情報処理学会論文誌 Vol.61 No.12 1792–1801 (Dec. 2020)

APIグループ間の相関性とフォルダ操作頻度に基づくマルウェア分類手法の提案

周 家興[1,a)] 廣瀬 幸[1] 柿崎 淑郎[1] 猪俣 敦夫[2] 寺田 真敏[1]

受付日 2020年3月10日, 採録日 2020年9月10日

概要：マルウェア攻撃手法の進歩にともない，既存のマルウェア検知および分類手法では対応が難しくなってきている．本論文ではマルウェアの動作パターンと特徴ある動作とを特徴量とする機械学習による分類手法を提案する．本提案方式では，動作パターンとしてAPIグループ間の相関性，特徴ある動作としてフォルダ操作頻度を使用する．さらに，研究用データセットFFRI Dataset2016を使用した分類実験を通して手法の有効性を評価する．分類実験の結果，不均衡データであることを考慮したオーバサンプリングの場合，ベンダが提供するマルウェア系列との一致度は99%となり，動作を特徴付けるAPIとフォルダ操作頻度に着目した機械学習による分類手法が有効であることを示した．今後は，検体数を増やして不均衡データが分類に対する影響を抑えるとともに，様々な特徴量を使用して分類精度を向上させ，未知のマルウェア分類を対応できるように取り組んでいく．

キーワード：マルウェア，不均衡データセット，オーバサンプリング

Proposal of Malware Classification Using Correlation between API Groups and File Path Operation Frequency

Jiaxing Zhou[1,a)] Miyuki Hirose[1] Yoshio Kakizaki[1]
Atsuo Inomata[2] Masato Terada[1]

Received: March 10, 2020, Accepted: September 10, 2020

Abstract: With the progress of malware attack techniques, it is becoming difficult to respond with existing malware detection and classification techniques. In this paper, we propose a classification method based on machine learning that uses the behavioral patterns and characteristic behaviors of malware as features. In the proposed method, the correlation between API groups presents the malware's operation pattern, and the folder operation frequency presents the malware's characteristic operation. In addition, efficacy is evaluated through classification experiments using the research dataset FFRI Dataset 2016. The experimental results show that in the case of oversampling considering imbalanced data, the degree of coincidence with the malware family provided by the vendor is 99%, and a classification method using machine learning that focuses on the API that characterizes the behavior and the folder name has been shown to be valid. In the future, we will work to increase the number of specimens to reduce the influence of imbalanced data on classification and to improve the classification accuracy by using various features to support unknown malware classification.

Keywords: malware, imbalanced dataset, oversampling

1. はじめに

マルウェアは，コンピュータの遠隔操作，機密情報の収集などの機能を備えたソフトウェアであり [1]，ウイルス，ワーム，バックドア，トロイの木馬，スパイウェアなどの多種類が存在する [2]．マルウェアはサイバー攻撃にも使用さ

[1] 東京電機大学
Tokyo Denki University, Adachi, Tokyo 120–8551, Japan
[2] 大阪大学
Osaka University, Suita, Osaka 565–0871, Japan
[a)] syuu@isl.im.dendai.ac.jp

れており，個人 PC を狙うだけではなく，組織 PC も狙ったものも存在する．MalwareBytes によれば，2018 年の個人 PC でのマルウェアの検知数は 2017 年より 3%減少しているが，対照的に，2018 年の組織 PC での検知数は 2017 年より 79%増加したと報告している．特に組織 PC でのバックドアの検知数は 2017 年より 173%増加している [3]．

マルウェアの被害を低減させるためには，どのような被害につながるものかを特定することが重要であり，マルウェアの検知だけではなく，分類することも必要となる．マルウェアの分類により，マルウェアの系列，機能，目標，感染経路などの情報と紐づけることができるため，マルウェア攻撃対策の促進にもつながる．さらに，それらの情報からマルウェアによる被害種別を想定できる．マルウェアに関する研究のうち，動的解析手法によるアプローチでは，マルウェアが呼び出す API からマルウェアの動作を特徴付けることで検知と分類が試みられてきた [4]．また，マルウェアが呼び出す Windows API の頻度 [5]，API 呼び出しの相関性 [6]，マルウェアのファイルシステム操作 [7] などに着目した研究もされている．

本論文では，既存研究ならびに著者らの過去研究 [8] をふまえて，マルウェアの動作パターンと動作上の特性とを組み合わせ，かつこれらを特徴量とする機械学習による分類手法を提案する．動作パターンとして API グループ間の相関性，動作上の特性としてフォルダ操作頻度を使用し，研究用データセット FFRI Dataset2016 を使用した分類実験を通して提案方式の有効性を示す．

マルウェア分類における本研究の貢献は，次のとおりである．

1) マルウェアの動作間の関係性を表現する方法として，API グループ間の関係を Pearson 相関係数で数量化した．
2) マルウェア系列ごとの動作を特徴付ける重要なフォルダ名を明らかにした．
3) 各マルウェア系列の分類にあたり各 API グループおよびフォルダ名の寄与度を明らかにした．

本論文の構成は次のとおりである．2 章でマルウェア検知および分類に関する既存研究をまとめ，3 章で本研究の提案手法の詳細，アルゴリズムを示す．4 章で提案方式の実験環境と結果について述べ，5 章でまとめと今後の課題を示す．

2. 関連研究

マルウェア解析方法には動的解析とハイブリット解析がある [9]．Idika らは従来のマルウェアの検出手法調査し [10]，マルウェア分析と分類手法の比較などに取り組んでおり [11]，マルウェアの検知と分類そのものも研究として進んでいる．

