪性な URL が含まれることを示した [56]．Shue らは，特定の AS が悪性サイトの温床となっていることを示している [57].

Nappa らは，また，ドライブバイダウンロードの攻撃者側の運用の観点で調査を行い，攻撃サイトの 10% が 1 週間以上生存，5% が 2 週間以上生存，最も長いもので 2.5 ヶ月生存したと報告している [58]．Gross らは，悪性な挙動が見られるサイトを識別する方式を提案し，フィッシングサイトよりもドライブバイダウンロード攻撃に用いられるサイトの方が長生きであることを示し，攻撃サイトの 15% が 60 日以上生存したと報告している [59]．本研究では，マルウェアダウンロードサイトに着目して観測を行い，攻撃サイトよりも大幅に寿命が長いことがわかった.

Zeeuwen らはマルウェアダウンロードサイトからのマルウェアの変化に着目し，異なるマルウェアがダウンロードされるスケジュールアルゴリズムを提案している [60]．Eshete らは HTTP トラフィックの 30 の特徴量を機械学習に適用し ExploitKit の識別を行っている [61]．Nelms らはドライブバイダウンローのシーケンスを遡ることで悪性サイトの元を識別する方式を提案している [62]．秋山らは，ハニートークンを用い攻撃者の活動を明らかにする研究を行っている [63].

本研究は，これらの関連研究とは異なり，マルウェアダウンロードサイトの生存期間，生き返り活動，マルウェアの変化を長期に渡り観測・分析したものである．特に，後述する COC カテゴリは，その存在自体が明らかになっていなかった.

4.3　観測環境とデータセット

4.3.1　観測環境

図 4.1 及び表 4.1 に示すシステムを用いマルウェアダウンロードサイトの観測を行った．実際の OS とブラウザで構成されるハイインタラクションウェブクローラを用い，サイトを日々アクセスし，死活監視を行った．攻撃者に気づかれにくい正確な監視を目指し，複数のソース IP アドレスを用いた．図 4.2 に死活監視の具体的定義を示す．アクセスした結果，ダウンロードされたファイルが悪性であるか良性であるかにより，サイトのステイタス（活動もしくは停止）定義する．悪性判定には 4 社のアンチウイルスソフトを用い，少なくとも 1 社が悪性と判定した場合，悪性とした．4 社の選定について，これら

はいずれも日本市場で高いシェアを持つ製品である．アンチウイルス製品の幾つかは過検知が多いことが知られるが，トップマーケットシェアの製品は販売量が少ない製品よりも過検知が少ないと考え選定した．この製品選定と閾値について，4.5.2 章で考察を行う．また今回，システムの性能制約により，ファイルの悪性判定は初回ダウンロード時のみ行っている．HTTP や DNS エラーが発生して，ファイルが入手できない場合は，停止とした．

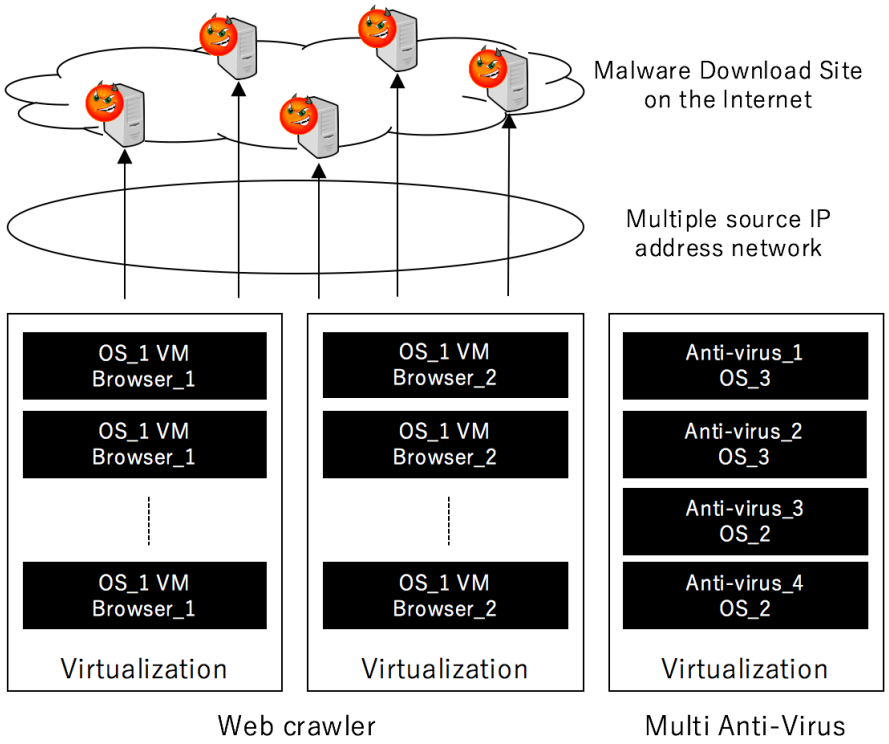

図 4.1　観測環境

この環境を用いて，1.5 年間に渡り，サイトを日々巡回した．表 4.2 に例として 4 つの URL の 12 日間の観測結果であり，URL_1 は 4,7,9 日目に活動，URL_2 は 2,6 日目に停止していることを示している．同様に表 4.3 にダウンロードされたファイルのハッシュ値の観測結果例を示す．この例では，URL_1 と URL_2 からは常に同じハッシュ値のマルウェアがダウンロードされる一方で，URL_3 からは常に異なるマルウェアがダウンロードされることを示している．また，URL_4 からは，多くの場合同じマルウェアがダウンロードされるが稀に異なるものがダウンロードされることがわかる．同様に，表 4.4 に IP アドレスの観測結果例を示す．

表 4.1　観測環境詳細

Item	Product and version
Virtualization	VMware ESXi 5.5
OS_1 (Web crawler)	Windows XP SP2
OS_2 (Anti-virus)	Windows 7 Professional SP1
OS_3 (Anti-virus)	CentOS 6.3
Browser_1 (Web crawler)	Internet Explorer 6
Browser_2 (Web crawler)	Internet Explorer 8
Anti-virus_1	Trend Micro ServerProtect for Linux [64]
Anti-virus_2	Symantec Endpoint Protection [65]
Anti-virus_3	McAfee Endpoint Protection Suite [66]
Anti-virus_4	Kaspersky Security [67]

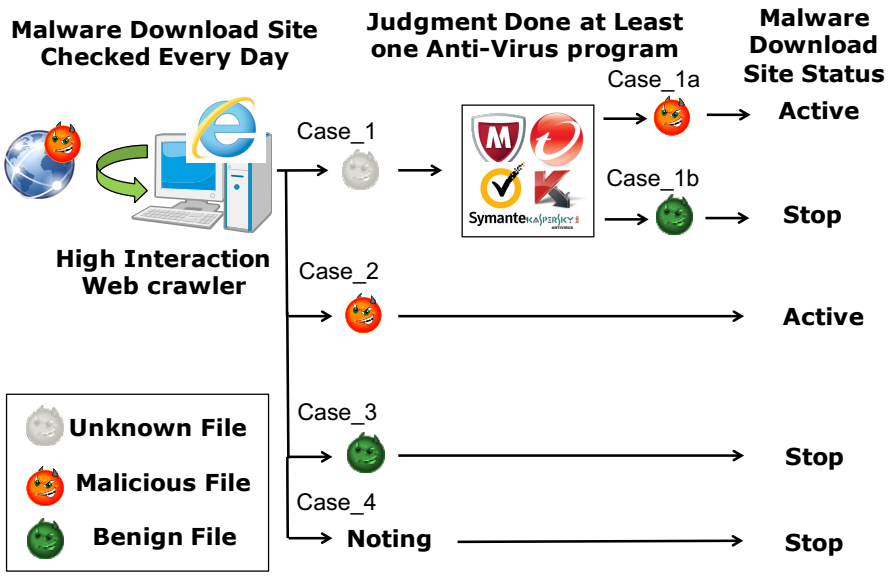

図 4.2　死活監視定義

4.3.2　データセットの特徴

観測期間は 2014/2/1 から 2015/9/30，観測総 URL 数は 43,034 個である．既知の悪性 URL の巡回に加えて，未知の悪性 URL の巡回も行った．これは一部の悪性 URL は短命で，すぐに消滅してしまうので，これだけを巡回していても十分な観測データは得ら

表 4.2　ステイタステーブルの例

DAY	1	2	3	4	5	6	7	8	9	10	11	12
URL_1	✓	✓	✓	×	✓	✓	×	✓	×	✓	✓	✓
URL_2	×	✓	×	×	×	✓	×	×	×	×	×	×
URL_3	✓	✓	✓	×	✓	✓	×	×	✓	✓	✓	×
URL_4	×	×	✓	✓	✓	×	✓	✓	✓	✓	×	×

表 4.3　マルウェアハッシュテーブルの例

DAY	1	2	3	4	5	6	7	8	9	10	11	12
URL_1	A	A	A		A	A		A		A	A	A
URL_2		B				B						
URL_3	C	D	F		G	H			I	J	K	
URL_4			O	O	O		P	O	O	O		

表 4.4　IP アドレステーブルの例

DAY	1	2	3	4	5	6	7	8	9	10	11	12
URL_1	a	b	a		a	b		b		a	b	a
URL_2		a				a						
URL_3	c	d	e		f	g			h	i	j	
URL_4			c	c	c		b	c	c	c		

れない為である．未知の悪性候補 URL はオープンフィードと呼ばれる悪性 URL が一般の Web サイトに掲載されるサイトである，

表 4.5 に，観測した総 URL 数 43,034 個の AS 名のトップ 10 を示す．データセットの信頼性を確認するために過去知見との一致を調査した．まず，これらの AS の多くはクラウドホスティングサービスを提供する業者であり，同様の知見は過去研究と一致する [58]．また，悪性サイトの数は，AS の規模よりむしろ，特定の AS に固まる傾向があることが示されている．AS の大きさを，CAIDA [68] ランキングを用いて調べたところ同様の傾向が見られた．

