APIコールの補完による機能推定

指導教員

甲斐 博 准教授

令和4年 年 2 月 10 日提出

愛媛大学工学部情報工学科

情報システム工学分野

ソフトウェアシステム研究室

開原　悠介

目 次

第1章　研究の背景

インターネットの普及に伴いマルウェアが急増しており，マルウェアによる被害が深刻化している．マルウェアの感染を防ぐには，アンチウイルスソフトウェアのインストールが不可欠である．しかし，マルウェアは高度化しており，アンチウイルスソフトウェアでは検出されないものもある．多くのマルウェアは，暗号化や難読化などの手法を使用することで, アンチウイルスソフトウェアによる検出を回避している．

また，現在のマルウェアの多くは，ツールによって自動的に生成された既存のマルウェアの亜種である．大量の亜種を効率的に解析するには，事前に機能を推定することが効果的である．また，マルウェアのすべての動作が解析されていなくても，動的解析の初期段階でマルウェアの機能を推定できることが望ましい．

マルウェアの機能推定に関する関連研究について, 大久保らの研究 [1] では，マルウェアの静的解析結果からバイトコードを抽出し，LCS や N-gram を用いて検体間の類似度を求め, 検体の推定機能に対してポイントを付与し判別分析法により機能推定を行う方法を提案した．しかし, この手法では, 機能推定の精度があまり高くないことと, 機能推定に時間がかかるというという問題点があった. その問題点を受けて, 児玉らの研究 [2] では, 動的解析結果から得られる API コー ル列と保有機能の関係を学習し機能推定を行う方法を提案した. しかし, この手法では, 動的解析では動作環境によって挙動が変わってくる場合があるため API コールが呼び出されない場合に対応できないという問題点がある.

そこで, 本研究では, 動的解析の段階で欠損したと考えられる API を補う手法を提案する. 提案手法では, 同一ファミリの検体は類似した機能を保有するという性質を考慮して, 欠損した可能性が高いと考えられる API をランダムに補う機能推定の検討を行った. 機能推定については，SVM を用いて API コール列の特徴と保有機能の関係を学習することにより推定器を作成した. 各機能の推定結果から平均正解率 89.25%, 平均再現率 86.67%, 平均適合率 87.36%,F 値 87.01%が得られた. 従来手法と比較すると保有機能の検出を特に F 値においてより高い値を得ることができた．以下の章構成について述べる．第 2 章では機械学習について述べる．第 3 章ではマルウェアの解析手法と FFRI Dataset について述べる．第 4 章では従来手法と提案手法について述べる．第 5 章では詳細な実験内容と実験結果について述べ

る．第 6 章では結論を述べる．

第2章　機械学習と評価手法

本章ではマルウェアの機能推定を行う際に使用した SVM [3] のアルゴリズムおよび評価指標について述べる.

2.1　機械学習 (Machine Learning)

機械学習 (Machine Learning) は, 電子メールのウイルス対策や写真の自動タグ付け, 映画の推薦など身近なあらゆるところで利用されている. 機械学習は, 人工知能 (Artifical Inteligence) の一種であり, コンピュータが過去のデータに基づいて未来を予測することを可能にする. また, 機械学習のほとんどの問題は以下の 3 つの主なカテゴリのいずれかに属する.

教師あり学習

教師あり学習では, それぞれのデータ点にカテゴリラベルや数値ラベルが付与されている. カテゴリラベルは, 例えば, 犬や猫を含む画像に対する,「犬」や「猫」などのようなラベルである. 数値ラベルは, 中古の車につけられた売値のようなものを指す. 教師あり学習の目的は, 大量のラベル付きのデータ例（学習データ）に基づいて新しく得られたデータのラベルを予測することでなる. データ点に付与されたラベルがカテゴリラベルの場合は, 分類問題, 数値ラベルの場合は, 回帰問題と呼ばれる.

教師なし学習

教師なし学習では, データ点はラベルを持たない. 教師なし学習の目的は, 何らかの方法で, データをまとめる, もしくは, データが持つ構造を見つけることである.