2.1 動的解析

1) マルウェアが呼び出す API に着目した研究

Hampton らは仮想環境でのランサムウェアと正常なソフトウェアの動作のベースラインを定義した [12]．動作のベースラインとは検体によって呼び出された API の種類である．Hampton らの研究により，ランサムウェアの検知と分類に役立つと考えられる Windows API コールを確定できた．Dahl らはプロセスのメモリで観察された null-terminated パターン，システム API コールの tri-grams および API の 1 つ目のパラメータを特徴量として使用し，ランダムプロジェクションでデータセットの次元数を削減して，最後に neural network を用いてマルウェアを分類した [13]．Medhat らは動的解析で得られた API コール，拡張子，暗号署名およびキーワードの YARA ルールを特徴量として，既存のランサムウェアの分類だけではなく，未知のランサムウェアの分類についても言及している [14]．Kakisim らはマルウェア検知手法を改善し，最適な動的特徴量を見つけるために，サンドボックスツールで収集したマルウェアの API コールと操作ログを特徴量として，多数の分類器に適用することで，これらの特徴量と分類器の効果を評価した [15]．

2) 機械学習を活用した研究

Kolosnjaji らの研究でマルウェアに呼び出された API コールを特徴量としている．API コールを数値特徴ベクトルシーケンスに変換した後，特徴選択を組み合わせながら CNN と RNN を用いてマルウェアを分類した [16]．Liu らはマルウェアに呼び出された API シーケンスを抽出し，各単語の意味をベクトル表現化する word2vec を使って API シーケンスをベクトル化した後，BLSTM 精度などを使ってマルウェアを分類した [17]．

3) マルウェアの検知と分類に関する研究

Jung らはランサムウェアの検知だけでなく分類も可能なシステムを提案した [18]．Stiborek らはマルウェアがアクセスするシステム資源，たとえば，フォルダ，ネットワークトラフィック，mutex 名，レジストリ名などを特徴量としたマルウェアの分類手法を提案している [19]．

2.2 ハイブリット解析

Islam らの研究では静的な特徴と動的な特徴を統合してから，いくつかのアルゴリズムでマルウェア系列を分類した [20]．Santos らは静的と動的の特徴量を利用して未知のマルウェアを検知する OPEM と呼ばれる手法を提案した [21]．Anderson らも動的の特徴と静的の特徴を利用して，SVM でマルウェアを分類した [22]．

3. 提案方式

既存手法はマルウェアを検出または分類できるが，それらの特徴量の抽出が複雑で難しいにもかかわらず，マル

ウェアの動作の特徴も十分に表現することができていない．本章で，その課題を解決する提案方式について説明する．

3.1 提案方式の概要

提案方式は，マルウェアの動作パターンと特徴ある動作とを特徴量とする機械学習による分類手法である．本論文は，動作パターンとして API グループ間の相関性，特徴ある動作としてフォルダ操作頻度を使用し，提案方式の有効性を明らかにする．動作パターンとして API グループ間の相関性，特徴ある動作としてフォルダ操作頻度を使用した提案方式（以降，提案分類方式と呼ぶ）は，API グループ間の相関係数の算出，フォルダ操作頻度の抽出，特徴量変換とラベル付けの 3 ステップからなる．

(1) API グループ間の相関係数の算出

提案分類方式は，動作パターンとして API グループ間の相関性を使用する．動作パターンとして API 間の関係ではなく，API グループ間の関係に着目する理由としては，マルウェア系列ごとの動作パターンや動作間の関係を明らかにするためには，抽象度を上げた API グループ化が有効であるという考え方に基づく．API グループとして，マルウェアのファイル操作に関連する FileAPI グループ，ファイル暗号化に関連する CryptoAPI グループ，レジストリ操作に関連する RegistryAPI グループ，ネットワーク通信に関連する SocketAPI および，スレッドまたはファイル実行に関連する ProcessAPI の 5 つのグループを使用する（**表 1**）．そして，API グループ間の相関係数については，API グループごとの操作頻度を抽出した後，式 (1) [23] から算出する．i, j は API グループ i と j であり，R_{ij} は API グループ i と j 間の相関係数であり，C は API グループの共分散行列である．

$$R_{ij} = C_{ij} / \sqrt{C_{ii} \times C_{jj}} \tag{1}$$

(2) フォルダ操作頻度の抽出

提案分類方式は，特徴ある動作としてフォルダ操作頻度を使用する．フォルダに着目する理由としては，感染した PC のシステム情報，利用者情報，利用者が使用するアプリケーション情報など，マルウェアの用途が，参照するフォルダによって現れてくるであろうという考え方に基づく．

フォルダ操作については，実際にフォルダ配下にあるファイルを開く，ファイルをコピーする，ファイルに書き出す，ファイルから読み出す，ファイルを削除する，という 5 つの操作を使用する．フォルダ操作頻度の抽出では，これら 5 つのファイル操作が行われたフォルダ名を抽出する．

(3) 特徴量変換

API グループ間の相関係数の計算とフォルダ操作頻度の抽出の処理を終了した後，相関係数と頻度を 1 つのリストにまとめ，特徴量とする．

表 1 API グループ
Table 1 API groups.

グループ名	API
FileAPI	FindNextFile, FindFirstFile, FindFirsFileEx, SetFilePointer, SetFilePointerEx, GetFileSize, GetFileSizeEx, SetFileAttributes, GetFileType, CopyFileEx, CopyFile, DeleteFile, EncryptFile, NtReadFIle, NtWriteFile, GetFileAttributes, GetFileAttributesEx
CryptAPI	CryptDerveKey, CryptDecodeObject, CryptGenKey, CryptImportPublicKeyInfo, CryptAcquireContext, CryptAcquireContextW
RegistryAPI	RegCloseKey, RegCreateKeyExW, RegDeleteKeyW, RegQueryValueExW, RegSetValueExW, RegEnumKeyExA, RegOpenKeyExW, NtQueryValueKey, NtOpenKey
SocketAPI	socket, InternetOpen, shutdown, sendto, connect, bind, listen, accept, recv, send, InternetOpenUrl, InternetReadFile, InternetWriteFile
ProcessAPI	CreateThread, CreateRemoteThread, NtResumeThread, NtGetContextThread, NtSetContextThread, CreateProcessInternalW, NtOpenProcess, Process32NextW, Process32FirstW, NtTerminateProcess