表 4.5　AS 名トップ 10 の詳細

NetName	ASN	CC	Count	Percent	AsRank
Amazon.com, Inc.	16509	US	16,632	37.6	3,407
Chinanet	4134	CN	11,469	26.0	62
OVH SAS	16276	CA	2,936	6.64	4,350
CHINANET	23650	CN	1,854	4.20	4,237
CloudFlare, Inc.	13335	US	1,194	2.70	3,412
CNCGROUP	4837	CN	489	1.11	178
Hanaro Telecom Inc.	9318	KR	328	0.742	80
LeaseWeb B.V.	60781	NL	278	0.630	5,771
Akamai International	20940	US	275	0.622	2,137
Aruba S.p.A.	31034	IT	246	0.557	4,229

4.3.3　分析方法

あるマルウェアダウンロードサイトから，異なるマルウェアが異なる日時に配布されることはよく知られる．長期間観測したデータを分析した結果，マルウェアダウンロードサイトは，マルウェアの変化に着目すると，3 つのカテゴリに分けられることがわかった．サイトの特徴に合わせた効果的なブラックリスティングに繋げるため，本研究では，この 3 つのカテゴリに分けた観点での詳細な分析を行う．1 つ目のカテゴリは，*unchanged* で，UNC と呼ぶことにする．このタイプの URL にアクセスした際には，いつも同じハッシュ値のマルウェアがダウンロードされる．2 つ目は，*every-time changed*，で ETC と呼ぶ．このタイプの URL にアクセスした際には，逆に，いつも異なるハッシュ値のマルウェアがダウンロードされる．最後が，*changed occasionally*，COC である．このタイプからは，時々，ハッシュ値の異なるマルウェアがダウンロードされることがある．特に COC について，多くの場合，同じハッシュ値のマルウェアがダウンロードされるが，ある短い短期間だけ，異なるハッシュ値のマルウェアに変化し，次のアクセスでは元のハッシュ値のマルウェアに戻る傾向がある．この短期変化するマルウェアの性質についても分析を行う．

まず，データセットである 43,034 個の URL を，先ほど定義した 3 つのカテゴリに分割した．表 4.6 に分割結果の個数を示す．次に，それぞれのカテゴリの悪性 URL につい

て，その生存期間や，復活の挙動，IP アドレスの変化等の運用リソースを分析するために，特徴量を定義し分析を行った． 4.4 章で，生存分析に関わる特徴量の定義と分析結果を， 4.5 章で，挙動分析に関わる特徴量と分析結果を示す．最後に， 4.6 章でこれらの分析に基づく，3 カテゴリの特徴とその対策について議論する．

表 4.6　3 カテゴリの分類

Category	Count
UNC	11,155
ETC	15,714
COC	16,165
Total	43,034

4.4　URL の生存分析

この章では，悪性 URL の生存期間と，特に正規サイトにはない特徴であるマルウェアダウンロードサイトの生き返りの観点で，3 カテゴリを分析する．これらの知見は，各悪性 URL の適切なブラックリスティング期間の決定に生かせると考える．1 章で被害者の感染の仕組みを説明したが，エクスプロイト・アズ・ア・サービスでは，特にマルウェアダウンロードサイトは，入口サイトや攻撃サイトに比べても攻撃者にとって重要な役割を担う．例えば 1 つのマルウェアダウンロードサイトが，多くの入口サイトや攻撃サイトにリンクされるケースが多くある．それ故，マルウェアダウンロードサイトの役割として，安定して長期間，入口・攻撃サイトにマルウェアを提供する必要がある．本研究の長期的観測から一部のサイトは非常に長期間生存し，また，多くの生き返りが観測された．

4.4.1　特徴量の定義

Lifetime 最初に活動を観測できた日から最後に活動を観測できた期間を，*Lifetime* と定義する．例えば，表 4.2 で `URL_2` と `URL_4` の *Lifetime* は，それぞれ 5 と 8 である．*Lifetime* の定義では，期間の途中のステイタスは考慮しない．

Activedays すべての観測期間において, 活動状態だった日数の合計日数を, *Activedays* と定義する．例えば，表 4.2 で `URL_2` と `URL_4` の *Activedays* は，それぞれ 2 と 7 である．*Lifetime* と比べて，観測期間の途中の状態を考慮した特徴量である．

IntervalMax と IntervalMin まず，インターバルを連続して停止ステイタスが続いた日数と定義する．*IntervalMax* はインターバルの最大値である．表 4.2 で URL_2 と URL_3 の *IntervalMax* は，それぞれ 3 と 2 である．*IntervalMax2* と *IntervalMax3* はそれぞれ 2 番目に大きい値と 3 番目に大きい値を示す．

IntervalAve と IntervalStd *IntervalAve* はインターバルの平均値，インターバルの分散である．

Stop5 と Stop10 これらは悪性 URL 特有である生き返りの活動を分析するために定義した．*Stop5* は，インターバルが 5 以上で 10 未満であった回数と定義する．*Stop10* は，インターバルが 10 以上であった回数と定義する．例えば，表 4.2 ですべての URL で *Stop5* と *Stop10* は 0 である．

4.4.2 分析結果

Lifetime 図 4.3 に，*Lifetime* と *Acrivedays* の CDF(cumulative distribution function) グラフを示す．明るい緑，赤，青が，それぞれ，UNC, ETC, COC の *Lifetiime* を示す．UNC, ETC, COC の平均値はそれぞれ，203.3, 68.3, 195.8 であった．UNC と COC は ETC と比べて値が大きいものが占める割合が大きく，長期間にわたり生存することがわかる．実際に UNC の 10% は，500 日以上の長期間にわたり生存している．先行研究では，攻撃サイトの 10% が 1 週間以上生存，5% が 2 週間以上生存，最も長いもので 2.5 ヶ月生存したと示されているが，本研究では，マルウェアダウンロードサイトは攻撃サイトに比べて長期間に渡り生存することを示した．

Activedays 図 4.3 の暗い緑，赤，青が，それぞれ，UNC, ETC, COC の *Activedays* を示す．UNC, ETC, COC の平均値はそれぞれ，120.9, 26.55, 105.1 であった． UNC と COC は ETC と比べて値が大きいものが占める割合が大きく，長期間にわたり活動することがわかる．実際に UNC の 10% は，290 日以上の長期間にわたり活動している．*Activedays* は *Lifetime* と類似するカーブを示し，*Activedays* の方がグラフの右側（すなわち少ない日数のゾーン）に位置する傾向となっている．

IntervalMax と IntervalMin 表 4.7 に, 3 カテゴリの, *IntervalMax*, *IntervalMax2*, *IntervalMax3*, *IntervalMin* を示す．COC は他カテゴリと比べて，高い *IntervalMax*, *IntervalMax2*, *IntervalMax3* を持つ一方で，*IntervalMin* については低い値となっていることがわかる．これは，COC は基本的に停止期間が長いが（高い *IntervalMax*），一旦活動を開始すると停止状態が少なく継続的に運用される（低い *IntervalMin*）ことを示している．

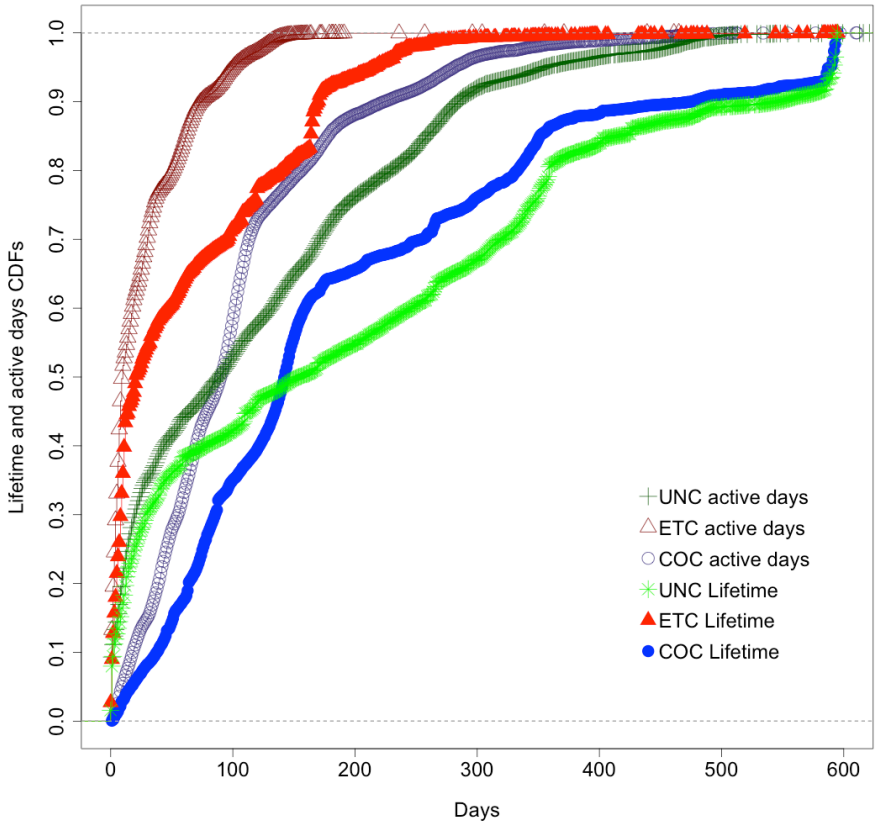

図 4.3　*Lifetime* と *Activedays* の CDF グラフ

表 4.7　*IntervalMax* と *IntervalMin* の平均値

Category	*IntervalMax*	*IntervalMax2*	*IntervalMax3*	*IntervalMin*
UNC	16.42	7.699	5.706	1.549
ETC	15.67	6.303	3.864	2.054
COC	20.25	9.810	7.034	0.9053

