親プロセスと子プロセスにおける APIコール列の類似性に着目したマルウェア分析

中村 英敏[1,a)]　松田 祥希[2]　高橋 健一[1,3]　川村 尚生[1,3]

概要：
インターネット利用の増加に伴い，マルウェアを用いたサイバー攻撃が増加している．このため，マルウェアのファミリー推定や，正規ソフトウェアとマルウェアの判別といった，マルウェア検知に向けた研究が行われている．これらの研究では API コール列やシステム負荷などの特徴量が用いられているが，プロセスの親子関係に着目した研究はあまり行われていない．そこで本研究では，親プロセスと子プロセスにおける API コール列の類似性に着目した分析を行った．API コール列の類似性を分析するために，自然言語処理手法を用いて類似度を測定した．その結果，正規ソフトウェアとマルウェアとでは，親プロセスと子プロセスの類似性に違いがあることがわかった．また，得られた類似度を用いて，SVM による正規ソフトウェアとマルウェアの判別を行った．

キーワード：マルウェア，動的解析，API，NLP，SVM

Malware analysis focusing on similarity of API call sequences in parent and child processes

Hidetoshi Nakamura[1,a)]　Yoshiki Matsuda[2]　Kenichi Takahashi[1,3]　Takao Kawamura[1,3]

Abstract: As the use of the Internet has been increasing, cyber attacks using malware become worldwide issues. Therefore, malware family estimation and difference between legitimate software and malware has been researched. These studies make use of API call sequences and system load etc, however the similarity of parent-child processes has not enough investigated. Therefore, we focused on the similarity of API call sequences between parent and child processes. To analyze the similarity of API call sequences, we measured similarity score by using natural language processing techniques. As the result, we found the similarity score between parent and child processes has differences in legitimate softwares and malwares. Further, we apply the similarity scores to SVMs for the inference of legitimate software and malware.

Keywords: Malware, Dynamic Analysis, API, NLP, SVM

1. はじめに

インターネット利用の増加に伴い，マルウェアを用いたサイバー攻撃が増加している．2022 年 2 月には，トヨタの仕入れ先の部品メーカーである「小島プレス工業」にサイバー攻撃が行われ，トヨタの国内全 14 工場 28 ラインがストップした [1]．攻撃には，PC 内にあるデータを暗号化し，データの復号化の代わりに身代金を要求するマル

[1] 鳥取大学大学院 持続性社会創生科学研究科
Graduate School of Sustainability Science, Tottori University
[2] 鳥取大学 工学部電気情報系学科
Department of Electrical Engineering and Computer Science, Faculty of Engineering, Tottori University
[3] クロス情報科学研究センター
Cross-informatics Research Center
[a)] M21J4039K@edu.tottori-u.ac.jp

ウェアであるランサムウェアが用いられた．同年 6 月には Emotet と呼ばれるマルウェアが再び流行した [2]．Emotet は 2021 年 1 月に欧州刑事警察機構と欧州司法機構の協力の元，C&C サーバを停止させることに成功した．しかし，2021 年 11 月に再び活動を再開し，セキュリティー機器による検出を回避する機能の強化や，Google Chrome 内のクレジットカード情報を盗む機能の追加が行われ，危険度が高まっている．

このようなサイバー攻撃を防ぐための研究は盛んに行われている．例えば，API コール列の特徴や，システム負荷などの情報を用いた正規ソフトウェア（以降，正規ウェアと呼ぶ）とマルウェアの判別 [3][4][5]，マルウェアから抽出した文字列，API コール列から生成したベクトルなどを用いたマルウェアファミリーの推定 [6][7][8] といった研究が行われている．しかし，これらの研究の多くはプロセスの親子関係に着目しておらず，親プロセスと子プロセスを一体として扱っている．そこで本稿では，親プロセスと子プロセスの類似性に着目し，正規ウェアとマルウェアの分析を行った．

2. プロセスの親子関係

プロセスの親子関係とは，どのプロセスがどのプロセスから作成されたのかといった情報である．作成した側のプロセスを親プロセス，作成された側のプロセスを子プロセスと呼ぶ．例えば，プロセス A がプロセス B を作成した場合，プロセス A が親プロセス，プロセス B が子プロセスとなる．

