Vol.2020-CSEC-90 No.8
Vol.2020-SPT-38 No.8
2020/7/20

対象スレッドの違いによるマルウェア検知精度の比較

梶原　友希[†]　鄭　俊俊[†]　毛利　公一[†]

概要：マルウェア検知において，機械学習を用いたマルウェアの挙動をベースとする手法が提案されている．しかし，機械学習によるマルウェアの検知精度は，学習に使用するデータセットの違いに大きく左右される．また，同一データセットに対して異なる特徴量が与えられたとき，検知精度が低下する恐れがある．そこで，機械学習を用いたマルウェア検知の比較に有効なデータセットの作成を目指し，マルウェアを特徴つける情報を明らかにすることが重要である．本論文では，システムコールトレーサ Alkanet を利用して取得したログを使用し，対象とするスレッドを 3 種類に分け，スレッド内で出現するシステムコールの遷移回数を特徴量としてマルウェア検知を行った．3 種類の学習データにおける検知結果の比較により，プライマリスレッドの情報はマルウェア検知に有効であることが示された．

キーワード：マルウェア検知，機械学習，システムコール，対象スレッド

A comparison of malware detection accuracy with different target threads

YUKI KAJIWARA[†]　JUNJUN ZHENG[†]　KOICHI MOURI[†]

Abstract: Machine learning (ML) has been widely adopted as a promising approach for malware detection, considering the dynamic behavior of malware. Nevertheless, the detection accuracy of ML algorithm is strongly depending on the training dataset used, and, unfortunately, another different feature concentrated may bring decrease in the detection accuracy. That means, creating an appropriate dataset for higher-accuracy detection of malware, and meanwhile revealing the key features of malware are essentially important. In this paper, based on the execution logs of malware samples on Alkanet, the targeted threads were classified into three types, and the transition times of system calls appearing in each type of thread were considered as features for the ML algorithm of malware detection. The comparison results among three types of training datasets shown that the information of primary threads is useful for higher-accuracy detection.

Keywords: Malware detection, machine learning, system call, target thread

1. はじめに

近年，既存のマルウェアに加え，未知のマルウェア（新種や亜種）が毎年数多く発見されており，マルウェアによる脅威の拡大が深刻なセキュリティ問題 [1][2] となっている．その数は，年々増加傾向にあり，マルウェアを正確に検知し，被害の予防や防止などの対策を早期に行うことが求められている．マルウェア検知において，機械学習（ML: Machine Learning）を用いたマルウェアの挙動をベースとする手法が提案されている [3][4]．機械学習は，コンピュータに様々なデータを学習させ，その学習経験に基づき，未知のデータに対する予測や推定をさせる技術である [5]. その学習方法は, 教師あり学習（Supervised Learning: 回帰と分類），教師なし学習（Unsupervised Learning: クラスタリング）と強化学習（Reinforcement Learning: バンディットアルゴリズムと Q 学習）の 3 つに大別される．教師あり学習は，学習データに正解ラベルを付けて学習する方法である．一方，教師なし学習は，学習データにラベルを付けないで学習する方法である．強化学習はシステムのある行動選択に基づき，それに対する報酬を与えることで学習を行う方法であり, 教師なし学習に分類される場合もある．

機械学習を用いたマルウェア検知では，マルウェアと悪性な挙動を示さない一般的な実行ファイル（以下，クリーンウェア）それぞれのデータセットを基に特徴量を生成し，学習データを作成する．その学習データを用いて，学習モデルを訓練させ，最適なモデルパラメータを決定することで，高い検知精度が得られる．そのため，機械学習を用いたマルウェア検知精度は，データセットの違いに大きく左右されるといえる．しかし，1 つの提案手法に対して複数のデータセットを使用して実験を行っているものは少ない．高い精度を得ることが可能とされる学習モデルであっても，他のデータセットを使用することで，精度が下がる可能性が考えられる．

一方，システムコール（System Call）は，OS（カーネル）の持つ特殊な機能にアクセスするときに使われる仕組みであり，アプリケーションから利用される際に呼び出すサービスという．近年，システムコールを用いたマルウェア検知の手法が提案されてきた [6][7]. Xiao ら [6] は，Android マルウェアを検知するため，システムコールシーケンスを定常マルコフ連鎖に落とし込み，逆伝播ニューラルネットワークを適用し，マルコフ連鎖内のシステムコールの遷移