IntervalAve と IntervalStd 表 4.8 に，3 カテゴリの *IntervalAve* と *IntervalStd* の平均値を示す．COC の *IntervalAve* は一番小さく ETC の 0.63 倍となった．これは，COC は一時的に集中して活動する一方で，ETC は疎な活動をすることを示している．

Stop5 と Stop10 図 4.4 に，*Stop5* と *Stop10* の CDF グラフを示す．明るい線が *Stop5* の結果である．*Stop5* の UNC, ETC, COC の平均値は，それぞれ，4.68, 1.90, 5.29 であった．概して，UNC と COC は類似するカーブを描き，ETC と比べると大きい回数

表 4.8　*IntervalAve* と *IntervalStd* の平均値

Category	*IntervalAve*	*IntervalStd*
UNC	2.571	2.343
ETC	3.460	2.979
COC	2.163	3.103

が占める割合が大きいことがわかる．実際に，COC の 10% は 15 回以上，活動と停止を繰り返している．特に，COC カテゴリの URL の中で最大値を持つものは，32 回，活動と停止を繰り返していた．暗い線が *Stop10* の結果である．*Stop10* の UNC, ETC, COC の平均値は，それぞれ，1.29, 0.77, 1.70 であった．概して，*Stop10* の方が，高い回数を占める割合は少ないことがわかる．実際に，COC の 5% は，8 回以上，活動と停止を繰り返し，全体の半数は，3 回以上の活動と停止を繰り返していることがわかる．

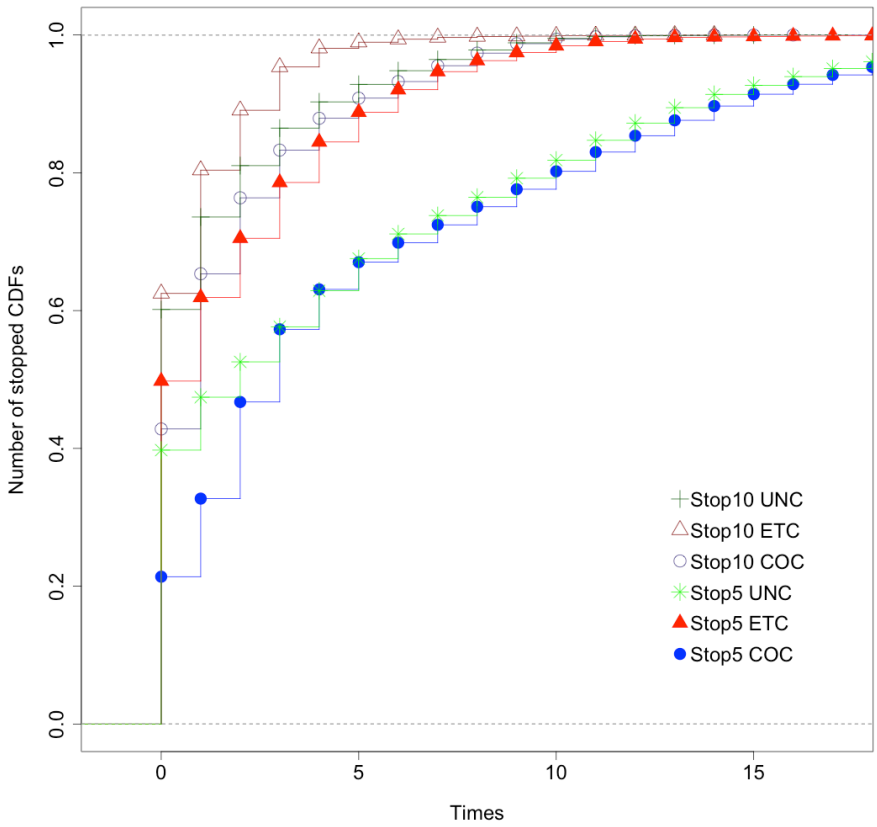

図 4.4　*Stop5* (dark lines) と *Stop10* (light lines) の CDF グラフ

4.5　URL の挙動分析

この章では，悪性 URL の IP アドレスやダウンロードされるマルウェアの変化の挙動を分析する．これらの分析により攻撃者の運用実態を明らかにし，より効果的なブラックリスティングに生かせると考える．例えば，COC カテゴリはブラックリスティングに適している．なぜなら，COC カテゴリは，時系列で見て，多くのタイミングで同一のハッシュ値のマルウェアを配布しているが，ある一時期のみ，異なるハッシュの未知マルウェア配布に切り替え，その後，また元の同一ハッシュ値に戻るという特性があるためである．ブラックリスティングを用いることにより，エンドポイントのアンチウイルスで検出できない未知マルウェアを，ネットワーク側で防御することが期待される．

4.5.1　特徴量の定義

UniqIP 長期間観測すると，IP アドレス変化する場合がある．この IP アドレスのユニークな数を，*UniqIP* と定義する．表 4.4 の例では，`URL_1` と `URL_3` の *UniqIP* の値は，それぞれ，2 と 8 である．

UniqIPMax と **UniqIPMin** それぞれの URL のユニーク IP の数の最大値を，*UniqIPMax* と定義する．表 4.4 の例では，`URL_1` と `URL_3` の *UniqIPMax* の値は，それぞれ，5 と 1 である．

UniqIPAve と **UniqIPStd** それぞれの URL のユニーク IP の数の平均値を，*UniqIPAve*，と定義する．表 4.4 の例では，`URL_1` と `URL_3` の *UniqIPMax* の値は，それぞれ，5 と 1 である．

IPEntropy IP アドレスの変化の度合いを考察するために次の式で算出されるシャノン情報エントロピー *IPEntropy* を定義する．

$$IPEntropy = -\sum_{i \in IPaddress} P(i) log P(i)$$

VTsubmit と **VTscore** VirusTotal から得られるマルウェアに関する情報から 2 つ変数を定義した．VirusTotal は，検査対象ファイルをアップロードすると現時点で 51 種類のアンチウイルス製品の判定結果を得られるサービスであり，過去に検査されたデータが蓄積されていて，その情報の検索結果を返すサービスも提供する．*VTsubmit* は，検査対象マルウェアのハッシュ値をキーとして，過去 VirusTotal にそのハッシュ値のマルウェアが検査されたか否かの指標と定義する．*VTscore* は，同じくハッシュ値をキーと

し，何個のアンチウイルス製品で検出したかを示す指標と定義する．

Netloc と **Path** と **Query** これは各 URL に対する特徴量であり，時系列変化データでは無い．各 URL のホスト部，パス部，クエリー部の長さをそれぞれ，*Netloc*，*Path*，*Query* と定義する．

4.5.2　分析結果

UniqIP *UniqIP* の CDF グラフを図 4.5 に示す．UNC, ETC, COC の平均値はそれぞれ，11.8, 5.10, 12.2 であった．ETC で用いられる IP アドレスのユニーク数は，UNC 及び COC よりも少ないことがわかる．また，実際に，この 1 年半の観測期間で，UNC の 2% が，180 個以上の IP アドレスを用いていることがわかる．

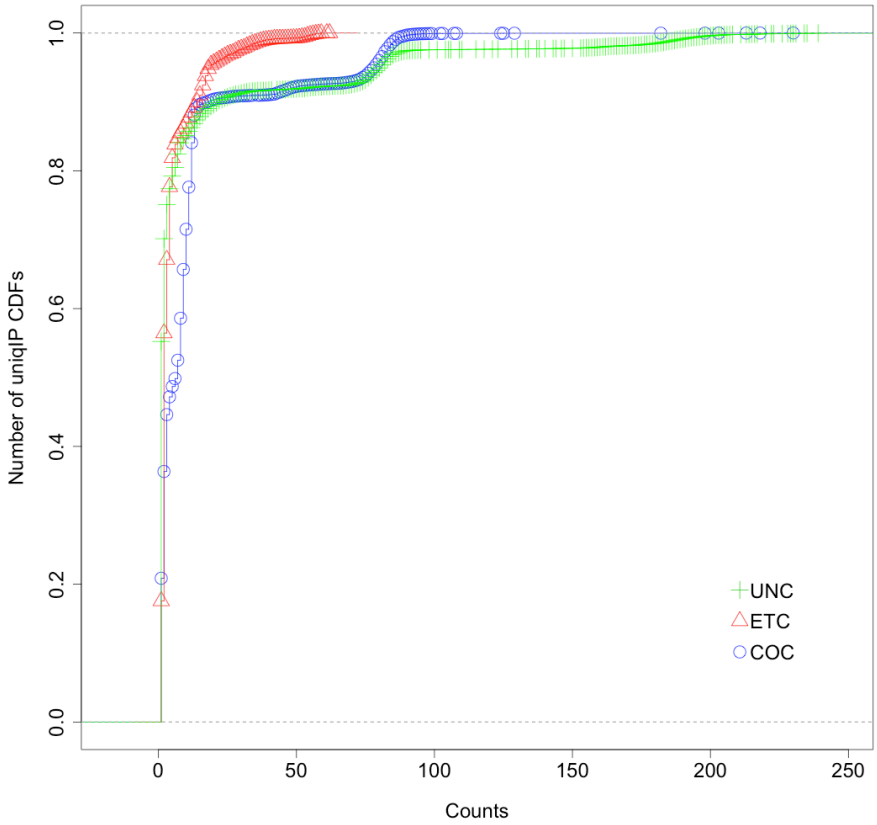

図 4.5　*UniqIP* の CDF グラフ

UniqIPMax と **UniqIPMin** 表 4.9 に *UniqIPMax* と *UniqIPMin* の平均値を示す．

ETC の値は，UNC 及び COC よりも小さく，*UnipIP* で分析したように ETC のユニーク IP アドレスが少ないことに起因すると考えられるが，これ以上の有意な結果は得られなかった．