子プロセスを作成して処理をまかせることで，並列実行による処理の高速化や，バグの発生による影響を最小限に抑えることができる．例えば，Google Chrome ではクラッシュ時の被害の分散や，タブ切り替えを高速に行うために子プロセスが作成されている．このような負荷分散や処理の高速化のために作成された子プロセスの動作は親プロセスと似ていることが多い．

一方でマルウェアでは親プロセスと子プロセスの動作が異なる傾向になると考えられる．例として Emotet の動作例を図 1 に示す．Emotet では不正なマクロが添付されたワードファイルからコマンドプロンプトが起動される．このとき，親プロセスはワード，子プロセスはコマンドプロンプトとなる．次に，コマンドプロンプトから PowerShell が起動され，指定の URL から Emotet 本体をダウンロードし，起動する．その後，Emotet 本体は自身を複製し，削除する．作成された Emotet（コピー）は，C&C サーバとの通信や，情報窃盗などの活動を行う．このように，Emotet の親プロセスと子プロセスはワードとコマンドプロンプトや，コマンドプロンプトと PowerShell など，異なるプロセスとなっていることがわかる．また，Emotet 本体と Emotet（コピー）の動作も，複製と削除，通信と情報窃盗

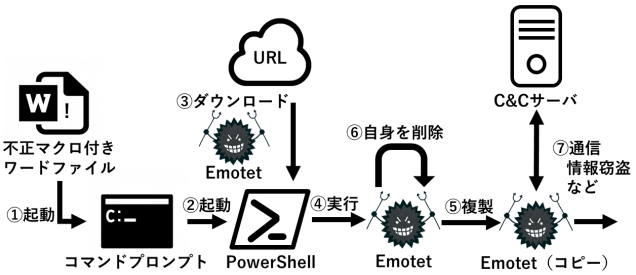


図 1: Emotet の動作
Fig. 1 How Emotet works.

など，異なる動作を行っていることがわかる．

3. 関連研究

飛山ら [3] は，RNN や CNN などの異なる Deep Neural Network を多段に用いることで，プロセスから得られた API コール列の特徴抽出と分類を行い，端末上で実行されているマルウェアプロセスの推定を行っている．小関ら [4] は，低対話型ハニーポットを設置し，そこに置かれたファイルを収集している．収集したファイルから，コマンドや通信先などの可読文字列を抽出し，それらの類似度などを用いて良性ファイルと悪性ファイルを分類している．佐藤ら [5] は，API 呼び出しの遷移・パターンに加えて，API 呼び出しの時間間隔と API 呼び出しに伴う API システム負荷に着目し，マルウェア検知に有効な特徴抽出を行っている．

大迫ら [6] は，IoT マルウェアのバイナリから文字列を抽出し，階層的クラスタリングによって分類木を作成し，マルウェアをファミリーごとに分類している．大久保ら [7] は，FFRI データセットから抽出した API コール列から BoW，Bit BoW，Word2Vec を用いて特徴ベクトルを生成し，マルウェアファミリーのクラスタリングを行っている．横瀬ら [8] は，静的解析と動的解析から得られた API コール列の結果を結び付け，静的解析のデータから関連する動的解析のデータを取りだすことで，マルウェアの亜種推定を行っている．

このようにマルウェア検知や亜種推定の研究では，API コール列や，システム負荷などの特徴を用いるものがよく見られる．しかし，これらの研究では，プロセスの親子関係に着目せず，親プロセスと子プロセスを分けずにマルウェア検知，亜種推定を行っている．

プロセスの親子関係を扱う研究として星澤ら [9] の研究がある．星澤らは，マルウェア動的解析の結果を視覚的に把握しやすくするために，プロセスの親子関係を用いたプロセスツリー，タイムライン，ヒートマップなどの可視化する手法を提案している．しかし，プロセスの親子関係の可視化を目的としており，マルウェア検知を目的としていない．そこで本研究では，親プロセスと子プロセス間の類似性に着目し，マルウェア検知に向けた分析を行う．

表 1: データ収集環境
Table 1 Data collection environment.

ホスト OS	Ubuntu 20.04.4 LTS
CPU	Intel Core i7-9700
ゲスト OS（実験用）	Windows10 Pro
ゲスト OS（INetSim 用）	Ubuntu 20.04.4 LTS
仮想化ソフトウェア	Oracle VM VirtualBox 6.1.32

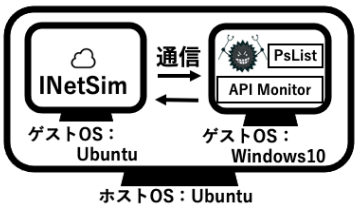


図 2: データ収集環境
Fig. 2 Data collection environment.