強化学習

強化学習では, 与えられたそれぞれのデータ点に対する動作を選択するアルゴリズムを学習する. これはロボティクス分野でよく用いられる手法で, ある時刻での各種センサからの出

力をデータ点として, ロボットの次の動作を決定する場合などに利用される. また,Internet of Things(IoT) への応用も可能である. この場合アルゴリズムは少し未来の時点で動作選択の適切さを示す報酬信号を受け取り, より高い報酬信号を得るために動作選択の戦略を修正する.

2.2　SVM (Support Vector Machine)

SVM は,2 クラス分類の線形識別関数を構築する機械学習モデルの一種である. 本研究では分類を扱った.SVM の目的は, 常に分類エラーを最小化することであり, あるクラスのほかのクラスに対するデータ点のマージン (決定境界とクラス端点間の距離) を最大化することで決定境界を引く. 線形決定境界を用いてデータを適切に分離できない場合, 元の特徴の非線形結合を作る. つまり, データが線形分離可能となるような, より高次元の空間にデータを写像（例えば,2 次元から 3 次元へ）と等価である. そうして, 高次元空間において線形決定境界 (すなわち,3 次元においては平面) を探索する. しかし, この写像アプローチは, 次元の間で数学的な写像を行うために, 多くの項を導入する必要があるため, 次元が大きくなると実用的でない点が問題となる. その問題点を解決するための関数にカーネル関数を用いて計算する. カーネル関数の例に, 放射基底関数 (Radial Basis Function;RBF) やガウス関数 (釣鐘曲線) がある.

2.3　評価手法

本研究で使用する評価指標について述べる. モデル性能を評価するための混同行列を表 2.1 に示す. 混同行列は, 機能 F を保有する検体に対して, 正しく保有していると予測した場合の数 (true positive), 機能 F を保有する検体に対して, 間違って保有していないと予測した場合の数 (false positive), 保有していない検体に対して, 正しく保有していないと予測した場合の数 (true negative), 保有していない検体に対して, 間違って保有していると予測した場合の数 (false negative) をまとめると, 次のような 2 × 2 の行列にまとめることができる.

表 2.1　混同行列

		機能 F の有無	
		Positive	Negative
機械学習モデルの予測	Positive	True Positive(TP)	False Positive(FP)
	Negative	False Negative(FN)	True Negative(TN)

機械学習のモデルの評価には正解率 (*Accuracy*)，再現率 (*Recall*)，適合率 (Precision),F 値 (*F − measure*) の 4 種類を使用する.

正解率（*Accuracy*）は式 2.1 で定義されていて, 機能 F について推定した全ての検体のなかで，モデルの予測と機能 *F* の有無が一致していた割合を表す.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.1}$$

再現率 (*Recall*) は，式 2.2 で定義されていて, 機能 *F* を保有する検体のうち，モデルが保有していると判定した割合を表す.

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

適合率 (*Precision*) は，式 2.3 で定義されていて, 機能 *F* を保有すると推定した検体のうち，実際に機能 *F* を保有していたものの割合を表す.

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

F 値 (*F − measure*) は，式 2.4 で定義されていて, 再現率と適合率の調和平均を表す.

$$F = \frac{2Recall \times Precision}{Recall + Precision} \tag{2.4}$$

第3章　マルウェアの解析手法と FFRI Dataset

本章では，マルウェアの解析手法と，研究者向けにデータセットとして配布されている FFRI-Dataset について述べる．

3.1　マルウェアの解析手法

マルウェアの挙動を知るための解析手法は大きく分けて, 表層解析, 静的解析, 動的解析の 3 つに分けられる [4].