UniqIPAve と **UniqIPStd** 表 4.9 から COC と比較して UNC の *UniqIPMax*, *UniqIPMin*, *UniqIPAve* は値が大きい．しかしながら，*UniqIPStd* については COC の方が大きく，逆の傾向となっている．これは，COC は持っている IP アドレスの数は UNC に比べて少ないにかかわらず，変化は多分に行われるという COC の特徴と分析できる．

表 4.9　*UniqIPMax* と *UniqIPMin* と *UniqIPAve* の *UniqIPStd* 平均値

Category	*UniqIPMax*	*UniqIPMin*	*UniqIPAve*	*UniqIPStd*
UNC	75.63	61.82	67.56	5.876
ETC	12.03	6.527	8.801	2.264
COC	52.88	29.99	37.94	8.655

IPEntropy 図 4.6 に，*IPEntropy* に関する CDF グラフを示す．UNC, ETC, COC の平均値は，それぞれ，0.456, 1.02, 1.50 であった．グラフからも COC は，他のカテゴリと比べて，*IPEntropy* が高いことが読み取れる．これは，COC の高い *UniqIPStd* と同様に，IP アドレスの変化が頻度が大きいことがわかる．

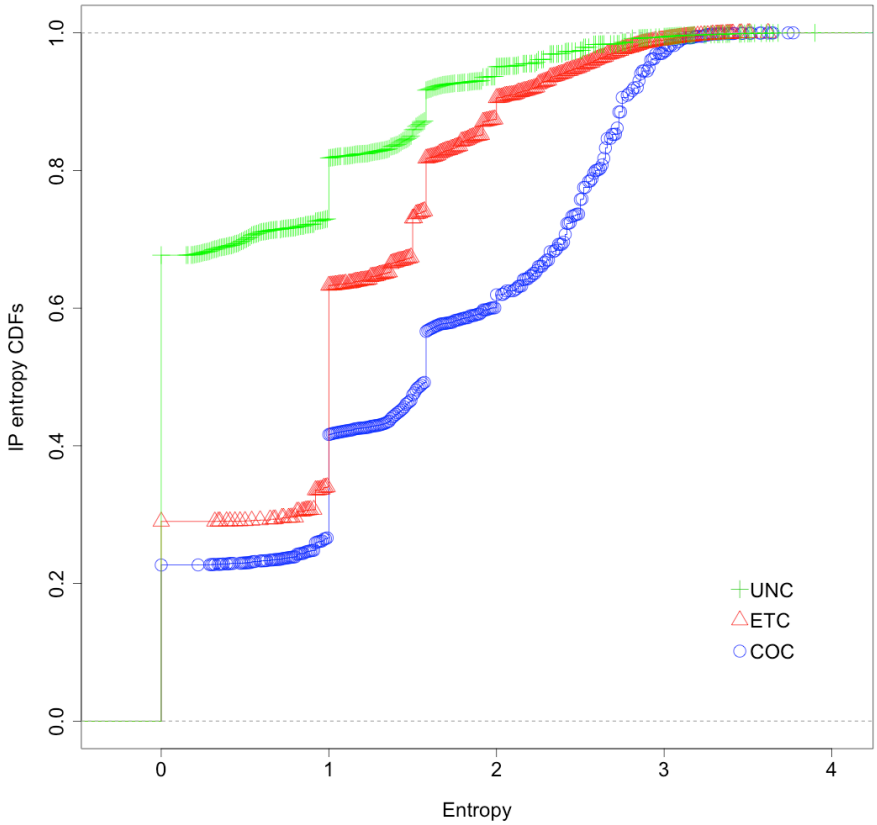

図 4.6　*IPEntropy* の CDF グラフ

VTsubmit と **VTscore** VirusTotal には検索制限があったため，各カテゴリからランダムに 50 個の URL を選び，対応するマルウェアハッシュ値をキーに検索を行い情報を得て，全体の平均をとることで値を算出した．ここで，UNC, ETC, COC のユニークなマルウェアハッシュの数は，50 個，755 個，1253 個である．表 4.10 に各カテゴリの *VTsubmit* と *VTscore* の平均値を示す．UNC の *VTsubmit* は非常に大きく，UNC から配布される 96% のマルウェアが VirusTotal に存在し，つまり既知マルウェアの割合が大きいことがわかる．また，ETC からはハッシュ値が毎回異なるマルウェアが配布されることから，ETC の *VTsubmit* は低くなると考えられる．一方で，ETC の *VTScore* は高いことから，ETC 配布マルウェアは，アンチウイルスソフトでは検出されやすいことを示している．

次に COC カテゴリから得られるマルウェアについて分析を行った．表 4.11 に COC カテゴリの長期観測において，各マルウェアが観測された回数と，VirusTotal へ投稿

表 4.10　*VTsubmit* と *VTscore* の平均値

Category	*VTsubmit*	*VTscore*
UNC	0.960	30.5
ETC	0.00265	33.0
COC	0.142	27.3

されていないマルウェアの割合 (*Unsubmitted rate*) を示す．観測される回数に応じて，*Unsubmitted rate* が異なることがわかった．図 4.7 に観測回数と *VTscore* の関係を示す．観測回数が増えるにつれて，図の赤線のように，*VTscore* は増加する，つまり多くのアンチウイルスソフトで検出可能になること，がわかる．これらの結果から，COC 配布マルウェアで観測回数が極端に少ないもの，つまり COC の特徴である稀に変わり元に戻るマルウェアは，未知マルウェアの傾向があると考えらえる．また，全体の 80% が 10 日未満の観測日数であり，特にこれらの 30% については *VTscore* が 20 未満と低く，注意が必要なマルウェアであると考えられる．

さらに COC カテゴリから得られるマルウェアについて分析を行った．図 4.8 に VirusTotal から得られた全てのアンチウイルス製品の判定結果の累計値を示す．ここで Positive は該当のアンチウイルス製品が悪性ファイルと判定したことを，Negative は良性と判定したことを示す．図では Positive の多い順に並べている．表 4.1 に示した，観測環境で選定したアンチウイルスソフトは，図 4.8 によると上位ランク (No.1 及び No.4) に，また中位のランク (No.26 及び No.28) に位置付けられ，検出精度と過検知のバランスで，本研究における 4 製品の選定とその閾値について不適切な要因は見つからなかった．さらに主に図 4.8 の下の方に目立つが，Positive より Negative の数が極端に多い製品群があることがわかる．実際には，Positive に関して TruePositive（確かに悪性）であるか FalsePositve（実際は良性）であるのかは不明であり，同様に，Negative に関して TrueNegative（確かに良性）であるか FalseNegative（実際は悪性）であるかは不明である．しかしながら図 4.8 の結果は，アンチウイルス製品には 2 つの製品グループ，極端に高い Positive を示すグループと極端に高い Negative を示すグループがあることを示唆している．本研究での，*VTscore* の定義では，Positive の値のみを用いてるが，将来的に Negative も何らかの指標として活用できる可能性があると考えられる．最後に，図 4.8 で Positive も Negative も極端に少ないグループが確認できるが，アンチウイルス製品としての働きが適切でないと考えられる．