4. データ収集

4.1 検体

正規ウェアは窓の杜 2019 年ランキング [10]，Microsoft-Office，利用率が高いブラウザ（Google Chrome，Microsoft Edge，Firefox）から収集した．マルウェアは TekDefense[11]，theZoo[12] から，Windows 環境向けの検体を収集した．

4.2 データ収集方法

正規ウェアとマルウェアの親/子プロセス情報と API コール列を収集するための環境を構築した（表 1，図 2）．データ収集環境は，ホスト OS として Ubuntu 20.04.4 LTS，ゲスト OS として Windows10 Pro，Ubuntu 20.04.4 LTS を用いた．オンライン環境を模倣するためにゲスト OS の Ubuntu 20.04.4 LTS 上で INetSim[13] を起動し，Windows10 Pro 上で起動した検体と通信を行うように設定した．

Windows10 Pro 上で検体を実行し，API Monitor を用いて検体が呼び出した API コールを取得する．また，API Monitor ではプロセスが API を呼び出していることは取得できるが，プロセスが起動中かどうかは取得できない．そこで，プロセスの起動時間を取得するために PsList を実行し，1 秒ごとに起動しているプロセスの情報を取得した．検体の実行時間は [14][15] などを参考に 60 秒間とした．

API Monitor により取得できる情報の例を図 3 に示す．API Monitor により，対象プロセス，子プロセスのプロセス名，プロセス ID，API コール呼び出し時間，スレッド数，モジュール，API コール名，引数を取得できる．図 3 を見ると，親プロセスがプロセス ID8184 の Chrome.exe，子プロセスがプロセス ID1964 の Chrome.exe ということがわかる．また，右枠には選択されているプロセス ID8184 の Chrome.exe が呼び出した API が時系列順に表示されている．ここから，chrome_elf.dll モジュールの GetProcAddress という API が 46 番目に呼び出されていることがわかる．

PsList はプロセス名，CPU 使用率，スレッドの数，ハンドルの数，仮想メモリ，ワーキングセット，プライベート仮想メモリを取得できる．1 秒ごとにこれらの情報を取得することで，API Monitor で得られた親/子プロセスがどれくらいの間起動しているか測定することができる．[*1]

4.3 データ収集結果

データを収集した結果，子プロセスを起動していた正規ウェア 20 組，マルウェア 23 組のデータが得られた．正規ウェアとマルウェアの親/子プロセス毎の API の平均の呼び出し回数と，平均の種類数を表 2 に示す．

正規ウェアでは，親/子プロセスの平均呼び出し回数の差は 17356.2 で，マルウェアの親/子プロセスの平均呼び出し回数の差，5578.6 より少ないことがわかる．また，平均呼び出し種類数も，正規ウェアの親/子プロセスの差は 23.0 であり，マルウェアの親/子プロセスの差，136.4 より少ないことがわかる．これらのことから，マルウェアは正規ウェアより親/子プロセスの API の平均呼び出し回数や平均種類数の差が大きく，類似していない傾向にあることがわかる．

また，親/子プロセスのそれぞれで得られた API コール列の最初の 1000 個を集計し，横軸を API 名，縦軸を API の呼び出し数としたグラフを作成した（図 4）．正規ウェアのグラフ（図 4 左）を見ると，親/子プロセスで呼び出す API や，呼び出し回数に似たような傾向があることがわかる．一方，マルウェアのグラフ（図 4 右）を見ると，親/子プロセスで呼び出す API や，呼び出し回数に異なる傾向があることがわかる．

更に，正規ウェア，マルウェアの親/子プロセスの終了時間の分析を行った．横軸に終了した時間，縦軸に 5 秒ごとのプロセス終了数としたグラフを図 5 に示す．Leg が正規ウェア，Mal がマルウェアのプロセス数を示す．ここでは，60 秒を超えて起動していたプロセス（終了しなかったプロセス）は 60 秒に終了したと見なした．親プロセスの終了時刻のグラフを見ると，正規ウェアに比べてマルウェアが 5 秒以内に多く終了していることがわかる．一方，正規ウェアの多くは最後まで終了せずに起動し続けていることがわかる．次に，子プロセスの終了時刻を見ると，マルウェアは 60 秒以内にほとんどのプロセスが終了していることがわかる．一方，正規ウェアの多くは最後まで終了せずに起動し続けていることがわかる．