4. 分類

本章で，FFRI が収集したマルウェアの動的解析ログである FFRI Dataset 2016 [24] を使用した提案分類方式の有効性検証（以降，分類実験）について述べる．

4.1 分類実験用データへの加工

本節で提案分類方式の有効性検証のために実施した FFRI Dataset 2016 から分類実験用データへの加工手順について述べる．

4.1.1 特徴量抽出手順

提案分類方式を検証するにあたり，FFRI Dataset 2016 から API グループの頻度の抽出，フォルダ操作頻度の抽出の具体的な手順について述べる．

(1) API グループの頻度の抽出

API 頻度の抽出アルゴリズムを**図 1** に示す．表 1 にある各 API の頻度を FFRI Dataset2016 から抽出した後，各 API グループ間の相関係数を計算し，相関係数リスト（$list_{corrcoef}$）に保存する（**図 2**）．

(2) フォルダ操作頻度

フォルダ操作頻度の抽出アルゴリズムを**図 3**，**図 4** に示す．まず FFRI Dataset2016 からフォルダに関する文字列を抽出した（図 3）後，抽出された文字列がファイルかフォルダかを判断する．フォルダである場合は，3.1 節 (2) に記述した 5 つの操作頻度を抽出する．ファイルである場合には，ファイル名を削除し，フォルダ名を抽出し操作頻度を積算する（図 4）．

4.1.2 分類のためのラベル付け手順

特徴量を抽出した FFRI Dataset 2016 の検体解析ログへのラベル付けについて述べる．ラベル付けは，特徴量を抽出した後，分類器を使って分類した結果の有効性を検証す

Algorithm 1: Correlation coefficient between API Groups

Input:
APIs' statistic information set: S;
API groups nested array: $list_{apig}$;
// $list_{apig}$ Structure:
// $[\{api_1 : fre_{api_1}, api_2 : fre_{api_2}, ...\},$
// $\{api_1 : fre_{api_1}, api_2 : fre_{api_2}, ...\},$
// $\{api_1 : fre_{api_1}, api_2 : fre_{api_2}, ...\},$
// $\{api_1 : fre_{api_1}, api_2 : fre_{api_2}, ...\},$
// $\{api_1 : fre_{api_1}, api_2 : fre_{api_2}, ...\}]$

Output:
$list_{corrcoef}$;

Create the frequency list of file operation-related: $list_{files}$;
Create the frequency list of crypto-related: $list_{crypto}$;
Create the frequency list of register-related: $list_{registry}$;
Create the frequency list of socket-related: $list_{socket}$;
Create the frequency list of process-related: $list_{process}$;
Create the correlation coefficient list : $list_{corrcoef}$;
// the frequencies of the APIs are extracted
// when they match
for $process_{id}$ *in* S **do**
  for api_{name} *in* $S[process_{id}]$ **do**
    for num *in* $list_{apig}$ **do**
      $list_{apig}[num][api_{name}] \leftarrow$ $list_{apig}[num][api_{name}] +$ $list_{apig}[process_{id}][api_{name}]$;

// Extract the frequency of the api and
// save it in their respective lists
// $list_{apig}$ Structure:
// $[\{api_1 : fre_{api_1}, api_2 : fre_{api_2}, ...\},$
// $\{api_1 : fre_{api_1}, api_2 : fre_{api_2}, ...\},$
// $\{api_1 : fre_{api_1}, api_2 : fre_{api_2}, ...\},$
// $\{api_1 : fre_{api_1}, api_2 : fre_{api_2}, ...\},$
// $\{api_1 : fre_{api_1}, api_2 : fre_{api_2}, ...\}]$
for num *in* $list_{apig}$ **do**
  for $fileAPI$ *in* $list_{apig}[num]$ **do**
    $list_{files} \leftarrow$list$_{apig}[num][fileAPI]$;
// Using numpy to calculate the
// Pearson Correlation Coefficient
// between API groups
$fc_{corr} \leftarrow numpy.correcoef(list_{files}, list_{crypto})$;
$fr_{corr} \leftarrow numpy.correcoef(list_{files}, list_{registry})$;
$fs_{corr} \leftarrow numpy.correcoef(list_{files}, list_{socket})$;
$fp_{corr} \leftarrow numpy.correcoef(list_{files}, list_{process})$;
$cr_{corr} \leftarrow numpy.correcoef(list_{crypto}, list_{registry})$;
$cs_{corr} \leftarrow numpy.correcoef(list_{crypto}, list_{socket})$;
$cp_{corr} \leftarrow numpy.correcoef(list_{crypto}, list_{process})$;
$rs_{corr} \leftarrow numpy.correcoef(list_{registry}, list_{socket})$;
$rp_{corr} \leftarrow numpy.correcoef(list_{registry}, list_{process})$;
$sp_{corr} \leftarrow numpy.correcoef(list_{socket}, list_{process})$;
$list_{corrcoef} \leftarrow fc_{corr}, fr_{corr}, fs_{corr}, fp_{corr}, cr_{corr},$ cs_{corr},cp_{corr}, rs_{corr}, rp_{corr}, sp_{corr};
return list$_{corrcoef}$

図 1 API グループ相関性抽出アルゴリズム：頻度の抽出
Fig. 1 API groups' correlation extraction algorithm: Frequency extraction.

図 2 API グループ相関性抽出アルゴリズム：相関係数の計算
Fig. 2 API groups' correlation extraction algorithm: Calculation of correlation coefficients.