表 4.11　観測回数と *Unsubmitted rate* の関係

観測回数	*Unsubmitted rate*
1	94.0%
2-10	29.3%
over 10	2.43%

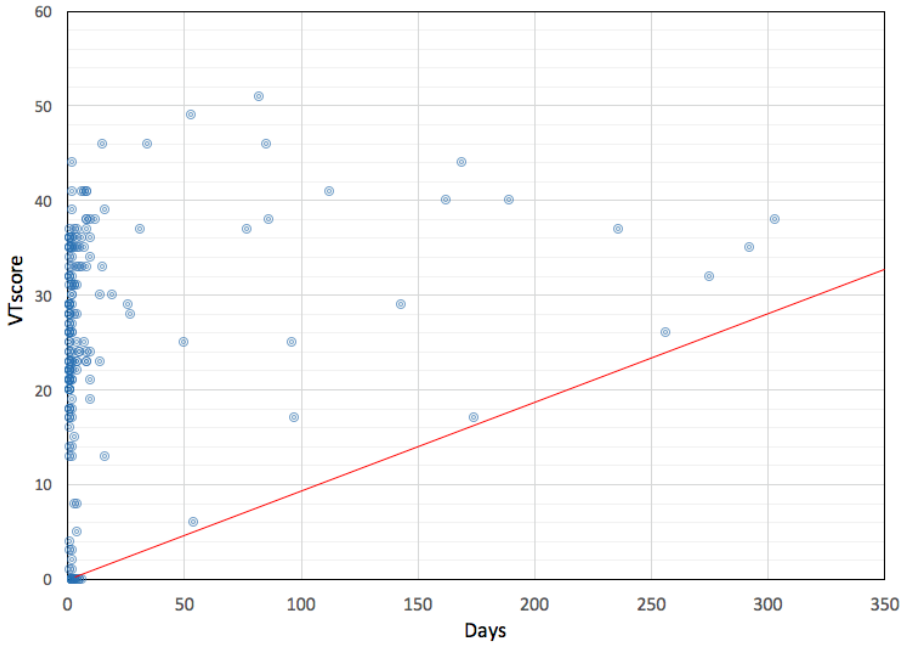

図 4.7　観測回数と *VTScore*

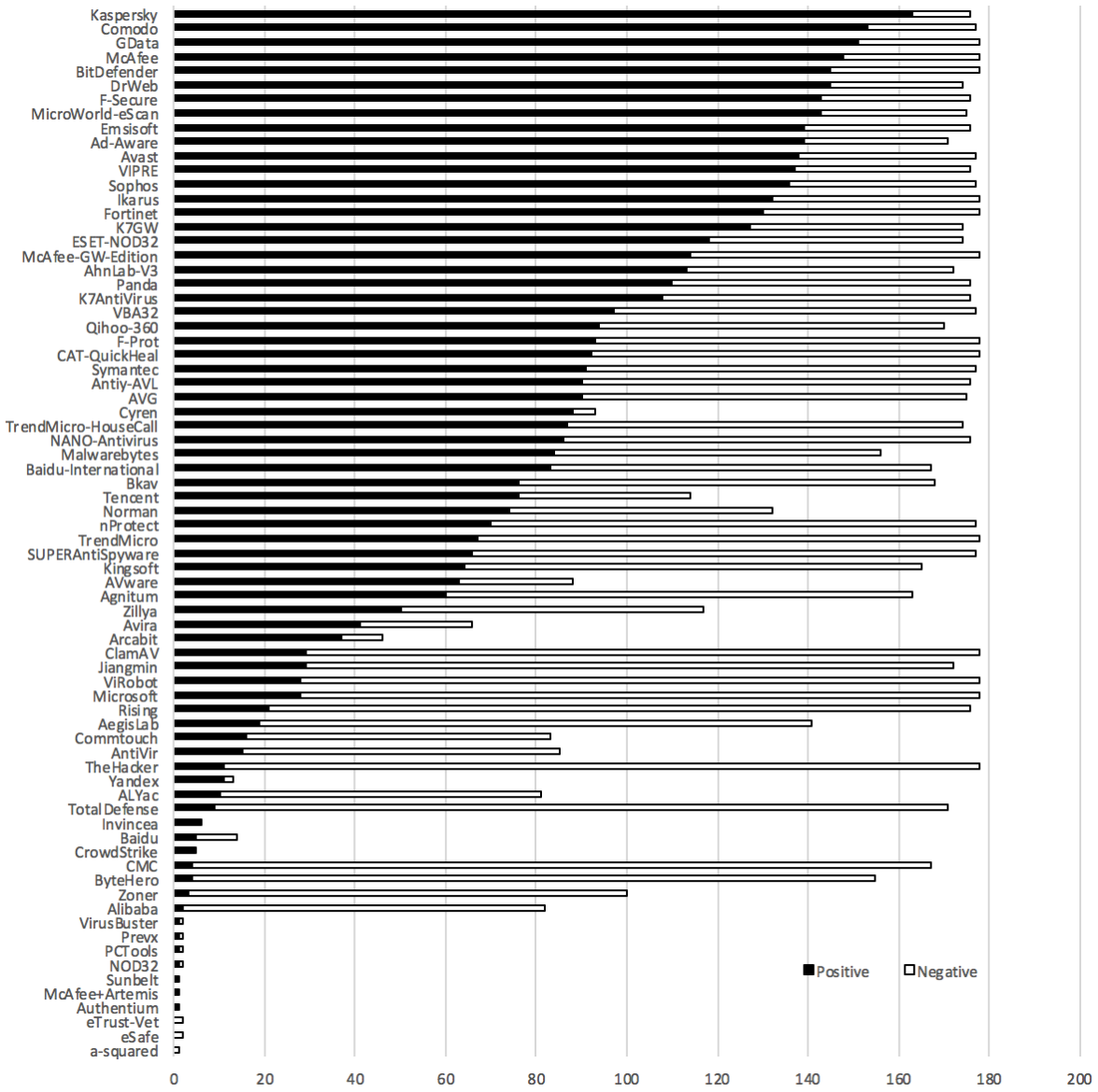

図 4.8　アンチウイルス製品の判定結果

Netloc と **Path** と **Query** 表 4.12 に各値の平均値を示す．他カテゴリに比べて，UNC は *Path* が大きい，また ETC は *Netloc* が大きい．COC はいずれも中間の値であった．特筆すべきは，ETC の *Query* で他カテゴリと比較して約 10 倍程度の値となった．

表 4.12　*Netloc* と *Path* と *Query* の平均値

Category	*Netloc*	*Path*	*Query*
UNC	19.77	38.05	13.97
ETC	25.64	30.99	247.2
COC	20.36	27.71	31.98

4.6　各カテゴリの特性考察

4.4 章の URL の生存分析では，10 個の特徴量を定義し，生存期間や生き返りの特異な活動を明らかにした．また 4.5 章の URL の挙動分析では，11 個の特徴量を定義し，IP アドレスや配布されるマルウェアの変化の特徴を明らかにした．本章では，これらの分析結果から，各カテゴリの特性を考察する．

4.6.1　UNC カテゴリの特性

長期間にわたり存在． 4.4.2 章の分析結果より, UNC は他のカテゴリより高い *Lifetime* と *Activedays* を持つ．実際，UNC の 10% が 500 日以上生存した．

大量の IP リソースを持つが変化は少ない． 4.4.2 章の分析結果より，UNC は他のカテゴリより高い *UniqIP* を持つ．実際，UNC の 2% が 180 個以上の IP アドレスを，1.5 年の観測期間の間に変化させている．しかしながら，この IP アドレスの変化の頻度は高くはない．これは 4.4.2 章の分析結果より，*IPEntropy* が他のカテゴリよりも低いためである．

4.6.2　ETC カテゴリの特性

短命． 4.4.2 章の分析結果より，ETC は他カテゴリより低い *Lifetime* と *Activedays* を持つ．

短期間でのステイタス変化有り． 4.4.2 章の分析結果より，ETC は他カテゴリより *Stop5* 及び *Stop10* は低いが，*IntervalAve* と *IntervalMin* は逆に高い．これは 5 日未満の短い期間で，活動と停止が行われやすいこと示す．

IP リソースは少ないが変化は多い． 4.4.2 章の分析結果より，ETC は他カテゴリより *UniqIP* は低いが，*IPEntropy* は逆に高い．これは少ない IP リソースで，可能な限り

変化させていることがわかる．

既知検体の配布． 4.5.2 章の分析結果より，ETC は他カテゴリより *VTscore* が高い．これは多くのアンチウイルスソフトが検出可能であることを示す．

長いクエリー部分． 4.5.2 章の分析結果より，ETC は他カテゴリより *Query* が 10 倍程度と極端に大きい．これは，常にマルウェアを変化させる攻撃者が共通して用いる ExploitKit に類似したシステムの存在がうかがえる．

4.6.3　COC カテゴリの特性

長期間にわたり存在． 4.4.2 章の分析結果より，COC は UNC 同様，高い *Lifetime* と *Activedays* を持つ．実際，UNC の 10% が 460 日以上生存した．また，130 日未満では，UNC よりも COC が生存期間が長い割合が高く，それ以上では逆の傾向がある．

短期間でのステイタス変化無し． 4.4.2 章の分析結果より，COC の *IntervalAve* は他カテゴリより低く，5 日未満の短い期間では，活動と停止は行われにくいことを示す．これは ETC とは逆の性質を持つことがわかる．

多数の生き返り． 4.4.2 章の分析結果より，COC は他のカテゴリと比べて生き返りが多い傾向にある．実際に，1.5 年の観測期間で，10% が 15 回以上の生き返りが行われた．観測データで生き返り回数の最大値は，32 回に及んだ．

大量の IP リソースを持ち変化も多い． 4.5.2 章の分析結果より，COC は他のカテゴリより高い *UniqIP* と *IPEntropy* を持つ．これは UNC や ETC とは異なる特徴である．

未知検体の配布． 4.5.2 章の分析結果より，COC は他カテゴリより *VTscore* が低く，一部のアンチウイルスソフトのみしか検出できないことを示す．また UNC と比べて *VTsubmit* が大幅に低く，VirusTotal 等のセキュリティインテリジェンスでの共有がなされていない可能性が高いことを示す．

4.7　議論

4.7.1　対策の考察

UNC について，ユーザのブログ等一般的な Web サイトが脆弱性を突かれたりしてマルウェアを配布するように改竄されているケースが多いと考えられる． 4.6 章で分析した，「大量の IP リソースを持つが変化は少ない」特徴は，母体とするホスティングサイトの特徴を引き継いだものである可能性が高いと思われる．サイトの管理者は悪用されているのに気づいていない可能性がある．またダウンロードされるマルウェアが全く変更されな

いのは，攻撃者も該当サイトでの運用に関心を失い，放棄してしまったのかもしれない．長期観測において，ある URL は AS 名が頻繁に変わるものもあった．これはホスティングサイトの商業上の都合等で移転が発生したケースと考えられ，この移転により，攻撃者の意図とは無関係に，IP アドレスレンジが変化していた．UNC に対する対策は，本質的には，例えばブログであればこれらのサイトをユーザ同意に基づき適切に削除していくことである．しかしながら，このような悪性サイトを発見して ISP 等に通知してもレスポンスがないケースも多く運用として回りづらい．特に [58] らの調査では 6 割が適切に扱われなかったとの報告がある．そこで，「長期間存在する」という特徴を考慮して，ネットワーク側で一元的な URL ブラックリスティングが効果的・効率的な対策であると考える．URL ブラックリスティングであれば，先ほどのような AS の変更や IP の変更の影響を受けない，また 4.7.2 章で議論するオーバーブロッキングも発生しないので適切な方法であると考えられる．

ETC について， 4.5.2 章で，アクセス毎に配布マルウェアを変更する攻撃者が共通して用いる配布アプリケーションの存在を推測したが，少なくとも 2 つの点で，通常の ExploitKit とは別の存在である可能性があることを言及する．まず，一般的に，ExploitKit の目的は，脆弱性を利用し攻撃コードを実行させ，最終的にマルウェアをダウンロードさせることである [69, 70, 71]．次に，本実験環境では，ハイインタラクション Web クローラを用いているため，攻撃コードの実行を検出可能であり，攻撃コードが実行されたサイトは攻撃サイトと定義して，本研究対象としないため，データセットから除外してる．特徴として見られた「短期間でのステイタス変化有り」，「IP リソースは少ないが変化は多い」，「長いクエリー部分」は，この配布アプリケーションを用いた配布基盤の特徴とも考えられる．またブラックリスティングを回避するために「短命」である可能性がある．ETC に対する対策は，現在までの検討では提言は難しい．より踏み込んだ観測・分析が必要である．配布アプリケーションは ExploitKit の亜種であるか，それとも別な物であるか等の検討も必要である．