表層解析

表層解析は, ファイル自体が悪性であるかどうかの情報収集や，ファイルのメタ情報の収集を目的として行う解析プロセスである．アンチウイルスソフトで判定を行い，既知のマルウェアであるかを判断や，ファイルタイプや動作する CPU アーキテクチャの情報を収集, 特徴的なバイト列（通信先 URL やマルウェアが用いるコマンド等）の収集を行う．表層解析では，解析対象に対してツールを適用して情報を取得する．その他の解析のように，実際にマルウェアを動作させたりマルウェア内のプログラムコードを分析はしない．

動的解析

動的解析（ブラックボックス解析）は, マルウェアが実際に動作した際に端末上にどのような痕跡が残るのか，またどのような通信が発生するのかの情報収集を目的とした解析プロセスである．監視ツールをインストールした環境で実際にマルウェアを動作させ，ファイルやレジストリアクセスの情報の収集や, 通信を監視して通信先の IP アドレスやドメイン，URL，通信ペイロードなどの情報を収集を行う．動的解析では，プログラムコードを詳細に分析しないためブラックボックス解析とも呼ばれる．

静的解析

静的解析（ホワイトボックス解析）は, 逆アセンブラやデバッガを用いてマルウェアのプログラムコードを分析し，具備されている機能や特徴的なバイト列など詳細な情報を収集することを目的とした解析プロセスである．動的解析で実行されなかったコードを分析し, 潜在的に保有している機能を明らかにし，マルウェア独自の通信プロトコルや通信先生成アルゴリズムのような動的解析だけでは特定が難しい情報の収集を行う．静的解析では，プログラムコードを詳細に分析するためホワイトボックス解析とも呼ばれる．

解析の難易度は, 表層解析, 動的解析, 静的解析の順に上がっていく. そのため, 解析対象の数という観点で見た場合, 表層解析が最も多くのマルウェア解析することができ, 静的解析では限られた数のマルウェアしか解析することができない. 表 3.1 に示すように, 解析には長所と短所がある. 各解析の長所・短所を理解し, 解析の目的の応じて解析プロセスの選択・組み合わせを行うことが重要である.

表 3.1　各解析プロセスの長所と短所

	長所	短所
表層解析	• 短時間に解析結果を取得できる • 解析者に要求されるスキルレベルは高くない	• 得られる情報が限定的である • 難読化されたマルウェアからは十分な解析結果が得られない
動的解析	• 解析者に要求スキルレベルは高くない • 難読化されたマルウェアからも解析結果を取得できる • 短時間で解析結果を取得できる	• 安全な解析環境を構築する必要がある • 解析妨害機能を有する検体を十分に解析できないことがある • 解析時に実行されなかったコードの振る舞いはわからない
静的解析	• 動的解析で実行されないコードの動作を把握できる • 具備された機能の詳細なアルゴリズムを解明できる	• 解析者に要求されるスキルレベルが高い • 詳細な解析結果を取得するのに時間がかかる

3.2　FFRI Dataset

本研究では, 情報処理学会コンピュータセキュリティ研究会マルウェア対策人材育成ワークショップ (MWS) が研究者向けに配布している FFRI Dataset を使用する [5]. 現時点では,2013-

2017 では動的解析ログ,2018-2021 では表層解析ログが提供されている. 本研究では, マルウェアの機能を推定するため, マルウェアの機能概要が出力されている FFRI Dataset 2016, 2017 を実験に使用する.

FFRI Dataset 2016 は 2016 年の 1 月から 2016 年の 3 月までに収集された検体, 計 8,243 検体分の動的解析ログである. これらの検体は,PE 形式かつ実行可能なものであり, それぞれ,10 ベンダー以上でマルウァア判定を受けている. 仮想環境内でマルウェアを実行し, 実行時のふるまいを 90 秒モニタリングしたログを json 形式で保存している. 具体的なデータ項目は表 3.2 [6] である.