るためのものである．ベンダが付与した検体名を使用し，ベンダが付与した系列名に合致するよう分類が可能かを評価する．なお，分類実験でラベルを抽出できない検体解析ログを使用しないこととした．

Algorithm 2: Folder Frequency Extraction Algorithm

Input:
"summary" information from samples log's generic: GS;
// GS Structure:
// $\{file_{opened} : [filepath_1, filepath_2, ...],$
// $file_{copied} : [filepath_1, filepath_2, ...],$
// $file_{written} : [filepath_1, filepath_2, ...],$
// $file_{read} : [filepath_1, filepath_2, ...],$
// $file_{deleted} : [filepath_1, filepath_2, ...]\}$

Output:
$dict_{folder}$;

Create file path array: $list_{folder}$ = [];
Create the "summary" subscript list: $list_{sub}$ = $[file_{opened}, file_{copied}, file_{written}, file_{read}, file_{deleted}]$;
Create folder frequency dictionary: $dict_{folder} = \{\}$;
Create folder operation type list: $list_{fre} = [0, 0, 0, 0, 0]$;
for num *in* $list_{sub}$ **do**
  if $list_{sub}[num]$ *is in* GS **then**
    for num_{opened} *in* $GS[num]$ **do**
      if $GS[list_{sub}[sum]][num_{opened}]$ *is directory* **then**
        $list_{folder} \leftarrow GS[list_{sub}[num]][num_{opended}]$;
      else
        $list_{folder} \leftarrow GS[list_{sub}[num]][num_{opened}][0]$;

図 3 フォルダ操作頻度抽出アルゴリズム：フォルダリストの抽出
Fig. 3 Extraction algorithm for folder operation frequency: Extraction of folder list.

for num *in* $list_{folder}$ **do**
  $dict_{folder}[list_{folder}[num]] \leftarrow list_{fre}$;
for num *in* $list_{sub}$ **do**
  if $list_{sub}[num]$ *is in* GS **then**
    for num_{opened} *in* $GS[list_{sub}[num]$ **do**
      if $GS[list_{sub}[num]][num_{opened}]$ *is* $directory$ **then**
        if $GS[list_{sub}[num]][num_{opened}]$ *is in* $dict_{folder}$ **then**
          $dict_{folder}[GS[list_{sub}[num]][num_{opened}]] \leftarrow$ $dict_{folder}[GS[list_{sub}[num]][num_{opened}]]+$ 1
      else if $GS[list_{sub}[num]][num_{opened}][0]$ *in* $dict_{folder}$ **then**
        $dict_{folder}[GS[list_{sub}[num]][num_{opened}][0]]$ $[num] \leftarrow$ $dict_{folder}[GS[list_{sub}[num]][num_{opened}][0]]$ $[num]+1$
return $dict_{folder}$

図 4 フォルダ操作頻度抽出アルゴリズムフォルダ：操作頻度の抽出
Fig. 4 Extraction algorithm for folder operation frequency: Extraction of operation frequency.

ラベル付けにあたり，マルウェアのラベル付けツール avclass [25] を使用して，FFRI Dataset2016 が提供する各ウイルス対策ベンダが付与した系列名を ["virustotal"] エントリから抽出する．avclass で抽出した各系列のうち，頻度が一番高い系列名を検体のラベルとした．なお，FFRI

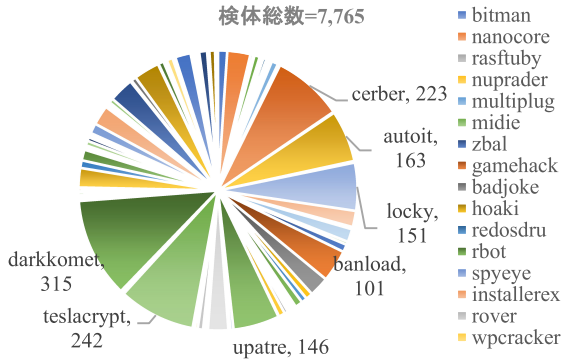


図 5 検体に付与したラベル数の分布

Fig. 5 Distribution of the number of labels assigned to the samples.

Dataset2016 の [“virustotal”] エントリには virustotal の各ウイルス対策ベンダのスキャン結果が記録されている．

4.1.3 特徴量数の最適化手順

分類器を使って分類した結果の有効性を検証するために実施した，除外すべき検体解析ログと特徴量数の最適化手順について述べる．

(1) 除外すべき検体解析ログ

すべての検体のラベルを抽出した後，各ラベルの検体数を積算する．そのうち，検体数が 1 個しかないラベルは機械学習に使用できないため，その検体解析ログを分類実験で使用しない．

(2) 特徴量の最適化

API グループ間の相関係数とフォルダ操作頻度を抽出したところ，特徴量の数は 9,520 個に達し，機械学習の学習時間の減少およびモデルの精度を向上させるために，SelectFromModel [26] を使用して特徴選択する．SelectFromModel は特徴の重要度で特徴を選択するため，今回の実験で特徴の選択用の閾値をデフォルト（すべての特徴の重要度の中央値 “median”）にして，重要度が中央値より大きく，分類に最適な 234 個の特徴量を抽出した．

4.2 分類実験の前提条件

- 図 5 に検体に付与したラベル数の分布を示す．図中の 1 つのブロックが 1 つのマルウェア系列に該当する．この図からも分かるとおり，マルウェア系列ごとに，ブロックの大きさが異なっている．すなわち，マルウェア系列ごとに検体として登録されている数に偏りがあり，FFRI Dataset 2016 は不均衡データセットであるといえる．不均衡データセットは機械学習の精度に影響することから [27]，RandomOversample を使用してオーバサンプリングの有無による分類を試みる [28]．
- テストデータサイズは，全体の 30%とする [29]．
- 有効性の評価にあたっては，評価指標として，Precision-Recall 曲線と Matthews Correlation Coefficient（MCC）値 [30] を用いる．MCC 値とは偽陽性と真陽性を考慮し，クラスが非常に異なるサイズであっても使用できるバランスの取れた尺度であり [31]，式 (2) により算出する．

$$MCC = \frac{c \times s - \sum_k^K p_k \times t_k}{\sqrt{\left(s^2 - \sum_k^K p_k^2\right) \times \left(s^2 - \sum_k^K t_k^2\right)}} \quad (2)$$