COC について， 4.4.2 章と 4.5.2 章の分析結果より，「長期間にわたり存在」，「大量の IP リソースを持つ」は UNC と共通であることから，配布基盤については UNC と類似すると考えられる．しかし UNC と COC ではその運用の観点で異なる．特に，マルウェアが時折変更され，「多数の生き返り」，「未知検体の配布」は，他のカテゴリに無い注意が必要な特徴である．COC に対する対策は，URL ブラックリスティングが効果的であると考える．「長期間にわたり存在」して，「多数の生き返り」や「未知検体の配布」が発生しても対策とできるためである．

4.7.2 ブラックリスティングとその課題及び展望

脅威情報に基づいたブラックリスティングは，サイバー攻撃の高度化に伴い，多層防御の観点からも，広く用いられる技術である．Microsoft は，フィッシングサイトやマルウェアダウンロードサイトと報告されるサイトからユーザを守るために，SmartScreen URL Filter [72] を提供している．McAfee は，ブラウザの検索結果に危険性を示すアイコンを付与する，SiteAdvisor を提供している [73]．このようなアラートはユーザが潜在的に危険性のあるサイトにアクセスするのを防ぎ，NortonSafeWeb [74] も類似する目的のサービスである．Yandex は Safe Browsing API SDK [75] を提供し，ユーザの環境に組み込んで用いることが可能である．Google Safe Browsing [76] は，Firefox, Safari, Chrome 等の著名なブラウザに標準で組み込まれており，同様のブラックリスティング技術が用いられている．

ブラックリスティングの課題は大きく 2 つあり，1 つ目はカバレッジの拡大である．Metcalf らは，86 個のインターネットブラックリストを比較した結果，ドメイン名ベースで，96.16% から 97.37% が一致せず，各リストでユニークであり，また，IP ベースでは，82.46% から 95.24% が同様にユニークであることを示した [77]．この結果は，各ブラックリストがほとんどオーバーラップしないことを示している．Kuhrer らは 15 個のパブリックなブラックリストと 4 個の商用アンチウイルスベンダのリストを調査し，各リストを分類しどのマルウェア関連するドメインをカバーできるかを示し，特に Domain Generation Algoriths(DGA) [78] を用いるマルウェアから守るのは困難であることを示した [79]．2 つ目の課題は，オーバーブロッキングと運用コストである．Web サービスに対する，ブラックリストのカバー率は，URL, FQDN, IP アドレスの順に大きくなる．カバー率が大きくなれば効果も高くなるが，同時にオーバーブロッキングの危険が高まる．オーバーブロッキングとは正規な Web サイトもブロックしてしまうことを意味する．例えば，FQDN ブロックを行い，その FQDN 配下に正規サイトの URL が含まれていた場合がある．オーバーブックに対しては，URL ブロックを行うことで回避ができるが，運用コストとのトレードオフとなる．特に FQDN ブロックは DNS 等で軽量に運用できる一方，URL は Paloalto [80] 等の L7Firewall で運用する必要があり一般に運用コストは高くなる．また，常に良性なものは排除し，悪性なもののみをリストとして導入して運用する必要がある．

ブラックリスティングの今後の展望として，効率的に適切なブラックリスト掲載期間を定める必要があると考えられる．インターネット業界最大手である Google も，本研究

同様に，悪性候補 URL を実際にクローリングすることで悪性か否かをチェックしている [81]．一方，Google Safe Browsing のアプローチとは異なり，本研究の目的は，ブラックリストのエントリのステイタスの確認の効率を改善することである．悪性候補 URL はインターネット上に大量にあり，かつ，本分析で見たようにステイタスが頻繁に変化するものも多いことから，効率的にクローリングすることは重要な課題である．本研究の分析結果は，特に，ブラックリスティングを扱うセキュリティ研究者やエンジニアへ貢献できると信じている．

4.7.3　観測環境と倫理の考察

観測環境を，攻撃者側が欺く手法としてクライアント IP のブラックリスティングや，ブラウザや OS のエミュレータの検知やフィンガープリンティング技術がある．本研究の観測環境では，IP ブラックリスティングを回避するために定期的に IP アドレスを変更している．さらに，エミュレータではなく，実際の OS とブラウザを用いているため，攻撃者には一般のユーザが長期的にアクセスしてる環境に見え，観測環境と気づかれるリスクを最小限としている．

また倫理面での考察になるが，長期間悪性サイトを観察するためには，実際に各サイトにアクセスする以外の代替え手段は無いと考えこの手法を選択した．このアクティブな観測において不要なトラフィック量をできる限り削減するために，短い期間での継続的なスキャン行為のようなことは一切行っておらず，各 URL について 1 日に 1 回のみのアクセスとしている．また，このアクセスは，HTTP の正規のリクエストを送っており，ペネトレーションテストや脆弱性スキャンのような細工をしたパケットを送ることは無いことを付け加える．

4.8　本章のまとめ

4 章では，約 43,000 個のマルウェアダウンロード URL を約 1.5 年の長期間にわたり観測した結果を分析した．一部のマルウェアダウンロードサイトは長期間にわたり生存を続け，また消失と復活を繰り返す特徴があることを明らかにした．特に，マルウェアの変化に着目し，3 つのカテゴリを定義し解析を行った．*unchanged* カテゴリの 10% は 500 日以上生存を続けること，*changed occasionally* カテゴリの 10% は 15 回以上消失と復活を繰り返すことを明らかにした．さらに IP アドレスの変化等の他の解析も行った．3 つのカテゴリごとに攻撃者のリソースやオペレーションが異なることを明らかにした．解析に

基づき，各カテゴリの特徴と効果的なブラックリスティング対策について検討を行った．本章は，1.2 章で述べた，マルウェアに関連する通信を特定する研究分野の中で，マルウェア感染前の対策として利用される入口系に位置付けられ，間接的なアプローチである．

第 5 章

結論

本研究では，未知のマルウェア対策技術を検討した．要約と貢献は次の通りである．2 章「統計的手法を用いたマルウェア検出」では，ファイルの静的な特徴に対して統計的手法を用いてマルウェア判定を行った．ファイルの多数のヘッダ情報から，判定に有効な情報に絞りモデルを構築し，そのモデルを利用することで軽量にマルウェア判定する手法を提案し未知マルウェアに対しても評価を行った．本手法は特に，インターネット上のマルウェアダウンロードサイトから配布されるような一般に広く流通するマルウェアに特に有効であることを示した．3 章「攻撃コード構成の特徴を用いた ROP コード検出」では，ROP(Return-Oriented Programming) と呼ばれる攻撃コードが，シェルコードの複合ルーチンの外側に配置されると言う攻撃コードを構成する上での特徴を利用して検出を行った．本手法は，2 章での検出対象である一般に広く用いられるマルウェアでなく，標的型攻撃や未知であるゼロデイ脆弱性とともにも用いられるマルウェアである悪性文書ファイルを対象とした．ROP コードを静的に検出することで，悪性文書ファイル判定を行う方式を提案し，実際の検体にて評価を行い有効性を確認した．4 章の「効果的なブラックリスティングの検討」では，2 章や 3 章で示したマルウェア自体の特徴を捉えて検出するのではなく，マルウェアのダウンロードサイトを，効果的にブラックリスティングし未知マルウェア対策につなげた．これにより，マルウェア自体のハッシュ値等の特徴が変化して，未知マルウェアがダウンロードされた場合でも，検出可能である．長期観測を行い，マルウェアの変化に応じて，マルウェアダウンロードサイトの 3 つのカテゴリに分けられることに着目・解析し，それぞれのサイトの特徴に応じた効果的なブラックリスティングを検討した．2,3,4 章で示した 3 つのアプローチは，独立した方式であり，多層防御の観点で併用することで，未知マルウェアに対する感染を未然に防ぐ効果を発揮する．

本研究では，1 章で述べた，直接的，準直接的，間接的の 3 つの異なるアプローチを用

いたが，現状までの動向と今後について考察を以下に行う．まず，直接的な方式については，過去から多くの研究がなされており，特に静的解析分野はアンチウイルス製品やIDS/IPS へ，動的解析分野はサンドボックス製品へと，商用化にも大きく貢献している．本研究は，ヘッダ情報に着目し特徴量を減らす方向で機械学習を用い判定精度向上と計算量削減の両輪を目指した．しかし，将来的に，計算資源が潤沢になり，さらに新しいアルゴリズムが登場するであろうことを考えると，特徴量を増やし高度なアルゴリズムの適用により，さらに優れた検出が可能になると考えられる．次に，準直接的な方式については，攻撃者は常に進化するため，SOC 等のセキュリティ分析官による手動解析と研究者による特性の発見が大切になると考えられる．本研究では，ROP コードに着目したが，このような特定のコードの特性の発見は，現段階では，前述した専門知識を持つ人間による発見が必要と考え，これらの蓄積によって，将来的には，機械学習等による自動化につながると考える．特に ROP というテクニック自体は，本研究で対象にしたアーキテクチャやレイヤ以外でも適用可能であり，特に様々なシステムに共通するような，かなり低いレイヤの脆弱性と組み合わされた場合，非常に脅威となりうるので今後も注意が必要である．最後に，間接的な方式についてだが，マルウェア配布や感染を未然に防ぐことができるため，今後最も技術の発展が期待される．過去からの研究が少なく，特にブラックリスティングはカバレッジやオーバーブロッキングの課題の解決が期待される．

最後に将来課題について述べる．機械学習をはじめとする AI は，今後ますます，セキュリティ分野への応用が発展すると考えられる．本研究でも用いたが，一部の機械学習アルゴリズムは，有効な特徴量を選択してくれるものもある．しかしながら，攻撃者は常に先手を行き，裏をかくことを考えるため，特徴量に関する人による分析や調整も大切になると考える．特に必要な技術力を持つセキュリティ分析官や研究者の育成が急務である．また，さらに高度な機械学習アルゴリズムの中には，識別性能が高いが，有効であった特徴量が得られないものもあり，この場合，判定の理由や根本原因の特定につなげにくいものもあり，原因特定が必要となるセキュリティ分野への適用には課題がある．また優れた研究をするためには，実データが必要であるが，セキュリティ分野では，公開される規模の大きいデータセットは少なく，自前での収集が一般的である．特に，組織，企業の持つログは重要な研究データとなる一方，機密性が高い情報も含まれるため扱いが困難な側面もあり，この課題解決が望まれる．最後に，本研究では対象としなかったが，セキュリティインテリジェンスを用い攻撃者をプロファイリングする取り組みは，攻撃自体を未然に抑止することができるため，今後の発展に期待したい．

参考文献

[1] マカフィー：McAfee 脅威レポート 2017 年 9 月, マカフィー. https://www.mcafee.com/jp/resources/reports/rp-quarterly-threats-sept-2017.pdf.