[*1] PsList で確認できなかったプロセスは 0 秒で終了したプロセスとして扱った．

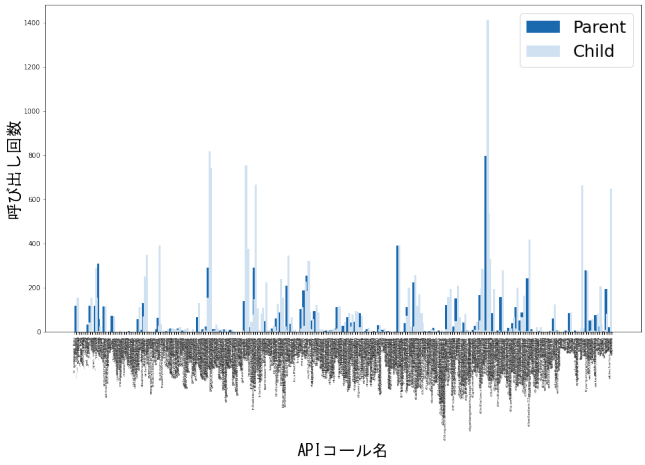
図 3: API Monitor で得られるログ

Fig. 3 API Monitor log.

表 2: 収集データ結果

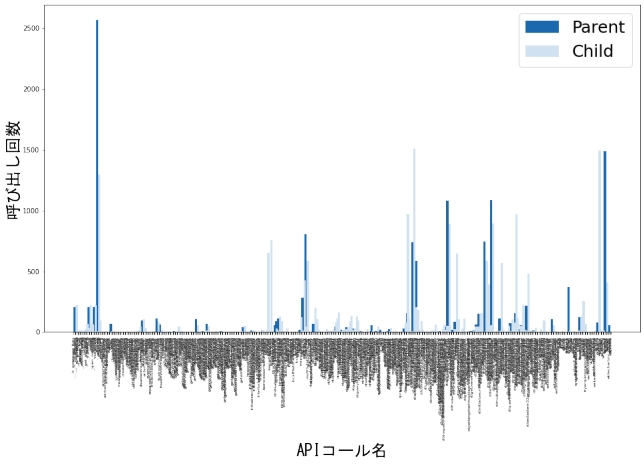
Table 2 Data collection results

	正規ウェア		マルウェア	
	親プロセス	子プロセス	親プロセス	子プロセス
API の平均呼び出し回数	119841.1	102484.9	55425.73	49847.13
API の平均呼び出し種類数	357.9	380.9	283.9	147.5



(a) 正規ウェア

Fig. 4 Legitimate software.

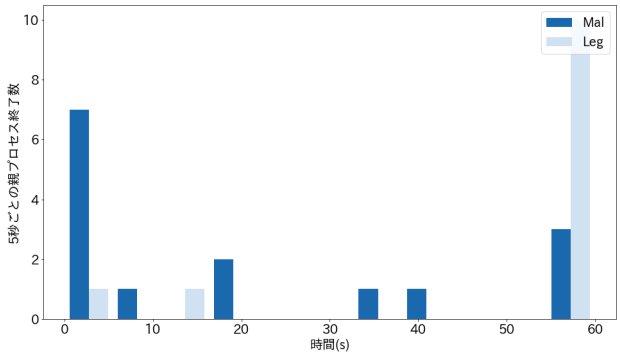


(b) マルウェア

Fig. 4 Malware.

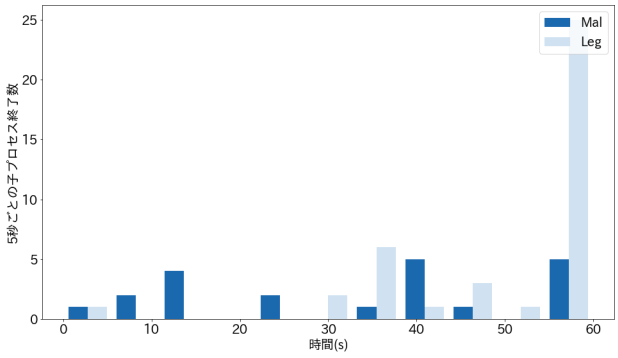
図 4: 親プロセスと子プロセスの API コール数の比較

Fig. 4 A number of API call in parent and child processes.