表 3.2　データ項目 2016

項目	内容
info	解析の開始, 終了時刻,id 等（id は 1 から順に採番）
signatures	ユーザー定義シグニチャとの照合結果（今回は使用無）
virustotal	VirusTotal の検査履歴との照合結果（検体の MD5 値に基づく）
static	検体のファイル情報（インポート API, セクション構造等）
dropped	検体が実行時に生成したファイル
behavior	検体実行時の API ログ（PID,TID,API 名, 引数, 返り値等）
processtree	検体実行時のプロセスツリー（親子関係）
summary	検体が実行時にアクセスしたファイル, レジストリ等の概要情報
target	解析対象検体のファイル情報（ハッシュ値等）
debug	検体解析時の Cuckoo Sandbox のデバッグログ
strings	検体中に含まれる文字列情報
network	検体が実行時に行った通信の概要情報

FRI Dataset 2017 は 2017 年の 3 月から 2017 年の 4 月までに収集された検体, 計 6,251 検体分の動的解析ログである. これらの検体は,PE 形式かつ実行可能なものであり, それぞれ,15 ベンダー以上でマルウァア判定を受けている. 仮想環境内でマルウェアを実行し, 実行時のふるまいを 90 秒モニタリングしたログを json 形式で保存している. 具体的なデータ項目は表 3.3 [7] である.

表 3.3　データ項目 2017

項目	内容
info	解析の開始, 終了時刻,id 等
signatures	ユーザー定義シグニチャとの照合結果
virustotal	VirusTotal から得られる情報
static	検体のファイル情報（インポート API, セクション構造等）
dropped	検体が実行時に生成したファイル
behavior	検体実行時の API ログ（PID,TID,API 名, 引数, 返り値等）
target	解析対象検体のファイル情報（ハッシュ値等）
debug	検体解析時の Cuckoo Sandbox のデバッグログ
strings	検体中に含まれる文字列情報
network	検体が実行時に行った通信の概要情報

図 3.1 は，FFRI Dataset から取得できる，本実験で使用するファミリ Backdoor.Win32.Androm のある検体の動的解析結果を json ファイルに出力したものである．本研究では，マルウェアの動的解析結果から，マルウェアの実行時に呼び出された Win32 API(以下，API) と動作概要を抽出する．ファミリ名については 5 行目の”Backdoor.Win32.Androm”といったファミリ名と定義する．

API については，21,27 行目の ”NtAllocateVirtualMemory”, ”LdrGetDllHandle”といった API 関数名を抽出する．

動作概要については，35〜38 行目の”file_created”, ”file_recreated”,”directory_created”,”dll_loaded” といった動作概要名を抽出する．また，これらの動作概要名をこの検体が保有する機能として定義する．

```
{
"virustotal":{
               "scans" {
                   "Kaspersky":{
                          "result":"Backdoor.Win32.Androm.jjcl"
                          ...
                                    }.
                           }.
                         ...
                         }
    ...
    "behavior": {
        "processes":[
            {...},
            {
                "process_path": ... ,
                "calls": [
                    {
                        "category": "process",
                        ...
                        "api": "NtAllocateVirtualMemory",
                        ...
                    },
                    {
                        "category": "system",
                        ...
                        "api": "LdrGetDllHandle",
                        ...
                    },
                    ...
                ], ...
            }
        ], ...
        "summary": {
            "file_created": [...],
            "file_recreated": [...],
            "directory_created": [...],
            "dll_loaded": [...],
 ...
        }, ...
    },
    ...
}
```

図 3.1　json ファイルの内容

第4章　APIコールの補完と機能推定の実験

本章では, 児玉らによって提案されたマルウェアの動的解析結果を用いた手法と, 本研究で提案する手法について述べる.

4.1　従来手法

児玉らの手法ではマルウェアの動的解析結果から得られる API コー ル列と保有機能の関係を学習し機能推定するという手法が提案された. その手法は以下の通りである.