† 立命館大学
Ritsumeikan University

確率を用いてマルウェアを検出する手法を提案した. 特に，あるシステムコールから別のシステムコールへの遷移確率が，マルウェアと良性アプリケーションとでは大きく異なるという仮定に基づき，状態遷移の異常を捉えることで効率的に検出を行った．Shiva Darshan ら [7] は，Cuckoo Sandbox のログを用いたシステムコールベースの Windows マルウェア検出手法を実現した．JSON 形式のログを MIST 形式に変換した上で，システムコールを示す値を基に *n*-grams の特徴量を生成し，6 種類の学習手法を用いて性能評価を行った．本稿では，マルウェアの動的解析によるシステムコールトレースログに着目する．

特徴量や機械学習手法を既存研究と比較する場合，同一のデータセットを利用する必要がある．オープンに提供されているデータセットが使用されている場合は，同じものを用意可能である．しかし，ウェブサイト等から個人で収集されたデータから成るデータセットが使用されている場合が多く，同等のものを用意することは困難である．異なるデータセットを使用することで，既存研究との厳密な比較ができず，自身の提案手法が有用であるかを正確に判断することが難しくなる．実際，梶原ら [8] は，システムコールベースの特徴量を用いた機械学習によるマルウェア検知を行ったが，既存研究における提案手法との比較検討を行った際，同一のデータセットを用意することが極めて難しく，比較対象以外のすべての条件を統一することが困難だった. そのため, 厳密な比較を行うことができなかった. そこで，機械学習を用いたマルウェア検知精度の比較に有効なデータセットの作成を目指し，マルウェアを特徴付ける情報を明らかにすることが重要となる．

本稿では, システムコールトレーサ Alkanet [9] を利用して取得したログを使用し，対象とするスレッドの扱い 3 パターンに分け，スレッド内で出現するシステムコールの遷移回数を特徴量として学習データの作成に着目する. また，作成した学習データを用いてマルウェア検知を行い，検知結果の比較を行う．以下，2 節でスレッドを対象としたシステムコールベースのマルウェア検知について，3 節で 3 種類の学習データの比較実験について述べる．4 節で議論について述べ，最後に 5 節でまとめる．

2. スレッドを対象としたシステムコールベースのマルウェア検知

本稿では，システムコールトレースログを使用し，対象とするスレッドの扱いを 3 パターンに分けて学習データを作成し，マルウェア検知を行う．本節では，マルウェア検知に使用する 3 種類の学習データおよび各学習データを使用したシステムコールベースのマルウェア検知手法について述べる．

2.1 Alkanet ログ

本稿では，システムコールを観測できる動的解析環境として，システムコールトレーサ Alkanet を利用して取得したログを使用して学習データを作成し，マルウェア検知に使用する．このとき，生成されたスレッド内で出現するシステムコールを発行順に並べたものを使用した．以下，システムコールを発行順に並べたものを「システムコール列」とする．

Alkanet は, 本研究室で開発されている動的解析環境であり，マルウェアの挙動を観測できる．仮想計算機モニタ（VMM: Virtual Machine Monitor）である BitVisor の拡張機能として動作し，ゲスト OS 上のプロセスが発行したシステムコールのログを記録する．具体的には，システムコールのエントリポイントがカーネル空間に存在するため，ユーザモードで動作するプロセスからはアクセスすることができず，エントリポイントを回避してシステムコールを呼び出すこともできないことを利用し，システムコールをフックすることでシステムコールトレースを実現している．そのため，マルウェアの挙動を確実に観測可能である．Alkanet には, マルウェアの耐解析機能によって解析環境を検知されにくいという特徴があり，動的解析妨害機能を持つマルウェアの挙動も観測可能であるため，マルウェアの解析に有効である. また, システムコールの発行に対して，sysenter 時のログと sysexit 時のログの 2 つのログを記録するが，本稿では sysenter 時のシステムコールのみを抽出するようフィルタリングを行い，システムコール列とする．