[2] Guardian News：Antivirus software is dead, says security expert at Symantec. http://www.the-guardian.com/technology/2014/may/06/antivirus-software-fails-catch-attacks-security-expert-symantec.

[3] FireEye AX. https://www.fireeye.jp/products/malware-analysis.html.

[4] Microsoft : Enhanced Mitigation Experience Toolkit. https://technet.microsoft.com/ja-jp/security/jj653751.aspx.

[5] 仲小路 博史, 重本 倫宏, 鬼頭 哲郎, 林 直樹, 寺田 真敏, 菊池 浩明. 多種環境マルウェア動的解析システムの提案および評価. 情報処理学会論文誌, Vol.56(9), pp.1730-1744, 2015.

[6] Chubachi, Y. and Aiko, K.：SLIME: Automated Anti-sandboxing disarmament system. http://www.ffri.jp/assets/files/research/research_papers/bhasia15_chubachi_aiko_v4.pdf.

[7] Kolter, J. Z. and Maloof, M. A. Learning to detect and classify malicious executables in the wild. In *Journal of Machine Learning Research 7, pp.27212744*, 2006.

[8] Ye, Y., Chen, L., Wang, D., Li, T., Jiang, Q. and Zhao, M. SBMDS: An interpretable string based malware detection system. In *Journal in Computer Virology, Vol.5, No.4, pp.283293*, 2009.

[9] 戸部和洋, 森達哉, 千葉大紀, 下田晃弘, 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. マルウェア対策研究人材育成ワークショップ (MWS), 2010.

[10] Schultz, M.G., Eskin, E., Zadok, E. and Stolfo, S.J. Data mining methods for detection of new malicious executables. In *IEEE Symposium on Security and Privacy, pp. 38-49*, 2001.

[11] Karim, E. M., Walenstein, A., Lakhotia, A. and Parida, L. Malware phylogeny generation using permutations of code. In *Journal in Computer Virology, Vol. 1, Issue 1-2, pp. 1323*, 2005.

[12] 岩村誠，伊藤光恭，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， vol. 51, pp. 16221632, 2010.

[13] Lyda, R. and Hamrock, J. Using Entropy Analysis to Find Encrypted and Packed Malware. In *IEEE Security and Privacy*, 2007.

[14] Shafiq, M., Tabish, S., Mirza, F., and M. Farooq. PE-Miner: Mining structural information to detect malicious executables in realtime. In *Recent Advances in Intrusion Detection, vol. 5758 of Lecture Notes in Computer Science, pp. 121141, Springer Berlin Heidelberg*, 2009.

[15] Raman, K.:Towards Classification of Polymorphic Malware. `https://www.blackhat.com/docs/webcast/TowardsClassificationofPolymorphicMalware-Final.pdf`.

[16] SANS Institute：Attributes of Malicious Files, SANS Institute reading room. `http://www.sans.org/reading-room/whitepapers/malicious/attributes-malicious-files-33979`.

[17] Ligh, M., Adair, S., Harstein, B. and Richard, M.: Malware Analyst's Cookbook and DVD:Tools and Techniques for Fighting Malicious Code.

[18] 八木　毅，針生　剛男，大崎　博之，村田　正幸．Web サイトへのマルウェア感染攻撃に用いられるボットネットの分析．マルウェア対策研究人材育成ワークショップ (MWS), 2013.

[19] PEiD v0.95 Final. `http://woodmann.com/BobSoft/Pages/Programs/PEiD`.

[20] Microsoft：Microsoft PE and COFF Specification. `https://msdn.microsoft.com/en-us/windows/hardware/gg463119.aspx`.

[21] pefile is a Python module to read and work with PE (Portable Executable) files. `https://github.com/erocarrera/pefile`.

[22] R-Projcet：Package bootStepAIC. `https://cran.r-project.org/web/packages/bootStepAIC/bootStepAIC.pdf`.

[23] R. Tibshirani. Regression shrinkage and selection via the lasso. In *Journal of the*

Royal Statistical Society, Series B., 1996.

[24] R-Projcet：Package glmnet. https://cran.r-project.org/web/packages/glmnet/glmnet.pdf.

[25] 田中恭之，有川隼，畑田充弘. 統計的手法を用いたマルウェア判定の実験結果. マルウェア対策研究人材育成ワークショップ (MWS), 2014.

[26] Microsoft：Developer Network Resource Type. https://msdn.microsoft.com/en-us/library/ms648009%28v=vs.85%29.aspx.

[27] Malware Classifier. https://sourceforge.net/projects/malclassifier.adobe/.

[28] R-Projcet：Package e1071. https://cran.r-project.org/web/packages/e1071/e1071.pdf.

[29] シマンテック：2014 年 インターネットセキュリティ脅威レポート 第 19 号, シマンテック. http://www.symantec.com/ja/jp/security_response/publications/.

[30] 情報処理推進機構：2013 年版 10 大脅威 身近に忍び寄る脅威. http://www.ipa.go.jp/security/vuln/10threats2013.html.

[31] Hovav Shacham. The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86). In *Proceedings of the 14th ACM conference on Computer and Communications Security (CCS)*, 2007.

[32] 日本のセキュリティチーム：攻撃コードのトレンド（Exploitation Trend）～セキュリティ インテリジェンスレポート第 16 版から～, Microsoft. http://blogs.technet.com/b/jpsecurity/archive/2014/06/11/exploitation-trend-16.aspx.

[33] 日経コンピュータ：添付ファイルに工夫の数々. http://itpro.nikkeibp.co.jp/article/COLUMN/20140317/544186/.

[34] Tyler Bletsch, Xuxian Jiang, Vince W. Freeh, and Zhenkai Liang. Jump-oriented programming: A new class of code-reuse attack. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, 2011.

[35] Mathias Payer and Thomas R. Gross. String oriented programming: when ASLR is not enough. In *Proceeding PPREW '13 Proceedings of the 2nd ACM SIGPLAN Program Protection and Reverse Engineering Workshop.*, 2013.

[36] FireEye Threat Research Blogs: New Zero-Day Exploit targeting Internet Explorer Versions 9 through 11 Identified in Targeted Attacks. https://www.fireeye.com/blog/threat-research/2014/04/

new-zero-day-exploit-targeting-internet-explorer-versions-9-through-11-identif
html.

[37] 三村守, 田中英彦. Handy Scissors:悪性文書ファイルに埋め込まれた実行ファイルの自動抽出ツール. 情報処理学会論文誌，Vol.54，No.3，pp.1211-1219, 2013.

[38] 大坪雄平, 三村守, 田中英彦. ファイル構造検査による悪性 MS 文書ファイルの検知. 情報処理学会研究報告, Vol.2013-IOT-22, No.16, 2013.

[39] Analyzing MSOffice malware with Office-MalScanner. http://www.reconstructer.org/papers/Analyzing%20MSOffice%20malware%20with%20OfficeMalScanner.zip.

[40] Lucas Davi, Ahmad-Reza Sadeghi, and Marcel Winandy. ROPdefender: A practical protection tool to protect against return-oriented programming. In *Proceedings of the 6th Sym- posium on Information, Computer and Communications Security (ASIACCS)*, 2011.

[41] Vasilis Pappas, Michalis Polychronakis, and Angelos D. Keromytis. Transparent ROP Exploit Mitigation Using Indirect Branch Tracing. In *Proceedings of 22nd USENIX Security Conference on Security*, 2013.

[42] Cheng, Y., Zhou Z., Yu, M., Ding, X. and Deng, H.R. ROPecker: A generic and practical approach for defending against ROP attacks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2014.

[43] Carlini, N., Wagner, D. ROP is Still Dangerous: Breaking Modern Defenses. In *Proceedings of 23nd USENIX Security Conference on Security*, 2014.

[44] Goktas, E., Athanasopoulos, E., Polychronakis, M., Bos, H. and Portokalidis, G. Size Does Matter: Why Using Gadget-Chain Length to Prevent Code-Reuse Attack is Hard. In *Proceedings of 23nd USENIX Security Conference on Security*, 2014.

[45] RAPID7 : metasploit. The world ' s most used penetration testing framework. https://www.metasploit.com/.

[46] Tanaka, Y., Goto, A. n-ROPdetector: Proposal of a method to detect the ROP attack code on the network. In *Proceedings of 7th Workshop on Cyber Security Analytics, Intelligence and Automation (Safeconfig)*, 2014.

[47] 田中恭之, 後藤厚宏. 攻撃コードの特徴から見た対策の検討. 電子情報通信学会, 暗号と情報セキュリティ シンポジウム (SCIS),4C2-4, 2014.

[48] Exploit Database : Offensive Security ' s Exploit Database Archive. https:

//www.exploit-db.com/.

[49] Ramirez-Silva, E., Dacier, M. Empirical Study of the Impact of Metasploit-Related Attacks in 4 Years of Attack Traces. In *Proceedings of 12th Asian Computing Science Conference on Advances in Computer Science: Computer and Network Security (ASIAN)*, 2007.