動的解析結果の API コール列を $C(c_1, c_2, \cdots, c_m)$ とする．ここで，$c_t(t = 1, 2, \cdots, m)$ は API 関数名を表す文字列とする．API コール列 C に対して，API c_t の有無 $e_{c_t}(C)$ を

$$e_{c_t}(C) = \begin{cases} 1 & \text{有} \\ 0 & \text{無} \end{cases}$$

とした時，m 次元特徴ベクトル $V(C)$ を

$$V(C) = (e_{c_1}(C), e_{c_2}(C), \cdots, e_{c_m}(C)).$$

と定義する．K 種類の推定機能 $F_1, F_2, ..., F_K$ とした時，推定する機能 F_k に対する分類器は，動的解析結果の API コール列から得られる特徴ベクトル $V(C)$ を入力として，推定機能 F_k を保有する又は保有しないに分類する．

N 個の動的解析結果 $D_1, D_2, ..., D_N$ に対して，得られる API コール列を $C_1, C_2, ..., C_N$ とする．特徴ベクトル V を入力し，機能 F_i を保有する又は保有しないを分類する分類器の作成方法は以下の手順となる．

(1) D_l に対して，機能 F_i を保有する場合 $L = 1$，しない場合 $L = 0$ とする．また，API コール列 C_l を抽出する．抽出したものを (L, C_l) で表現する．

(2) (L, C_l) に対して，C_l を用いて 3 つの方法により特徴ベクトル $V(C_l)$ を作成し，$(L, V(C_l))$ を得る．

(3) $(L, V(C_l))$ を SVM の教師データとして学習させ，機能 F_i を保有する又は保有しないを分類する分類器を得る．

この分類器を利用して，機能が未定義のマルウェアの特徴ベクトル $V(C)$ を与えたとき，K 種類の各機能を保有する又は保有しないを推定するための手順は以下の通りである．

1. K 個の分類器に特徴ベクトル $V(C)$ を入力し，$(A_1, A_2, \cdots, A_K)$ を得る．
2. $A = 1$ の時，機能を保有すると推定する．$A = 0$ の時，機能を保有しないと推定する．

4.2　従来手法の課題

動的解析では動作環境によって挙動が変わってくる場合があるため API コールが呼び出されない場合がある図 4.1 は左図では NtCriateFile が呼び出されているが, 右図では呼び出されていない. 従来手法ではこのような API コールが欠損している状況に対応できないという課題がある．