(a) 親プロセス終了時間

Fig. 5 Terminate time in Parent Process.



(b) 子プロセス終了時間

Fig. 5 Terminate time in Child Process.

図 5: 親プロセスと子プロセスの終了時間

Fig. 5 Terminate time in parent and child processes.

5. 自然言語処理手法による類似性の分析

正規ウェアとマルウェアの親/子プロセスで類似性に違いがあることがわかった．そこで，1 つの API コールを単語とみなし，自然言語処理手法を用いてそれぞれの親/子プロセスの類似度を測定する．

5.1 BoW

BoW（Bag-of-Words）とは文書を単語の出現回数のベクトルで表現したものである．シンプルな考え方で扱いやすいが，単語の順番が考慮されておらず，全ての単語を同じようにカウントすると必要以上に強調される単語が出てくるなどの欠点がある．

5.2 TF-IDF

TF-IDF とは文書内の単語の重要度を示す手法の一つであり，TF（Term Frequency）と IDF（Inverse Document Frequency）により計算できる．

TF は各文書においてその単語がどのくらい出現したのかを意味し，(1) 式で表すことができる．

$$TF = \frac{\text{文書 A における単語 X の出現回数}}{\text{文書 A における全単語の出現回数の和}} \quad (1)$$

IDF はその単語の希少性を表す指標であり，(2) 式で表すことができる．

$$IDF = \log \frac{\text{文書数}}{\text{単語 X を含む文書数}} \quad (2)$$

このとき，TF-IDF は (3) 式で表すことができる．

$$TF\text{-}IDF = TF * IDF \quad (3)$$

5.3 n-gram

n-gram とは任意の文字数 n で文章を分割する手法である．任意の文字数は連続する n 個の単語や文字のまとまりを表し，n が 1 の場合 uni-gram（ユニグラム），n が 2 の場合 bi-gram（バイグラム），n が 3 の場合 tri-gram（トライグラム）と呼ばれる．

5.4 類似度測定

類似度を測定するために，cos 類似度と一致率を用いる．cos 類似度は-1 から 1 の値をとり，1 に近ければ似ており，-1 に近ければ似ていないことを示す．ベクトル $\boldsymbol{x}$ とベクトル $\boldsymbol{y}$ の cos 類似度は (4) 式で定義される．

$$\cos(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{|\boldsymbol{x}||\boldsymbol{y}|} \quad (4)$$

一致率は 1 に近ければ似ており，0 に近ければ似ていないことを示す．一致率は (5) 式で定義される．

$$\text{一致率} = \frac{\text{一致した数}}{\text{総当たり数}} \quad (5)$$

BoW，TF-IDF はベクトルで表されるため，cos 類似度を用いて類似度を測定する．n-gram はベクトルではなく，API コールの組になっているため，一致率を用いて類似度を測定する．

5.5 分析方法

それぞれの検体における親/子プロセスの類似度を測定するために，それぞれの API コール列を BoW，TF-IDF，n-gram を用いて処理し，類似度を計算した．それらの類似度を有限の標本点から全体の分布を推定する手法であるカーネル密度推定を用いて分析を行った．また，それぞれの特徴量の関係性を比較するための散布図を作成した．

5.6 分析結果

カーネル密度推定による分析結果と，散布図による分析結果を示した散布図行列を図 6 に示す．図 6 の左上から右下にかかる対角線上のグラフは，その特徴量のカーネル密度推定結果を示す．それ以外のグラフは横軸と縦軸にそれぞれの特徴量をとった散布図を示す．Label の Leg が正規ウェアの値，Mal がマルウェアの値を表す．

5.7 カーネル密度推定の分析

カーネル密度推定（図 6 左上から右下にかかる対角線上のグラフ）の結果を見ると，BoW，TF-IDF では同じような形となっていることがわかる．これは，BoW と TF-IDF の両方が API コール列の時系列を考えず，その呼び出し頻度を重視するという指標であるためだと考えられる．また，マルウェアのデータの山が正規ウェアの山に比べて-1 に近いことがわかる．これはマルウェアの親/子プロセスの API コール列は正規ウェアに比べて似ていないということを表す．このことは uni-gram，bi-gram，tri-gram のカーネル密度推定の結果からも確認できる．マルウェアのデータの山が正規ウェアの山に比べて 0 に近く，マルウェアの親/子プロセスの API コール列は正規ウェアに比べて似ていないということがわかる．

