

動的解析システムのネットワーク接続の有無による マルウェア検知精度の比較

梶原 友希¹ 鄭 俊俊¹ 毛利 公一¹

概要：機械学習を用いたマルウェアの挙動をベースとしたマルウェア検知の精度は、使用するデータセットに依存するため、検知に有効なデータセットを作成することが重要である。そのため、まずはマルウェアを特徴付ける情報を明らかにする必要がある。マルウェアの多くは、外部と通信を行うため、検体実行時のネットワーク接続の有無が検知精度に影響を与える可能性が考えられる。本論文では、ネットワーク切断環境および接続環境でシステムコールトレサ Alkanet を利用してログを取得し、マルウェアを特徴付ける情報としてスレッドに着目し、スレッド情報に基づいた特徴量を生成することでマルウェア検知を行った。その結果、ネットワーク接続により、検知精度が下がる傾向に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、スレッド、ネットワーク接続

A comparison of malware detection accuracy in cases of dynamic analysis system with/without network connection

Abstract: It is important to create a dataset that is effective for malware detection because the accuracy of malware detection based on the behavior of malware using machine learning depends on the dataset used. Therefore, first of all, it is necessary to clarify the information that characterizes malware. Most malware communicates with the outside, so there is a possibility that executing samples with/without network connection affect the accuracy of malware detection. In this paper, based on the execution logs of malware samples on Alkanet with/without network connection, we focused on threads as information that characterizes malware, and detected malware by generating features based on thread information. As a result, it was shown that the accuracy tends to decrease due to network connection.

Keywords: Malware detection, machine learning, system call, thread, network connection

1. はじめに

毎年新しいマルウェアが数多く発見されており、マルウェアによる脅威の拡大が問題となっている [1]。既存のマルウェアだけでなく、未知のマルウェアや亜種による被害の予防などの対策も早期に行う必要があり、マルウェアをシグネチャに頼らずに正確に検知することが求められる。

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検知手法が多く提案されている [2][3]。機械学習では、学習や評価に使用するデータセットの質や量が、最終的に得られる結果に影響を与えるため、重要な要素である。しかし、データセットにおいて、以下のよう

な問題がある。

- 提案手法に対する客観的な評価が困難
- 共通のデータセットの利用が困難

機械学習によるマルウェア検知精度は、使用するデータセットの内容に依存する。しかし、既存研究では、提案している 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研究は少なく、特定のデータセットの内容に特化した結果になっている。そのため、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、精度が変動する。評価に使用するデータセットが異なることで、自身の提案手法の有効性や汎用性などを既存研究と明確に比較評価することが難しくなる。特徴量や機械学習手法などを比較評価する

¹ 立命館大学
Ritsumeikan University

場合、共通のデータセットを利用する必要があるが、共通のデータを利用することは難しい。機械学習を利用したマルウェア検知の研究には、マルウェアおよび悪質な挙動を示さないもの（以下、クリーンウェア）両方のデータが必要となる。オープンに提供されているデータセットではマルウェアの情報のみを提供しているものが多く、クリーンウェアは個人で収集や解析を行って使用される場合が多い。そのため、オープンで提供されているデータ以外において、共通のデータを用意することが困難である。

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が期待される。そのためには、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。マルウェアを特徴付ける情報の検証にあたり、実際にそれらの情報を利用した特徴量を基に学習データを作成し、機械学習を用いてマルウェア検知を行う必要がある。本論文では、特に以下について検討する。

- 動的解析ログと静的解析ログのどちらが有効か
- マルウェアを特徴付ける挙動を取得するため、どの情報に着目すべきか
- 動的解析システムのネットワーク接続の有無は検知精度にどのような影響があるか

機械学習を用いたマルウェアの挙動をベースとする検知手法として、難読化などの影響を受けない動的解析によるログを用いた手法が有効とされており、具体的にはマルウェアが発行したシステムコールを観測する手法がある。マルウェアがユーザモードで動作している限り、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ため、システムコールトレースログはマルウェアの挙動の把握に有効である。また、検体実行時に実際に実行されるのはスレッドであることから、発行されたシステムコールをスレッドレベルで区別することで、マルウェアを特徴付ける挙動をより明確に把握できる。