この式において，K はクラスの総数である．

- $t_k = \sum_i^K C_{ik}$ はクラス k の実際に発生した回数，C は混同行列である．
- $p_k = \sum_i^K C_{ki}$ はクラス k の予測された回数である．
- $c = \sum_k^K C_{kk}$ は正確に予測された検体の数である．
- $s = \sum_i^K \sum_j^K C_{ij}$ は検体の総数である．

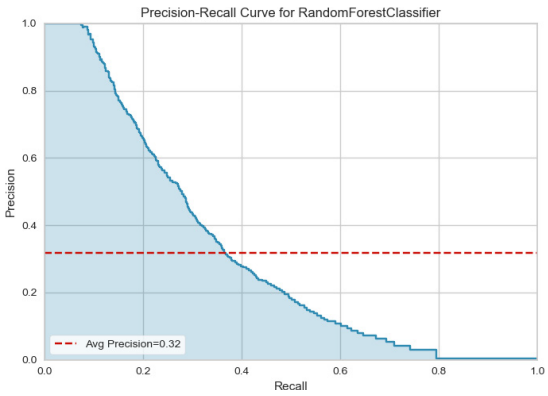


図 6 Precision-Recall 曲線（オーバサンプリングなし）

Fig. 6 Precision-Recall curve (without oversampling).

4.3 実験結果

本節では，前提条件に基づき，Precision-Recall 曲線と MCC 値を用いてオーバサンプリングの有無による分類結果を示す．

(1) オーバサンプリングなし

オーバサンプリングなしの分類では，Precision-Recall 曲線は図 6 のとおりであり，ベンダが付与した系列名に合致するよう分類できた正確度は 0.35 であり，精度は 0.32（赤色破線）で，MCC 値も 0.33 にとどまった．

(2) オーバサンプリングあり

不均衡データであることから，オーバサンプリングを実施した場合には，ベンダが付与した系列名に合致するよう分類できた正確度は 0.99 であり，精度は 100%（図 7 の赤色破線）となった．MCC 値は 0.988 であり，Precision-Recall 曲線（図 7）に示すとおりであった．

4.4 考察

既存手法との比較を行った後，オーバサンプリングなしの場合については，オーバサンプリングが分類結果に与える影響を，オーバサンプリングありの場合については，特徴量の寄与度について考察する．

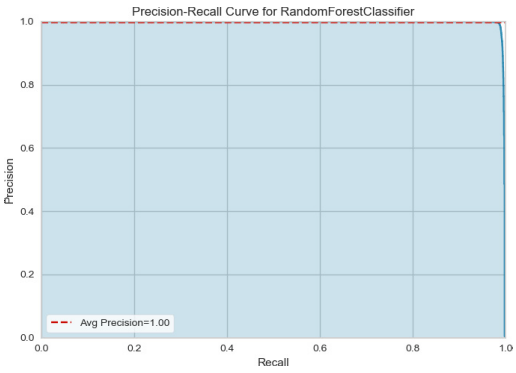


図 7 Precision-Recall 曲線（オーバサンプリングあり）
Fig. 7 Precision-Recall curve (with oversampling).

(1) 既存手法との比較

機械学習を用いたマルウェアの分類実験において，実験結果に影響を与える要素として，特徴量の選択と分類器の設定がある．

表 2 では，特徴量の選択と分類器の設定の視点から，提案分類方式の評価結果と既存手法の結果の比較を示す．既存研究では，静的な特徴量（関数の長さ，印刷可能な文字列など），または動的な特徴量（API 名，パラメータなど）を使用してマルウェアを検知または分類するとともに，実験精度に影響を与える要因を分析して改善方法も言及しているが，特徴量，マルウェアの動作の特徴と実験精度の関係を分析していない．一方，本研究では，動作パターンとして API グループ間の相関性，特徴ある動作としてフォルダ操作頻度に着目し，それらが実験精度にどのように影響するかを分析し，API グループ間の相関を考慮することが分類に効果があることを示している．また，Kakisim らの手法での正確度は 100%であり，提案分類方式よりも高いが，提案分類法は，マルウェアの動作間の関係を表現できる API グループ間の相関を明らかにしている点で優れていると考える．

また，提案分類方式の評価は，FFRI データセットを用いて行っているため，次の点から未知のマルウェアの分類には制限があると考えている．

- 攻撃者が意図的に大量の無駄な API を呼び出した場合，分類精度を低下させることができる．
- 未知のマルウェアの場合には，フォルダを特定できない．

(2) オーバサンプリングなしが分類結果に与える影響について

オーバサンプリングなしの分類精度が低い原因は，次の 2 つであると考えている．

a) データセットの不均衡

不均衡データが原因で各マルウェア系列の特徴を十分学習できない可能性がある．

b) API グループ間の相関係数の差が小さい

系列ごとの rp（RegistryAPI と ProcessAPI）グループ間の相関係数を**図 8** に示す．オーバサンプリングあり/なしのいずれも，各系列の rp グループ間の相関係数ほぼ 1 である．差が小さいために分類が難しくなってしまっているが，検体数を増やすと，改善する可能性があると考えている [32]．また，オーバサンプリングを使用して検体数を