5.8 散布図の分析

BoW と TF-IDF の類似度を比較した散布図は直線上となっている．これは，BoW と TF-IDF の両方が時系列を考えず，呼び出し頻度を重視する指標であるため，同じような結果になったためだと考えられる．また，bi-gram と tri-gram でもあまり結果が変わらないことがわかる．uni-gram と BoW の散布図を見ると，正規ウェアの値は右上に集まっているが，マルウェアの値は左側に集まってい

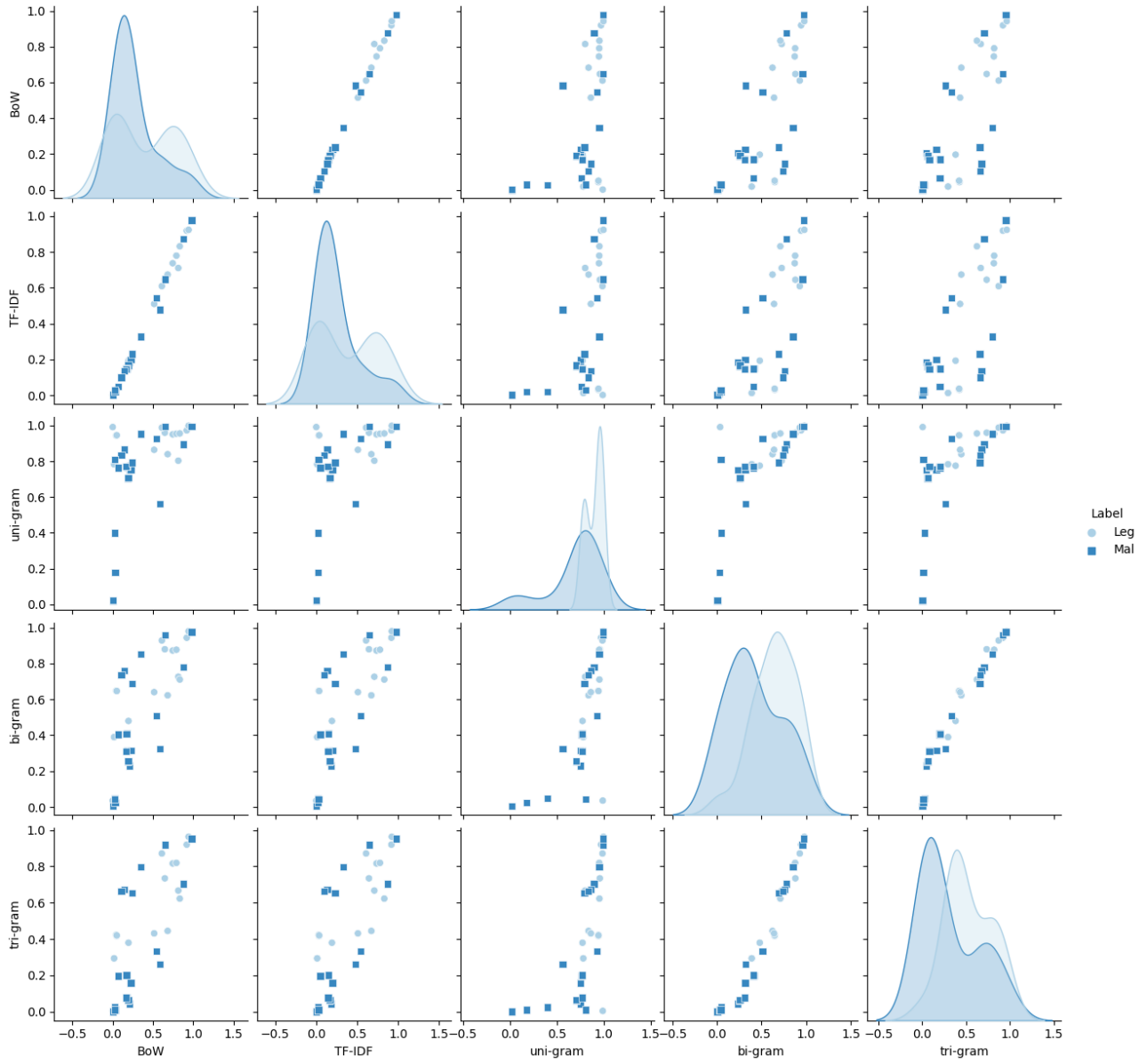


図 6: 散布図行列

Fig. 6 Scatter plot matrix.