2.2 生成されるスレッド数と検体数

悪意あるスレットがシステムに影響を与えるにはシステムコールが必要であるが，どのスレッドが悪性コードによって生成されたのか，どのスレッドに着目すべきなのかといった問題があるため，システムコール列をスレッドレベルの観点から区別することが重要である．また，マルウェアおよびクリーンウェアにおいて，検体実行時に生成されるスレッド数に差がある．今回使用したデータセットにおいては, マルウェア検体のスレッドは合計 1301 個, クリーンウェア検体のスレッドは合計 441 個生成された．マルウェアおよびクリーンウェア検体において生成されたスレッド数と検体数をそれぞれ表 1，表 2 にまとめる．

表 1, 表 2 よりマルウェアおよびクリーンウェア検体共に，生成されたスレッド数が 1 または 2 である検体が多いことが分かる．しかし，1 検体あたりに生成されたスレッド数が 100 を超える検体は，今回使用したデータセットにおいて，マルウェアには存在するが，クリーンウェアには存在しなかった．このように，マルウェアとクリーンウェア間で差があることから，生成されるスレッド数の差に関する情報はマルウェア検知結果に影響を与える可能性があると考える．

2.3 3 種類の学習データ

マルウェアおよび悪性な挙動を示さないプログラム（以下, クリーンウェア）の 2 種類の検体を Alkanet で実行し，

表 1：マルウェア検体におけるスレッド数

スレッド数	検体数
1	24
2	13
3	8
4	5
5	4
6	8
7	4
8	11
9	2
10	6
11	2
12	1
14	1
15	1
22	2
23	1
25	1
28	1
35	1
36	2
38	1
42	1
43	2
54	1
69	1
108	1
258	1

取得したシステムコールトレースログを用いて学習データを作成した．その際，検体の実行時に生成されたスレッドの扱いを 3 パターンに分けた．

具体的には，各検体において取得したログから，以下に示す 3 パターンのログを抽出し，各ログを基に 3 種類の学習データを作成した．

【パターン１】: 各検体におけるプライマリスレッドで出現したシステムコールを抽出したログ

【パターン２】: 各検体において生成される各スレッドで出現したシステムコールを抽出して 1 つにつなげたログ

【パターン３】: プライマリスレッド以外のスレッドで出現したシステムコールを抽出して 1 つにつなげたログ

上記 3 つのログそれぞれにおいて対象としているスレッドは，上から順に，「プライマリスレッドのみ」，「検体実行時に生成されたスレッドすべて」，「検体実行時に生成されたスレッドのうち，プライマリスレッドを除いたスレッドすべて」である．

表 2：クリーンウェア検体におけるスレッド数

スレッド数	検体数
1	35
2	35
3	7
4	11
5	3
6	5
7	3
8	2
10	1
11	1
35	1
66	1
67	1

プライマリスレッドには，パッカーの処理などの特徴的な挙動が現れる場合が多い．しかし，プライマリスレッドの後続のスレッド群内にマルウェアらしい挙動が現れる場合もあるため，マルウェアを検知するにあたってプライマリスレッドのみに着目して判別することが適切であるとはいえない．そこで，Alkanet を用いて取得した各検体のログにおいて，生成されたスレッドを上記の 3 パターンに分け，スレッド内で出現したシステムコール列を抽出してマルウェア検知を行い，検知結果を比較する．Alkanet は，プロセス生成やスレッド生成などを網羅的に観測可能であり，また，実行した検体の挙動を確実に観測可能であるため，より確実に情報を取得可能であると考える．

2.4 特徴量生成および機械学習手法

本稿では，各検体の実行時に生成されたスレッド内で出現したシステムコールの遷移回数を特徴量として使用し，ランダムフォレスト（Random Forest） [10] によるマルウェア検知を行う．

マルウェアがシステムに影響を与えるためには，必ずシステムコールの発行が必要であることから，マルウェアの挙動がシステムコールの発行順序や発行回数に反映される可能性が高いと考えられるためである．また，ログ内におけるシステムコールの出現および遷移に関する情報は，既存研究でも特徴量として使用され，高い検知精度を出しているものが多い [6, 11]．さらに，先行研究 [8] として，出現回数，出現確率，遷移回数，遷移確率の 4 つを特徴量とした学習データをそれぞれ作成し，ランダムフォレスト，SVM（Support Vector Machine），ロジスティック回帰（Logistic Regression）の 3 つの機械学習アルゴリズムを用いて，合計 12 パターンの特徴量と機械学習アルゴリズムの組み合わせによる検知精度を算出し，比較する実験を行った．その結果，システムコールの遷移回数を特徴量とした