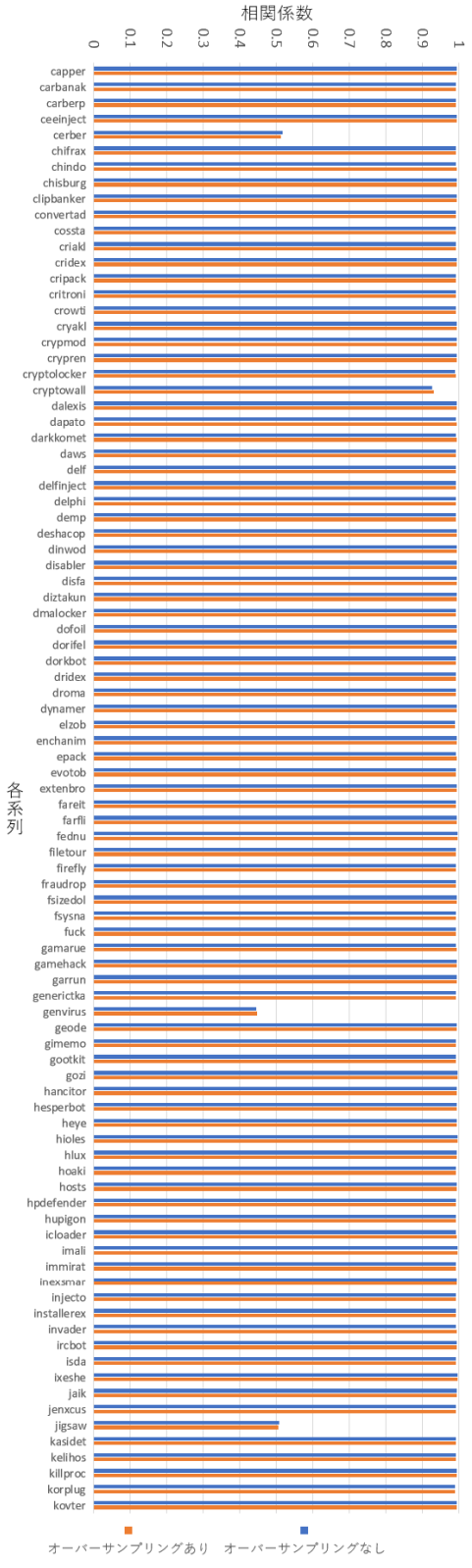


図 8 rp グループ間の相関係数の統計
Fig. 8 Statistics of correlation coefficients between rp groups.

表 2 既存手法との比較
Table 2 Comparison with related works.

手法	特徴量		検体数(データ分割比率)	分類器	正確度
Medhat et al.	API functions,cryptographic,etc.		2,568 (1,798:770)	検知手法そのものを提案	94.14%
Kakisim et al.	API call, usage system library		14,202 (10-分割交差検証)	HMM, SVM, J48, RF	100%
Kolosnjaji et al.	System calls		4,753 (3-分割交差検証)	DNN, HMM, SVM	98.4%
Liu and Wang	API sequences		13,518 (8:1:1)	GRU, BGRU, LSTM, BLSTM, SimpleRNN	97.8%
Stiborek et al.	Similarity between behaviors		250,527 (5-分割交差検証)	分類手法そのものを提案	94.4%
Islam et al.	API calls		2,939 (10-分割交差検証)	FLF,PSI,dynamic,integrated 手法	97.4%
Santos el al.	Opcodes sequences, system calls trace		13,189 (10-分割交差検証)	KNN, DT, SVM, Naïve Bayes, Bayesian network	96.6%
提案手法	API グループ間の相関係数とフォルダ操作頻度	Oversampling あり	148,135 (7:3)	Random Forest (n_estimators=268)	99%
		Oversampling なし	7,765(7:3)	Random Forest (n_estimators=268)	35%

注：データ分割比率は，(訓練データ：テストデータ)あるいは(訓練データ：テストデータ：検証データ)の比率を示す．

表 3 各系列の分類に寄与度が高いフォルダと API グループ（一部）
Table 3 Folders and API groups that contribute significantly to the classification of each series (partial).

系列	寄与度が高いフォルダと操作タイプ	API グループ
sysn	C:¥tmp2funuz¥modules¥packages を開いた回数	fr(Files-Registry) fp(Files-Process)
zegostzlader	C:\ProgramData\Microsoft\Windows\AppRepository\Packages\Microsoft.Windows.Cortana_1.6.1.52_neutral_neutral_cw5n1h2txyewy を開いた回数 C:\Users\rihoko\AppData\Roaming\Yfpey を開いた回数	fc(Files-Crypto)
fraudrop	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\SetupCache\v4.6.01055\1031 を開いた回数	fs(Files-Socket)
crypmod	C:\Users\rihoko\Documents を開いた回数	fc(Files-Crypto) fr(Files-Registry) fs(Files-Socket) fp(Files-Process)
dynamer	c:\$WINDOWS.~BT\Sources\Panther をコピーした回数	fc(Files-Crypto)

増やした場合，rp グループ間の相関係数の平均値は，オーバサンプリングなしの場合には 0.978307 であったものがオーバサンプリングありの場合には 0.978344 となり，相関係数が改善する可能性のあることを示している．

(3) オーバサンプリングありにおける特徴量の寄与度について

オーバサンプリングありにおける特徴量の寄与度については，4.1.3 項で抽出した 234 個の特徴量を対象とした分析について述べる．特徴量の寄与度を分析する目的としては，寄与度が高い特徴量から読み取れるマルウェアの動作上の特徴について検討することにある．

分析するにあたっては，分類器として OneVsRestClassifier と RandomForestClassifier を使用する．この 2 つの分類器を選択した理由としては，RandomForestClassifier で分類した場合，特徴量の寄与度を出力できる．しかし，RandomforestClassifier から出力する寄与度は分類実験全体に対する寄与度であり，各系列の分類に対する寄与度ではない．したがって，RandomforestClassifier を OneVsRestClassifier の推定器として使用すれば各系列の分類に対する特徴の寄与度を出力できる．