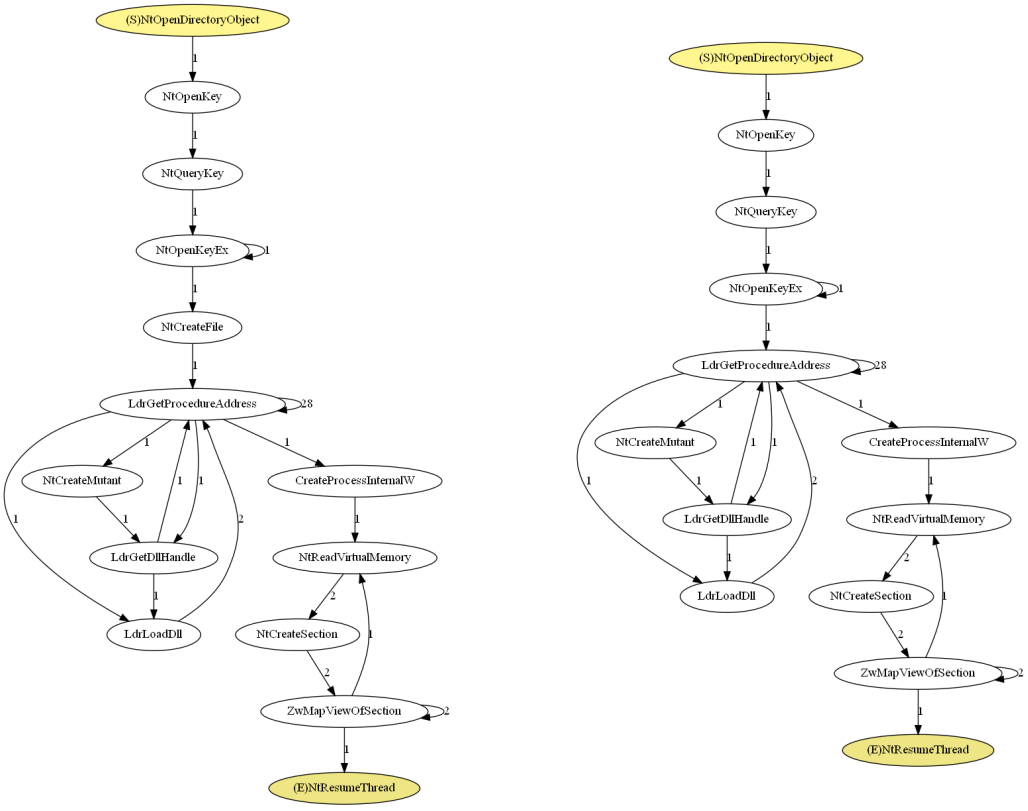

図 4.1　ファミリ Backdoor.Win32.Androm の 2 検体

4.3　提案手法

本研究では欠損した可能性が高い API コールを補うことによって機能推定の精度の向上を図る手法を提案する. 動的解析結果のファミリ名を g とする.
g から n 個の補完用の検体を取得する．n 個の検体を

$$g_1, g_2, \cdots, g_n$$

と表す. また, 検体 g_1 から g_n が呼び出す k 種類の API コールを

$$a_1, a_2, \cdots, a_k$$

と表す. また g_1 から g_n が各 API コールを呼び出す回数を,

$$b_1, b_2, \cdots, b_k$$

と表す. 以上の情報を使って, 以下の 3 つの手法を用いて API コールを補完する.

提案手法 1　a_1 から a_k から等確率でランダムに API を補完する.

提案手法 2　$B = b_1 + b_2 + \cdots + b_k$ とすると,b_i/B の確率で a_i を補完する. 例えば,API コールの数 $k = 4$ のとき,$b_1 = 1, b_2 = 1, b_3 = 2, b_4 = 3$ とすると,a_1, a_2, a_3, a_4 が補完される確率はそれぞれ $1/7, 1/7, 2/7, 3/7$ である.

提案手法 3　考え方は提案手法 2 と同様であるが. 呼び出し回数が少ない API コールを優先して補完する. 例えば,API コールの数 $k = 4$ のとき,$b_1 = 1, b_2 = 1, b_3 = 2, b_4 = 3$ とすると,a_1, a_2, a_3, a_4 が補完される確率はそれぞれ $3/7, 3/7, 2/7, 1/7$ である.

提案方法 1〜3 に対し, 従来手法と同様の方法で, 特徴ベクトル V_1〜V_3 を作成する. 以降の分類器の作成, 機能推定の手順は従来手法と同様である.

4.4　検体の取得・推定機能の定義・データセットの作成

マルウェアの機能推定をファミリごとに行うため,FFRI Dataset 2016, 2017 を「kaspersky 社」命名のファミリ名ごとに分類した. 実験の手法は提案手法に述べたとおりである．また, 本実験で使用したファミリは Backdoor.Win32.Androm で, 取得した検体数は $n = 10$, 実験用検体に付加する API の個数を $i = 3$ として特徴ベクトルを作成した. 取得した検体の呼び出

す API から呼び出されている全ての API を取得すると，合計で 204 種類の API が得られた．取得した全ての検体の動作概要から保有する機能を取得すると，合計で 26 種類の機能が得られた．このことから，26 種類の機能を本研究で推定する機能として定義した．推定する機能の一覧を表 A.1 に示す．ここで，呼び出される API 関数名は 204 種類であることから，特徴ベクトル V_1, V_2, V_3 の次元数は 204 次元としている．また, 本実験では不均衡データを取り扱うため, 以下の imbalanced-learn が公開するアンダーサンプリングモジュールを使用した [8].

$$imblearn.under\_sampling.RandomUnderSampler$$

4.5　機械学習

本研究では，データセットの特徴ベクトルを入力，ラベルを出力として機械学習モデルの学習と評価を行う．ここで，データセットは推定する機能を保有する検体，保有しない検体の特徴ベクトル，ラベルにより構成され，機械学習モデルは機能を保有する又は保有しないを推定するため，2 値分類を行うモデルを推定する機能（26 種類）の数だけ用意する．