ることがわかる．また，tri-gram と TF-IDF の散布図を見ると，正規ウェアの値が右上に，マルウェアの値が左下に集まっていることがわかる．それぞれの値が 1（右上）に近づくほど類似度が高く，値が 0（左下）に近づくほど類似度が低いと言えるため，正規ウェアの方がマルウェアより，親/子プロセスの API コール列の類似度が高いと言える．

6. SVM による判別

類似度を用いて，SVM による正規ウェアとマルウェアの判別ができるかを調べた．

6.1 SVM

SVM（Support Vector Machine）は，機械学習モデルの一種であり，少ない学習データでも安定的な結果を出せることで知られている．SVM はクラス間を分ける超平面を，それぞれのクラス間での最も近いデータ点（Support Vector）との距離ができるだけ大きくなるように決定することでクラスを分類する．

6.2 評価指標

評価指標として正解率（Accuracy），適合率（Precision），再現率（Recall），F 値（F1-score）を用いる．各指標の計算式を (6)〜(9) 式に示す．ここで，TP は True Positive，FP は False Positive，FN は False Negative，TN は True Negative を表す．TP はマルウェアのデータを正しくマルウェアだと予測したことを表す．

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (6)$$

$$Precision = \frac{TP}{TP + FP} \quad (7)$$

$$Recall = \frac{TP}{TP + FN} \quad (8)$$

$$F1 - score = \frac{2Recall * Precision}{Recall + Precision} \quad (9)$$

また，BoW，TF-IDF，n-gram の特徴量はスケールを合わせるために，平均が 0，標準偏差が 1 になる標準化を行った．特徴量内の各値を x^i，特徴量内の平均値を u，特徴量内の標準偏差を σ と置くと，標準化した値 x^i_{std} は (10) 式で表すことができる．

$$x^i_{std} = \frac{x^i - u}{\sigma} \quad (10)$$

6.3 判別方法

scikit-learn[16] を用いて SVM を構築し，BoW，TF-IDF，n-gram の類似度を用いて学習を行った．その後，random_state を 0 に固定し，5 分割交差検証によるモデルの性能の評価を行った．

6.4 判別結果

SVM による判別結果を表 3 に示す．

BoW，TF-IDF，BoW と TF-IDF の結果を見ると，正解率 0.59，F 値 0.60 と同じ結果になっており，スコアも一番低いことがわかる．これは，BoW，TF-IDF のカーネル密度推定の結果が正規ウェアとマルウェアとで大きな違いがなく，判別が困難だったためだと考えられる．BoW と TF-IDF の組み合わせが低かったのは，BoW と TF-IDF が API コール列の時系列を考えず，その呼び出し頻度を重視するという指標であるため，同じような結果になり，散布図上で違いが見られなかったためだと思われる．

正解率と適合率は tri-gram が一番高かった．しかし，再現率が 0.67 と一番低い結果となった．これは，適合率と再現率がトレードオフの関係にあるためだと考えられる．適合率を高くしようとすると FP を少なくしなければならないが，FP を少なくしようとすると FN が多くなってしまうため，再現率が低くなりやすい．このため，適合率は高かったが，再現率が低くなってしまったのだと考えられる．

再現率が一番高かったのは BoW，TF-IDF，uni-gram の組み合わせで 0.76 だった．この組み合わせは適合率が 0.83 であり，他の組み合わせと比べても低くない結果だった．また，この組み合わせの F 値は 0.78 と一番高かった．

7. まとめ

本稿では親/子プロセスの類似性に着目した分析を行った．API コール列，プロセスの終了時間を比較し，正規ウェアとマルウェアとでは親/子プロセス間の類似性に違いがあることがわかった．また，正規ウェアとマルウェアの親/子プロセスの API コール列の類似度を BoW，TF-IDF，n-gram を用いて測定した．得られた類似度を用いて SVM で正規ウェアとマルウェアの判別が可能かどうか確かめた．