RandomforestClassifier は決定木に基づいているため [33]，ツリーの決定ノードとして使用される特徴の深さを使用して，ターゲット変数の予測可能性に関する特徴の寄与度を評価できる．また，ツリーの最上部で使用される特徴は，入力データの大部分の最終的な予測決定に役立つため，これらの特徴が貢献する予測割合は特徴の寄与度の推定値として使用できる [34]．

OneVsRestClassifier（OvR）はマルチクラスのストラテジであり，クラス（本論文の系列と同等）ごとに 1 つの分類器を適合させることにある．各分類器について，クラスはそのほかのすべてのクラスに対して適合される．計算効率（$n_{クラス}$ 個分類器のみが必要）に加えて，OvR の利点の 1 つはその解釈可能性である．各クラスは 1 つの分類器のみで表されるため，対応する分類器を調べることでクラ

スに関する情報を得られる [35]．分析結果は，各系列の分類に寄与度が高いフォルダ上位 5 位を寄与度の高い API グループとともに**表 3** に示す．たとえばマルウェア系列 crypmod の場合，C:¥Users¥rihoko¥Documents フォルダのファイルを開くことが特徴ある動作となる．このフォルダ操作からマルウェア crypmod の目的はユーザが作成したドキュメントフォルダ配下のファイルへの不正な操作であると推定でき，これはベンダ Microsoft の解析結果と一致している [36]．また，crypmod のファイル操作と関連する暗号化，レジストリとプロセス操作を virustotal のスキャン結果と一致している [37]．

5. まとめ

本論文では，既存研究をふまえて，マルウェアの動作パターンと動作上の特性とを組み合わせ，かつこれらを特徴量とする機械学習による分類方式を提案した．本論文では，動作パターンとして API グループ間の相関性，動作上の特性としてフォルダ操作頻度を使用した．分類実験で使用したデータセット FFRI Dataset 2016 は，データセットとして不均衡であること，特徴量が多すぎることから特徴選択をした後分類した結果，オーバサンプリングありの分類精度は 99%となり，オーバサンプリングなしの分類精度は 35%にとどまった．そして分類への寄与の高い特徴量については，API グループ間の相関係数，ファイルを開いた回数，コピーした回数の特徴量の寄与が高いことを明らかにした．

今後は，検体数を増やして不均衡データが分類に対する影響を抑えるとともに，様々な特徴量を使用して分類精度を向上させ，未知のマルウェア分類を対応できるように取り組んでいく．

参考文献

[1] Malcolm, J.: History of Malwa, *A Mem. Cent. India*, pp.22–57 (2011).

[2] BitDefender: Malware History, available from ⟨https://download.bitdefender.com/resources/files/Main/file/Malware_History.pdf⟩ (accessed 2020-02-11).

[3] Malwarebytes LABS: 2019 State of Malware, available from ⟨https://resources.malwarebytes.com/files/2019/01/Malwarebytes-Labs-2019-State-of-Malware-Report-2.pdf⟩ (accessed 2020-02-11).

[4] Fan, C.I., Hsiao, H.W., Chou, C.H. and Tseng, Y.F.: Malware detection systems based on API log data mining, *International Computer Software and Applications Conference* (2015).

[5] Alazab, M., Venkataraman, S. and Watters, P.: Towards understanding malware behaviour by the extraction of API calls, *2nd Cybercrime and Trustworthy Computing Workshop* (*CTC 2010*), pp.52–59 (2010).

[6] Seideman, J.D., Khan, B. and Vargas, A.C.: Identifying malware genera using the Jensen-Shannon distance between system call traces, *9th IEEE International Conference on Malicious and Unwanted Software, MALCON 2014* (2014).

[7] Cabau, G., Buhu, M. and Oprisa, C.: Malware classification using filesystem footprints, *2016 20th IEEE International Conference on Automation, Quality and Testing, Robotics, AQTR 2016* (2016).

[8] Zhou, J., Hirose, M., Kakizaki, Y. and Inomata, A.: Evaluation to Classify Ransomware Variants based on Correlations between APIs, *6th International Conference on Information Systems Security and Privacy* (*ICISSP 2020*), pp.465–472 (2020).

[9] Sihwail, R., Omar, K. and Ariffin, K.A.Z.: A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis, *Int. J. Adv. Sci. Eng. Inf. Technol.*, Vol.8, No.4-2, pp.1662–1671 (2018).

[10] Idika, N. and Mathur, A.P.: A Survey of Malware Detection Techniques, *Purdue Univ.*, Vol.48, p.2 (2007).

[11] Gandotra, E., Bansal, D. and Sofat, S.: Malware Analysis and Classification: A Survey, *J. Inf. Secur.*, Vol.5, No.2, pp.56–64 (2014).

[12] Hampton, N., Baig, Z. and Zeadally, S.: Ransomware behavioural analysis on windows platforms, *J. Inf. Secur. Appl.*, Vol.40, pp.44–51 (2018).

[13] Dahl, G.E., Stokes, J.W., Deng, L. and Yu, D.: Large-scale malware classification using random projections and neural networks, *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp.3422–3426 (2013).

[14] Medhat, M., Gaber, S. and Abdelbaki, N.: A New Static-Based Framework for Ransomware Detection, *2018 IEEE 16th Intl Conf Dependable, Auton. Secur. Comput. 16th Intl Conf Pervasive Intell. Comput. 4th Intl Conf Big Data Intell. Comput. Cyber Sci. Technol. Congr.*, pp.710–715 (2018).

[15] Kakisim, A.G., Nar, M., Carkaci, N. and Sogukpinar, I.: Analysis and evaluation of dynamic feature-based malware detection methods, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), LNCS, Vol.11359, pp.247–258 (2019).

[16] Kolosnjaji, B., Zarras, A., Webster, G. and Eckert, C.: Deep learning for classification of malware system call sequences, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), LNAI, Vol.9992, pp.137–149 (2016).