表 3：混合行列

実際＼予測	マルウェア	クリーンウェア
マルウェア	TP：真陽性 (True Positive)	FN：偽陰性 (False Negative)
クリーンウェア	FP：偽陽性 (False Positive)	TN：真陰性 (True Negative)

ランダムフォレストによるマルウェア検知精度が最も高かったことから，本稿ではこの組み合わせを採用した．ランダムフォレストは，教師あり学習アルゴリズムの 1 つである決定木を複数集めてアンサンブル学習である．異なった方向に過剰適合した決定木を複数作成し，それらの予測結果の平均をとることで，過学習の度合いを減らす．分類の場合，各決定木による予測結果が出た後，その結果の多数決によって最終予測結果を出力する．

対象とするシステムコールは，今回使用したマルウェアおよびクリーンウェアの検体におけるシステムコールログ内で出現した全システムコールである．全部で 176 種類のシステムコールが出現しており，特徴量であるシステムコールの全遷移パターンは 30976（=176×176）パターンが存在する．各遷移において出現回数をカウントし，特徴量としている．

2.5 評価指標

機械学習のモデルの性能を評価する方法の一つに，交差検証（Cross-validation）[12] がある．交差検証では，学習データの分割を何度も繰り返して行って複数のモデルを訓練し，最後は複数のモデルの平均値をとって最終的な性能とする．ここでは，10 分割交差検証を用いて 2 値（クラス）分類（Binary Classification）の機械学習を行い，マルウェア検知の結果を評価する．2 値分類のモデルを評価する指標として，混合行列（Confusion Matrix）が重要である．混合行列は，実際のクラスを縦軸，モデルが予測したクラスを横軸として表した行列である．表 3 は，未知のデータをマルウェアもしくはクリーンウェアの 2 クラスに分類する場合の混合行列を示している．具体的に，実際のマルウェアを正しくマルウェアと推測できている状態（クラス）を TP (True Positive) なクラスといい，マルウェアではないもの（クリーンウェア）を正しくクリーンウェアと推測できる状態を TN (True Negative) なクラスといい，実際のマルウェアを誤ってクリーンウェアと推測している状態を FN (False Negative) なクラスといい，クリーンウェアを誤ってマルウェアと推測している状態を FP (False Positive) という．

また，分類の精度指標として，混合行列を基に，正解率（Accuracy），適合率（Precision），再現率（Recall），F 値（F-measure）を算出して使用する．正解率は，マルウェア検知の正確数の割合である．適合率と再現率は，混合行列の結果をまとめる方法として最もよく使用される指標である．適合率の値が大きいほど，誤検知が少なく，再現率の値が大きいほど，検知漏れが少ないと判断できる．適合率と再現率は，トレードオフの関係にあり，これらのバランスを示す値として F 値が用いられる．F 値は，適合率と再現率の調和平均値であり，使用したデータセットの各データがどの程度正しく分類できているかを表す指標である．以上の 4 つの精度指標の各値を算出する式をそれぞれ以下に示す．

- Accuracy =（TP + TN）/（TP + TN + FP + FN）
- Precision = TP /（TP + FP）
- Recall = TP /（TP + FN）
- F-measure =（2×Precision×Recall）/（Precision + Recall）

3. 比較実験

本節では，使用データ，実験手順および 3 種類の学習データを用いたマルウェア検知の比較評価について述べる．

3.1 実験データと手順

Alkanet の実行環境は，ゲスト OS として，Windows XP servicePack3 を利用した．マルウェアおよびクリーンウェアの検体それぞれ 106 検体を使用して得られた合計 212 個のデータを用いてマルウェア検知を行った．具体的に，マルウェアは，Windows 実行ファイルを 106 検体，クリーンウェアは「窓の杜 [13]」からダウンロードした 106 検体を用いた．ダウンロードしたファイルは，フリーソフトであること，Windows 実行ファイルであること，「窓の杜」から直接ダウンロードできることの 3 つの条件を満たすものを対象とした．実験手順は，以下の通りである．