参考文献

[1] Yahoo!Japan ニュース:”仕入先に”ランサムウェア”とみられるサイバー攻撃…トヨタが国内全工場の稼働停止 2 日からの再開も発表”，https://news.yahoo.co.jp/articles/9ea8a4aab7152a88ed7df41934b4c94212494a49

[2] NHK:”「エモテット」感染 再び増加傾向 新たな手口も 対策徹底を”, https://www3.nhk.or.jp/news/html/20220624/k10013685961000.html

[3] 飛山駿, 山口由紀子, 嶋田創, 秋山満昭, & 八木毅. (2016). Deep Neural Network 多段化によるプロセスの挙動に着目したマルウェア推定手法. コンピュータセキュリティシンポジウム 2016 論文集, 2016(2), 310-317.

[4] 小関純, 佐藤大造, & 阿曽村一郎. (2019). 類似性に基づくハニーポット収集データの分類. コンピュータセキュリティシンポジウム 2019 論文集, 2019, 1033-1038.

[5] 佐藤順子, 三須剛史, 花田真樹, 鈴木英男, & 布広永示. (2016). API 呼び出しとシステム負荷を用いたマルウェアの特徴抽出に関する一検討. コンピュータセキュリティシンポジウム 2016 論文集, 2016(2), 305-309.

[6] 大迫勇太郎, 山内利宏, 吉岡克成, 藤橋卓也, 渡辺尚, & 猿渡俊介. (2021). IoT マルウェアの分類方法に関する検討. コンピュータセキュリティシンポジウム 2021 論文集, 697-704.

[7] 大久保潤. (2019). CEM アルゴリズムを用いたマルウェアのクラスタリング. コンピュータセキュリティシンポジウム 2019 論文集, 2019, 1052-1058.

[8] 横瀬謙信, & 大久保潤. (2019). word2visualvec を応用したマルウェア亜種推定の枠組みの提案. コンピュータセキュリティシンポジウム 2019 論文集, 2019, 1047-1051.

[9] 星澤裕二, & 神薗雅紀. (2014). マルウェア動的解析結果の可視化の一手法. 研究報告コンピュータセキュリティ (CSEC), 2014(17), 1-4.

[10] 窓の杜:”2019 年総ダウンロード数トップ 100、「IP Messenger」が 4 位に浮上”, https://forest.watch.impress.co.jp/docs/serial/countdown/1227061.html

[11] TEKDEFENSE:”DOWNLOADS”，http://www.tekdefense.com/downloads/

[12] ytisf:”theZoo”, https://github.com/ytisf/theZoo

[13] Thomas Hungenberg & Matthias Eckert:”INetSim: Internet Services Simulation Suite” https://www.inetsim.org/index.html

[14] 朝倉紗斗至, 中川恒, 押場博光, 吉浦裕, & 市野将嗣. (2020). 動的解析ログを用いた特徴量の予測によるマルウェアの早期機能推定に関する検討. コンピュータセキュリティシンポジウム 2020 論文集, 602-609.

[15] Monnappa K A．初めてのマルウェア解析：オライリー・ジャパン，2020，425p．9784873119298.

[16] scikitlearn:”scikit-learn”, https://scikit-learn.org/stable/

表 3: SVM による判別結果
Table 3 SVM inference result.

	正解率	適合率	再現率	F 値
BoW	0.59	0.67	0.68	0.60
TF-IDF	0.59	0.67	0.68	0.60
uni-gram	0.70	0.72	0.75	0.72
bi-gram	0.70	0.76	0.67	0.70
tri-gram	0.80	0.95	0.67	0.77
BoW, TF-IDF	0.59	0.67	0.68	0.60
BoW, uni-gram	0.75	0.83	0.71	0.75
BoW, tri-gram	0.75	0.79	0.68	0.71
TF-IDF, uni-gram	0.725	0.78	0.71	0.73
TF-IDF, tri-gram	0.78	0.83	0.68	0.73
uni-gram, tri-gram	0.72	0.91	0.67	0.75
BoW, TF-IDF, uni-gram	0.78	0.83	0.76	0.78
BoW, TF-IDF, tri-gram	0.70	0.76	0.71	0.72
BoW, uni-gram, tri-gram	0.77	0.88	0.71	0.76
TF-IDF, uni-gram, tri-gram	0.77	0.88	0.71	0.76
BoW, TF-IDF, uni-gram, tri-gram	0.75	0.84	0.67	0.73