また, 本研究では，10 分割交差検証により分類器の検証を行う．10 分割交差検証は, データセットを 10 個に分割してそのうち 1 つをテストデータに残りの 9 個を学習データとして正解率の評価を行う. これを 10 個のデータすべてが 1 回ずつテストデータになるように 10 回学習を行なって精度の平均をとる手法である. 本研究で使用した SVM の実装では，scikit-learn が公開する分類問題を扱う SVM アルゴリズムである以下のモジュールを使用する [9].

$$sklearn.svm.SVC$$

このモジュールの全てのハイパーパラメータについてデフォルトの値を使用した ($C = 1.0, kernel = 'rbf', gamma = 'scale'$).

4.6　実験結果と考察

10 分割交差検証を行い，各手法ごとに正解率，再現率，適合率,F 値の平均値を算出した結果を表 4.1 に示す．また, 本実験では, 精度の信頼性の観点からサンプル数が 50 以上の機能を対象とした.

表 4.1　実験結果

	Accuracy Avg(%)	Recall Avg(%)	Precision Avg(%)	F-measure
従来手法	89.26	86.62	86.57	86.59
V_1	88.85	86.16	86.93	86.54
V_2	89.25	86.67	87.36	87.01
V_3	88.74	85.95	86.94	86.44

実験の結果から, 正解率は従来手法が最も高いことと, 再現率, 適合率,F 値は V_2 が最も高いことがわかる. モデルの評価において特に重要とされる F 値の最も高い推定方法が V_2 であることから, 存在数の多い API を高確率で補完することが, 機能推定に対して有効であることがわかる. 以上のことから従来手法と比べて提案手法が高い F 値でマルウェアの機能推定を行えることがいえる.

従来手法と F 値の最も高い推定方法である V_2 の正解率，再現率，適合率,F 値の平均値を各機能ごとに算出した結果を, 表 4.2, 表 4.3 に示す.

また, 表 4.2, 表 4.3 について，各評価値の単位を%とする.

表 4.2　従来手法

	Accuracy Avg	Recall Avg	Precision Avg	F-measure
command_line	94.37	93.35	90.29	91.79
connects_ip	95.10	81.19	87.55	84.24
directory_created	88.76	88.38	82.01	85.08
directory_enumerated	86.79	82.84	93.60	87.89
file_copied	94.86	88.90	95.46	92.07
file_created	82.88	87.38	74.54	80.45
file_deleted	85.79	71.89	76.89	74.31
file_exists	92.15	92.77	93.10	92.94
file_failed	87.51	85.27	96.66	90.61
file_read	96.33	96.72	97.29	97.00
file_recreated	83.33	80.97	87.46	84.09
file_written	84.55	91.58	75.23	82.93
guid	84.84	90.81	78.09	79.43
mutex	84.32	82.57	82.93	82.75
regkey_deleted	95.35	89.11	90.29	89.70
regkey_opened	90.72	87.76	98.38	92.77
regkey_written	90.73	91.76	87.09	89.37
resolves_host	88.23	85.89	71.36	77.95

表 4.3　v_2

	Accuracy Avg	Recall Avg	Precision Avg	F-measure
command_line	96.07	94.67	94.20	94.43
connects_ip	94.50	79.52	86.87	83.03
directory_created	87.15	86.43	80.88	83.56
directory_enumerated	88.20	85.55	94.49	89.80
file_copied	93.96	87.14	94.98	90.89
file_created	93.96	87.14	94.98	84.88
file_deleted	83.20	84.91	68.14	75.61
file_exists	92.65	93.01	94.21	93.61
file_failed	88.18	86.64	96.80	91.44
file_read	95.27	96.21	96.31	96.26
file_recreated	82.91	79.60	78.20	84.07
file_written	95.53	87.89	92.99	93.08
guid	87.65	82.69	84.19	83.43
mutex	81.38	79.60	78.20	78.89
regkey_deleted	95.53	87.8	92.99	90.37
regkey_opened	90.02	88.82	96.79	92.63
regkey_written	89.25	85.29	89.46	87.33
resolves_host	88.21	87.22	73.29	79.65