(1) **システムコールログの抽出**
(2) **システムコールの遷移回数を特徴量の算出および学習データの作成**
(3) **ランダムフォレストによるマルウェア検知**
(4) 検知結果の評価

3.2 検知結果

3 種類の学習データそれぞれを用いてマルウェア検知を行い，得られた検知結果をそれぞれ示す．

2.2 節で説明した 3 パターンのログを抽出し，各ログを基に 3 種類の学習データを作成した．以下，パターン 1 を基に作成した学習データを「プライマリスレッド」，パターン 2 を基に作成した学習データを「全スレッド」，パターン 3 を基に作成した学習データを「プライマリスレッド以外」と表記する．

3.2.1 「プライマリスレッド」

「プライマリスレッド」を用いたマルウェア検知結果を，混合行列として表 4 に示す．検知正解率は，約 95.3%である．このときの各精度指標の値を表 5 に示す．

3.2.2 「全スレッド」

「全スレッド」を用いたマルウェア検知結果を，混合行

表 4：混合行列（プライマリスレッド）

実際＼予測	マルウェア	クリーンウェア
マルウェア	102	4
クリーンウェア	6	100

表 5：精度指標（プライマリスレッド）

	適合率	再現率	F 値
マルウェア	0.94	0.96	0.95
クリーンウェア	0.96	0.94	0.95

表 6：混合行列（全スレッド）

実際＼予測	マルウェア	クリーンウェア
マルウェア	100	6
クリーンウェア	6	100

表 7：精度指標（全スレッド）

	適合率	再現率	F 値
マルウェア	0.94	0.94	0.94
クリーンウェア	0.94	0.94	0.94

表 8：混合行列（プライマリスレッド以外）

実際＼予測	マルウェア	クリーンウェア
マルウェア	92	14
クリーンウェア	8	98

表 9：精度指標（プライマリスレッド以外）

	適合率	再現率	F 値
マルウェア	0.92	0.87	0.89
クリーンウェア	0.88	0.92	0.90

列として表 6 に示す．検知正解率は，約 94.3%である．このときの各精度指標の値を表 7 に示す．

3.2.3 「プライマリスレッド以外」

「プライマリスレッド以外」を用いたマルウェア検知結果を, 混合行列として表 8 に示す. 検知正解率は, 約 89.6%である．このときの各精度指標の値を表 9 に示す．

3.3 評価

表 4 より，「プライマリスレッド」におけるマルウェア検知の場合，マルウェアの方がクリーンウェアに比べて正しく検知できた検体が多いという結果になった．表 5 より，マルウェアにおいては適合率よりも再現率の値の方が高く，クリーンウェアにおいては再現率よりも適合率の方が高いという結果になった．このことから，マルウェアであると誤検知した検体は，クリーンウェアであると誤検知した検体よりも多く存在するが，マルウェアの検体において検知漏れは少ないことが分かる．F 値は，マルウェアおよびクリーンウェア共に同じであり，今回使用したデータセットの各データはそれぞれ約 95%の割合で正しく分類できていることが分かる．

表 6 より，「全スレッド」におけるマルウェア検知の場合，「プライマリスレッド」の場合に比べると検知精度は低いが，マルウェアおよびクリーンウェア共に正しく検知できた検体数が同等であるという結果になった．表 7 より，適合率，再現率，F 値それぞれの値が，マルウェアおよびクリーンウェア共に同じ数値であった．このことから，今回使用したデータセットに関して，「全スレッド」を用いた場合，マルウェアまたはクリーンウェアを同程度の精度で検知可能であるといえる．

表 8 より，「プライマリスレッド以外」におけるマルウェア検知の場合，クリーンウェアの方がマルウェアに比べて正しく検知できた検体が多いという結果になった．表 9 より，マルウェアにおいては再現率よりも適合率の値の方が高く，クリーンウェアにおいては適合率よりも再現率の方が高いという結果になった．このことから，クリーンウェアであると誤検知した検体は，マルウェアであると誤検知した検体よりも多く存在するが，クリーンウェアの検体において検知漏れは少ないことが分かる．F 値は，マルウェアよりもクリーンウェの方の数値が高いことから，今回使用したデータセットにおいて，クリーンウェアであると正しく予測できた割合が高いことが分かる．