[50] Microsoft：Compound File Binary File Format, Microsoft. http://download.microsoft.com/download/9/5/E/95EF66AF-9026-4BB0-A41D-A4F81802D92C/%5BMS-CFB%5D.pdf.

[51] contagio. malware dump. http://contagiodump.blogspot.jp/.

[52] malwr. https://malwr.com/.

[53] HP Security Research Blog: Technical Analysis of CVE-2014-1761 RTF Vulnerability, HP. http://h30499.www3.hp.com/t5/HP-Security-Research-Blog/Technical-Analysis-of-CVE-2014-1761-RTF-Vulnerability/ba-p/6440048.

[54] ファイル構造検査による悪性 MS 文書ファイル検知手法の評価（発表スライド）. http://www.iwsec.org/mws/2013/presentation/3B1-3-slide.pdf.

[55] Chris Grier, Lucas Ballard, Juan Caballero, Neha Chachra, Christian J. Dietrich, Kirill Levchenko, Panayiotis Mavrommatis, Damon McCoy, Antonio Nappa, Andreas Pitsillidis, Niels Provos, M. Zubair Rafique, Moheeb Abu Rajab, Christian Rossow, Kurt Thomas, Vern Paxson, Stefan Savage, and Geoffrey M. Voelker. Manufacturing Compromise: The Emergence of Exploit-as-a-Service. In *Proceedings of the ACM Conference on Computer and Communications Security(CCS)*, 2012.

[56] Niels Provos, Panayiotis Mavrommatis, Moheeb Abu Rajab, and Fabian Monrose. All Your iFRAMEs Point to Us. In *Proceedings of the 17th USENIX Security Symposium*, 2008.

[57] Craig Shue, Andrew J. Kalafut, and Minaxi Gupta. Abnormally malicious autonomous systems and their internet connectivity. In *IEEE/ACM Transactions of Networking 20(1)*, 2012.

[58] Antonio Nappa, M. Zubair Rafique, and Juan Caballero. The MALICIA dataset: identification and analysis of drive-by download operations. *International Journal of Information Security*, Vol. 24, No. 1, pp. 15–33, February 2015.

[59] B. Stone-Gross, C. Kruegel, K. Almeroth, A. Moser, and E. Kirda. Fire: Finding

rogue networks. In *Proceedings Annual Computer Security Applications Conference*, 2009.

[60] K. Zeeuwen, M. Ripeanu, and K. Beznosov. Improving Malicious URL Re-Evaluation Scheduling Through Empirical Study of the Malware Download Centers. In *Proceedings of the 2011 Joint WICOW/AIRWeb Workshop on Web Quality(WebQuality)*, 2011.

[61] B. Eshete and V. N. Venkatakrishnan. Webwinnow: Leveraging exploitkit workflows to detect malicious urls. In *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy(CODASPY)*, 2014.

[62] T. Nelms, R. Perdisci, M. Antonakakis, and M. Ahamad. WebWitness: Investigating, categorizing, and mitigating malware download paths. In *Proceedings of the 24th USENIX Security Symposium*, 2015.

[63] M. Akiyama, T. Yagi, K. Aoki, T. Hariu, and Y. Kadobayashi. Active Credential Leakage for Observing Web-Based Attack Cycle. In *Proceedings of Conference on Recent Advances in Intrusion Detection(RAID)*, 2013.

[64] Trend Micro: ServerProtect for linux. `http://www.trendmicro.co.jp/jp/business/products/splx/`.

[65] Symantec: Endpoint Protection. `https://www.symantec.com/ja/jp/theme.jsp?themeid=endpointsecurity_lineup_sep`.

[66] McAfee: Endpoint Protection Suite. `https://www.mcafee.com/jp/products/endpoint-protection-suite.aspx`.

[67] Kaspersky: Kaspersky Security. `http://home.kaspersky.co.jp/store/kasperjp/ja_JP/pd/ThemeID.37143200/productID.5064647600`.

[68] Caida: As Ranking. `http://as-rank.caida.org`.

[69] 2015 Internet Security Threat Report, Volume 20. `http://www.symantec.com/security\_response/publications/threatreport.jsp/`.

[70] Exploring the Blackhole Exploit Kit. `https://nakedsecurity.sophos.com/exploring-the-blackhole-exploit-kit/`.

[71] malware-traffic-analysis. `http://malware-traffic-analysis.net/`.

[72] Microsoft: SmartScreen URL Filter. `https://technet.microsoft.com/en-us/library/jj618329(v=ws.11).aspx`.

[73] McAfee: SiteAdvisor. `http://www.siteadvisor.com/howitworks/index.html`.

[74] Symantec: Web of Trust, Norton Safe Web. https://safeweb.norton.com/about?ulang=eng.

[75] Yandex: Safe Browsing. https://tech.yandex.com/safebrowsing/.

[76] Google Safe Browsing. https://developers.google.com/safe-browsing/.

[77] L. Metcalf and J. M. Spring. Blacklist Ecosystem Analysis: Spanning Jan 2012 to Jun 2014. In *Proceedings of the 2nd ACM Workshop on Information Sharing and Collaborative Security*, 2015.

[78] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon. From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware. In *Proceedings of the 21st USENIX Security Symposium*, 2012.

[79] M. Khrer, C. Rossow, and T. Holz. Paint it Black: Evaluating the Effectiveness of Malware Blacklists. In *Proceedings of the 17th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2014.

[80] Palo Alto Networks, Inc. https://www.paloaltonetworks.com/.

[81] T. Gerbet, A. Kumar, and C. Lauradoux. A Privacy Analysis of Google and Yandex Safe Browsing. In *Proceedings of the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2016.

謝辞

本研究の遂行にあたり，研究の進め方や方向性について日頃より熱心にご指導を頂いている後藤厚宏教授に深く感謝致します．また，合同ゼミで多くの助言をいただきました，田中英彦教授，大久保隆夫教授，大塚玲教授，橋本正樹准教授へ心より感謝申し上げます．統計的手法を指導頂いた廣松毅客員教授へ心より感謝申し上げます．本研究の有効性の評価に繋がる貴重なデータの解析をさせて頂いた JPCERT/CC の真鍋敬士氏，朝長秀誠氏に心より感謝申し上げます．NTT セキュアプラットフォーム研究所の秋山満昭氏には様々な助言を頂き心より感謝致します．博士後期課程への社会人入学をサポートしていただいた NTT コミュニケーションズ山下達也部長，宮川晋部長，畑田充弘課長に深く感謝致します．業務をサポートして頂いた技術開発部の皆様にも深く御礼申し上げます．様々な助言や研究への励みをいただきました情報セキュリティ大学院大学の皆様に重ねて感謝致します．最後に，様々な支えとなってくれた妻・娘に深く感謝致します．

研究業績

査読付学術論文

A-1 田中　恭之，後藤　厚宏．統計的方法を用いた未知マルウェア検出手法の提案と評価，情報処理学会論文誌，Vol.57，No.9，pp. 2003-2011，2016．(受付日 2015 年 11 月 7 日, 採録日 2016 年 6 月 2 日)

A-2 田中　恭之，後藤　厚宏．悪性文書ファイル内の ROP 攻撃コード静的判定手法，情報処理学会論文誌，Vol.56，No.9，pp. 1693-1705，2015．(受付日 2014 年 11 月 27 日, 採録日 2015 年 6 月 5 日)

A-3 Y. Tanaka, M. Akiyama, and A. Goto. Analysis of malware download sites by focusing on time series variation of malware, ELSEVIER, Journal of Computational Science, Volume 22, Pages 201-313, 2017. (Received at Editorial Office : 14 Sep 2016, Article accepted for publication : 28 May 2017)

査読付国際会議（プロシーディングとして採録）

B-1 Y. Tanaka and A. Goto. n-ROPdetector: Proposal of a method to detect the ROP attack code on the network. In Proceeding of the 7th Workshop on Cyber Security Analytics, Intelligence and Automation(Safeconfig), page 33-36. ACM, November 2014.

B-2 Y. Tanaka and A. Goto. Analysis of Malware Download Sites by Focusing on Time Series Variation of Malware. In Proceeding of Computers and Communication (ISCC), 2016 IEEE Symposium, 2016.

B-3 Y. Tanaka and A. Goto. Suspicious FQDN Evaluation based on Variations in Malware Download URLs. In Proceeding of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM), IEEE/ACM, 2017.

B-4 Y. Tanaka and A. Goto. SEEDSTRAINER: An Approach to Improve the Hit Ratio of Malicious Candidate URLs. In Proceeding of the IEEE Real-Time System Symposium(RTSS), IEEE, 2017.

査読付国際会議（プレゼン資料による講演）

C-1 Actual Condition Survey of Malware Download Sites for A Long Period. Apricot 2016. https://conference.apnic.net/41/program#sessions/networksecurity

C-2 Characteristic of Malware Site and its Blocking Countermeasure. Apricot 2017. https://2017.apricot.net/program/schedule/#/day/10/network-security-1

国内会議

D-1 田中　恭之，後藤　厚宏．攻撃コードの特徴から見た対策の検討, 暗号とセキュリティシンポジウム 2014(SCIS2014), 4C2-4, January 2014.

D-2 田中　恭之，後藤　厚宏．ROP 攻撃コード検出による悪性文書ファイル特定手法の提案．第 66 回 CSEC 研究発表会, July 2014.

D-3 田中　恭之，後藤　厚宏．ネットワーク側で ROP 攻撃コードを検出する手法の提案．コンピュータセキュリティシンポジウム 2014・マルウェア対策研究人材ワークショップ 2014 (CSS2014/MWS2014) , 3A2-1, October 2014.

D-4 Y. Tanaka and A. Goto. Suspicious FQDN Evaluation based on Variations in Malware Download URLs. 暗号とセキュリティシンポジウム 2017(SCIS2017), 2C4-5, January 2017.