以上の結果から従来手法と V_2 の F 値を機能ごとに比較すると, (command_line,directory_enumerated, file_created, file_deleted, file_exists,file_failed,file_written,guid, regkey_deleted,resolves_host) の機能において V_2 の F 値が高いことがわかる. 補完済みの API コール列が補完前の API コール列と比べ保有機能の特徴をとらえているため, 補完に用いた API と保有機能が深く関係していると考えられる. 本実験で, 補完に用いたファミリ Backdoor.Win32.Androm の 10 検体が呼び出した API とその回数は表 4.4 である.

表 4.4　補完に用いた API とその存在数

API	存在数
NtClose	31
NtProtectVirtualMemory	25
LdrGetDllHandle	20
GetSystemMetrics	14
NtOpenKeyEx	12
NtAllocateVirtualMemory	11
NtQueryValueKey	10
NtMapViewOfSection	9
NtOpenSection	8
NtOpenKey	7
FindResourceA	6
LdrUnloadDll	5
NtOpenFile	4
SetWindowsHookExA	2
NtFreeVirtualMemory	2
NtCreateSection	2
GetCursorPos	1
RegCloseKey	1
GetForegroundWindow	1
NtCreateFile	1
NtQueryAttributesFile	1
NtCreateMutant	1
CoInitializeEx	1
OpenSCManagerA	1
EnumWindows	1

第5章　結論

本研究では, 同一ファミリから欠損した可能性が高い API コールを補い,SVM を用いて機械学習することにより, 機能推定を行った. 特徴ベクトル V_2 を用いた場合の提案手法の実験結果として，各機能の推定結果から平均正解率は約 89.25%，平均再現率は約 86.67%, 平均適合率は約 87.36%,F 値は約 87.01%, が得られた. 従来手法と比較すると保有機能の検出を特に F 値においてより高い値を得ることができた.

また, 本研究ではファミリ Backdoor.Win32.Androm のみを用いて実験を行ったが, ほかのファミリを用いての比較・検討を行うことが今後の課題である.

謝辞

本研究を行うにあたり，常日頃より懇切丁寧に御指導いただきました高橋寛教授, 甲斐 博 准教授, 王森レイ講師に心より御礼申し上げます. また, 本研究に際しご審査頂きました遠藤 慶一准教授, 宇戸 寿幸准教授に深く御礼申し上げます. 最後に, 日頃から助言や励ましをいただきました諸先輩方, 並びに同研究室の皆様に深く御礼を申し上げます.

参考文献

［1］ 大久保諒，伊沢亮一，森井昌克，井上大介，中尾康二: マルウェアの類似度に基づく機能推定，情報処理学会コンピュータセキュリティシンポジウム 2013(CSS2013), pp.193-196, 2013.

［2］ 児玉光平:機械学習を用いたマルウェア機能推定に関する研究, 愛媛大学修士論文,2020.

［3］ Michael Beyeler:Machine Learning for OpenCV Inteligent image processing with Python,2018.

https://analysis-navi.com/?p=550

［4］ 八木毅，青木一史，秋山満昭，幾世知範，高田雄太，千葉大紀: 実践サイバーセキュリティモニタリング, コロナ社，2016.

［5］ MWS Datasets.

https://www.iwsec.org/mws/datasets.html

［6］ FFRI:FFRI Dataset 2016 のご紹介 - IWSEC .

http://www.iwsec.org/mws/2016/20160530-ffri-dataset-2016.pdf

［7］ FFRI:FFRI Dataset 2017 のご紹介.

https://www.iwsec.org/mws/2017/20170606/FFRI_Dataset_2017.pdf

［8］ imblearn.under_sampling.RandomUnderSampler.

https://glemaitre.github.io/imbalanced-learn/generated/imblearn.under_sampling.RandomUnderSampler.html

［9］ Support Vector Machines — scikit-learn 1.0.2 documentation.

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html