表 4，表 6，表 8 より，マルウェア検体をマルウェアであると正しく検知できた検体数が最も多いのは，「プライマリスレッド」の場合であることが分かる．一方，クリーンウェア検体をクリーンウェアであると正しく検知できた検体数が多いのは，「プライマリスレッド」の場合と「全スレッド」の場合であることが分かる．

「プライマリスレッド」の場合が最も誤検知数が少ないことから，プライマリスレッドの情報はマルウェア検知に有効であることが分かる．「全スレッド」の場合も，「プライマリスレッド」の場合と同様に誤検知数が少ないが，「プライマリスレッド以外」の場合では誤検知数が多いという結果になった．このことから，対象スレッドにプライマリスレッドが含まれていることで，マルウェア，クリーンウェア共に正しく検知可能な検体数が増えるのではないかと考えられる．また，「プライマリスレッド以外」の場合において誤検知数が多くなった理由として，プライマリスレッドが学習対象に含まれていないことで，パッカーの処理などといったマルウェア特有の挙動がマルウェア検知を行う

際に学習経験として含まれなかったため，そのような検体における誤検知数が増えたのではないかと考えられる．

各学習データを用いた場合において，それぞれ誤検知した検体が存在したことから，マルウェアまたはクリーンウェアの特徴量として使用したシステムコールの遷移は，クリーンウェアまたはマルウェアに似ているものがあると考えられる．表 9 より，「プライマリスレッド以外」の場合，クリーンウェアであると誤検知したマルウェア検体が多かったことから，マルウェア検体において，プライマリスレッド以外のスレッド内のシステムコールの遷移は，クリーンウェアにおける挙動と似ているものが多いと考える．

4. 議論

今回使用した特徴量の数が，30976 個であり，膨大な数になっていた．そのため，学習データの作成や検知結果を取得するまでに時間がかかった．そのため，特徴量の削減を行い，学習データの作成や検知結果取得までの時間の短縮を検討するべきである．特徴量の削減を行うことで，検知精度の向上も期待できる．時間の短縮を行うにあたり，各特徴量の重要度（Importance）を求め，マルウェアであるか否かを判別するにあたって重要でないと判断できるシステムコールの遷移に関しては除外するといったことを検討している．

表 6 より，「全スレッド」におけるマルウェア検知の結果，マルウェアとクリーンウェア共に誤検知した検体数が同等であった．ここでマルウェアをクリーンウェアであると誤検知している検体とクリーンウェアをマルウェアであると誤検知している検体が一致した場合，マルウェアにもクリーンウェアにも誤検知しやすいシステムコールの遷移が多く含まれていると考えられる．そのため，3 種類の学習データを用いたマルウェア検知結果それぞれにおいて誤検知した検体を特定し，誤検知しやすい検体がどのような検体であるかの調査を検討する．

今回，「プライマリスレッド」，「全スレッド」，「プライマリスレッド以外」の 3 種類の学習データを作成し，それぞれについてマルウェア検知を行った．「全スレッド」や「プライマリスレッド以外」においては，各スレッド内で出現したシステムコールを発行順に並べた上で，スレッドの生成順に 1 つにつなげ，学習データとして扱った．しかし，スレッドとスレッドの間の遷移も学習対象として良いのか疑問が残る．たとえば，ある検体を実行した際，スレッド 1→スレッド 2 の順にスレッドが生成された場合，スレッド 1 の最後に発行されたシステムコールからスレッド 2 の最初に発行されたシステムコールの遷移を他の遷移と同等に扱っても良いのかが分からない．そのため，スレッド間の遷移をどう扱うべきか検討する必要があると考える．