[17] Liu, Y. and Wang, Y.: A robust malware detection system using deep learning on API calls, *Proc. 2019 IEEE 3rd Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2019*, No.Itnec, pp.1456–1460 (2019).

[18] Jung, S. and Won, Y.: Ransomware detection method based on context-aware entropy analysis, *Soft Comput.*, Vol.22, No.20, pp.6731–6740 (2018).

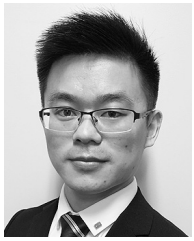
[19] Stiborek, J., Pevný, T. and Rehák, M.: Multiple instance learning for malware classification, *Expert Syst. Appl.*, Vol.93, pp.346–357 (2018).

[20] Islam, R., Tian, R., Batten, L.M. and Versteeg, S.: Classification of malware based on integrated static and dynamic features, *J. Netw. Comput. Appl.*, Vol.36, No.2, pp.646–656 (2013).

[21] Santos, I., Devesa, J., Brezo, F., Nieves, J. and Bringas, P.G.: OPEM: A static-dynamic approach for machine-learning-based malware detection, *Advances in Intelligent Systems and Computing* (2013).

[22] Anderson, B., Storlie, C. and Lane, T.: Improving malware classification: Bridging the static/dynamic gap,

Proc. ACM Conf. Comput. Commun. Secur., pp.3–14 (2012).
[23] Virtanen P. et al.: SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python, pp.1–22 (2019).
[24] 株式会社 FFRI：FFRI Dataset 2016 のご紹介，入手先 ⟨http://www.iwsec.org/mws/2016/20160530-ffri-dataset-2016.pdf⟩（参照 2020-02-11）.
[25] Sebastián, M., Rivera, R., Kotzias, P. and Caballero, J.: Avclass: A tool for massive malware labeling, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (2016).
[26] Pedregosa F. et al.: Scikit-learn: Machine learning in Python, *J. Mach. Learn. Res.* (2011).
[27] Luque, A., Carrasco, A., Martín, A. and de las Heras, A.: The impact of class imbalance in classification performance metrics based on the binary confusion matrix, *Pattern Recognit* (2019).
[28] Lemaître, G., Nogueira, F. and Aridas, C.K.: Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning, *J. Mach. Learn. Res.* (2017).
[29] Shahin, M., Maier, H. and Jaksa, M.: Data division for developing neural networks applied to geotechnical engineering, *Journal of Computing in Civil Engineering*, Vol.18, No.2, pp.105–114 (2004).
[30] Matthews, B.W.: Comparison of the predicted and observed secondary structure of T4 phage lysozyme, BBA - Protein Struct (1975).
[31] Boughorbel, S., Jarray, F. and El-Anbari, M.: Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric, *PLoS One* (2017).
[32] Chu, C., Hsu, A., Chou, K., et al.: Does feature selection improve classification accuracy? Impact of sample size and feature selection on classification using anatomical magnetic resonance images, *Neuroimage*, Vol.60, No.1, pp.59–70 (2012).
[33] Breiman, L.: Random forests, *Machine Learning*, Vol.45, No.1, pp.5–32 (2001).
[34] Louppe, G.: Understanding random forests: From theory to practice, arXiv preprint arXiv:1407.7502 (2014).
[35] Bishop, C.M.: *Pattern Recognition and Machine Learning*, Springer (2006).
[36] Microsoft: Ransom: Win32/Crypmod.A!bit, available from ⟨https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=ransom:win32/crypmod.a!bit&ThreatID=2147734432⟩ (accessed 2020-02-11).
[37] Virustotal, available from ⟨https://www.virustotal.com/gui/file/17b923990673b325ab31bbe045ba49d4f39a6b9412c36cce136da2ce9fbb8995/detection⟩ (accessed 2020-06-16).



周 家興

2015年(中国)安慶師範学院コンピューター科学と技術専攻卒業．2020年東京電機大学大学院未来科学研究科情報メディア専攻修士課程修了．同年東京電機大学大学院先端科学技術研究科情報通信メディア工学専攻に入学，現在に至る．

廣瀬 幸

2012年東京電機大学工学第II部情報通信工学科卒業．2017年同大学大学院先端科学技術研究科博士課程修了．博士（工学）．同年より東京電機大学未来科学部情報メディア学科助教．アンテナ・電波伝搬，無線通信システム，物理層セキュリティ，ハードウェアセキュリティ等の研究に従事．

柿崎 淑郎

2003年東海大学工学部電子工学科卒業．2008年同大学大学院博士課程修了．博士（工学）．2008年東京理科大学工学部第一部電気工学科助教．2013年東京電機大学未来科学部情報メディア学科助教．2018年4月より同大学研究推進社会連携センター准教授．電子認証技術，情報システムとしての情報セキュリティ応用等の研究に従事．電子情報通信学会，IEEE各会員．

猪俣 敦夫 （正会員）

2008年奈良先端科学技術大学院大学情報科学研究科准教授，2016年東京電機大学未来科学部教授，2019年大阪大学サイバーメディアセンター教授，現在に至る．博士（情報科学）．一般社団法人JPCERTコーディネーションセンター理事，一般社団法人公衆無線LAN認証管理機構代表理事，奈良県警サイバーセキュリティ対策アドバイザー，情報セキュリティの研究開発および若手人材育成活動に従事．電子情報通信学会会員．

寺田 真敏 （正会員）

1986 年千葉大学大学院工学研究科写真工学専攻修士課程修了．同年（株）日立製作所入社，研究開発グループ，Hitachi Incident Response Team にてサイバーセキュリティの研究に従事．博士（工学）．現在，東京電機大学先端科学技術研究科教授．2004 年より JPCERT コーディネーションセンター専門委員，（独）情報処理推進機構セキュリティセンター研究員，2008 年より中央大学大学院客員講師，2015 年より日本シーサート協議会運営委員長，ICT-ISAC 運営委員を兼務．