また，スレッドを生成された順に並べて 1 つにつなげるのではなく，スレッド単位で分割し，すべてのスレッドを個別で扱った場合，マルウェアの判別がどの程度正確に行えるのか検討する．今回のマルウェア検知では，プライマリスレッド以外のスレッドに関してはスレッドを生成された順に並べて 1 つにつなげた．しかし，マルウェア検体が生成したスレッドがすべてマルウェアの挙動を示しているとは限らない．スレッドを個別で見た場合，マルウェア検体が生成したスレッドであっても，クリーンウェアのような挙動を示す可能性が考えられる．その逆もまた然りである．そのため，スレッド単位で分割し，各検体の実行時に生成されたすべてのスレッドを個別で扱った場合のマルウェア検知を検討する．マルウェア検体で生成されたがクリーンウェアと予測される，またはその逆が起こることが多いスレッドやシステムコールの遷移を特定できれば，そのようなスレッドまたは発行されるシステムコールの遷移を学習データ作成時に除外でき，その後の学習コストを小さくすることが期待できると考える．

5. おわりに

本稿では，システムコールトレーサ Alkanet を利用して取得したログを使用し，対象とするスレッドの扱いを 3 パターンに分け，スレッド内で出現するシステムコールの遷移回数を特徴量として学習データを作成し，マルウェア検知を行い，検知結果の比較を行った．3 種類の学習データにおける検知結果の比較により，プライマリスレッドの情報はマルウェア検知に有効であることが示された．今後は，4 節で述べた事柄について検討を行う．

参考文献

[1] Grygorenko, O., Sozonnik, G., and Al-Dulaimi, A. M. K.. Security Problem on The Internet of Things Networks. Information and Telecommunication Sciences. 2019, vol. 2, pp. 34-39.
[2] Kothari, S., Santhanam, G. R., Awadhutkar, P., Holland, B., Mathews, J., and Tamrawi, A.. Catastrophic Cyber-physical Malware. Versatile Cybersecurity. Springer, Cham, 2018, pp. 201-255.
[3] Suciu, O, Coull, S. E., and Johns, J.. Exploring Adversarial Examples in Malware Detection. In 2019 IEEE Security and Privacy Workshops. IEEE, 2019, pp. 8-14.
[4] Jung, J., Kim, H., Cho, S., Han, S, and Suh, K.. Efficient Android Malware Detection Using API Rank and Machine Learning. Journal of Internetl Services and Information Security. 2019, vol. 9, no. 1, pp. 48-59.
[5] Alpaydin, E.. Introduction to Machine Learning. MIT Press, 2020.
[6] Xiao, X., Wang, Z., Li, Q., Xia, S., and Jiang, Y.. Back-propagation Neural Network on Markov Chains from System Call Sequences: A New Approach for Detecting Android Malware with System Call Sequences. IET Information Security. 2017, vol. 11, no. 1, pp. 8-15.
[7] Shiva Darshan S. L., Ajay Kumara, M. A., and Jaidhar, C. D.. Windows Malware Detection Based on Cuckoo Sandbox Generated Report Using Machine Learning Algorithm. In 2016 11th International Conference on Industrial and Information Systems (ICIIS). IEEE, 2016, pp. 534-539.
[8] 梶原友希，鄭俊俊，毛利公一．システムコールを用いたマルウェア検知における機械学習アルゴリズムの比較．第 18 回情

報科学技術フォーラム. 2019.
[9] 大月勇人, 瀧本栄二, 齋藤彰一, 毛利公一. マルウェア観測のための仮想計算機モニタを用いたシステムコールトレース手法. 情報処理学会論文誌. 2014, vol. 55, no. 9, pp. 2034-2046.
[10] Cutler, A., Cutler, D. R., and Stevens, J. R.. Random Forests. In Ensemble Machine Learning. Springer, Boston, MA, 2012, pp. 157-175.
[11] Naval, S., Laxmi, V., Rajarajan, M., Gaur, M. S., and Conti, M.. Employing Program Semantics for Malware Detection. IEEE Transactions on Information Forensics and Security. 2015, vol. 10, no. 12, pp. 2591-2604.
[12] Wong, T. T.. Performance Evaluation of Classification Algorithms by *k*-fold and leave-one-out cross validation. Pattern Recognition. 2015, vol. 48, pp. 2839-2846.
[13] 株式会社インプレス : 窓の杜, https://forest.watch.impress.co.jp/, (2019.12.1 最終アクセス).