一般に、マルウェアには、実行中に外部と通信を行うものが多く存在する。そのため、ログ取得に利用する動的解析システムにおいて、検体実行時のネットワーク接続の有無により、取得できるログの内容が異なる。特にネットワークに接続する場合、外部との通信に関する挙動なども含んだマルウェア本来の挙動に近いシステムコールトレースログを取得することで、マルウェアをより正確に検知できると考えられる。したがって、動的解析システムのネットワーク環境により、最終的に得られる検知精度に影響を与える可能性がある。

本論文では、ネット切断環境およびネット接続環境においてシステムコールトレサ Alkanet[4] を利用してログ取得を行い、マルウェアを特徴付ける情報としてスレッドに着目し、スレッド情報に基づいた特徴量を生成して学習データを作成することでシステムコールベースのマルウェア検知を行う。その結果に基づき、動的解析システムの

ネットワーク接続の有無によって検知精度に与える影響について述べる。

以下、本論文では、2章でスレッド情報に基づいたマルウェア検知手法について述べ、3章でネットワーク接続の有無における検知精度の比較について述べる。4章で議論について、5章で関連研究について述べ、最後に6章でまとめる。

2. スレッド情報に基づいたマルウェア検知

本章では、まずマルウェアを特徴付ける情報について検討する。次に、検討した情報に基づき、Alkanetのログを用いた学習データの作成およびマルウェア検知について述べる。

2.1 マルウェアを特徴付ける情報の検討

2.1.1 動的解析ログ

機械学習を用いたマルウェアの機能や挙動をベースとする検知手法には、静的解析や動的解析のログを使用するものがある。一般的なマルウェア解析では、まず短時間で解析可能な動的解析を行い、その解析結果に基づき、必要に応じて静的解析による詳細な解析を行う。しかし、静的解析の場合、パッキングや難読化などによる影響を受けやすく、短時間での解析では十分な情報を得られず、その機能を検知できない可能性がある。また、新種や亜種も含め、様々なマルウェアが次々に発見されていることから、迅速かつ正確な検知および対策が求められる。そのため、難読化などの影響を受けず、比較的短時間でマルウェアの挙動の把握が可能な動的解析によるログを用いた手法が有効とされる。

また、マルウェアの挙動をベースとして検知を行うためには、動的解析により、マルウェアの挙動に関する情報を網羅的に収集可能であることが望ましい。マルウェアには、アンチデバッグ機能や、他のプロセスへ悪意あるコードを挿入する機能により、自身の活動を隠蔽したり解析を困難にしたりするものが存在する。このようなマルウェアにおいて、静的解析ですべての挙動を把握することは難しく、また動的解析を行う上でも、上記の機能への対応が課題になっている。したがって、アンチデバッグ機能への耐性を持ち、かつ他のプロセスへ悪意あるコードが挿入された場合でも正常な場合と区別して解析可能な動的解析システムを使用し、ログを収集することが望ましい。

2.1.2 システムコール

本論文では、マルウェアの挙動の理解が容易な情報を高速で提供可能な動的解析ログとして、システムコールトレースログを使用する。

動的解析において、観測可能な挙動の最小単位は、機械語命令単位やメモリアクセス単位であり、マルウェアの挙動を詳細に解析可能である。しかし、粒度が細かいため、挙

動の理解に時間がかかる上、解析時のオーバーヘッドが大きくなる。一方、マルウェアが実行中に呼び出した Windows API を記録する API トレースや、発行されたシステムコールを記録するシステムコールトレースは、機械語命令単位やメモリアクセス単位での観測に比べると粒度が粗い。これらは、API やシステムコールが呼び出された時の処理のみをフックするため、オーバーヘッドが小さくなる。また、挙動の理解が容易な情報を短時間で取得可能である。その一方で、フック対象の API やシステムコール以外が利用される挙動は観測できない。したがって、高速にログを取得する必要がある場合、API トレースやシステムコールトレースが有効である。

高速に解析を行うという点では、最も粒度が粗く、抽象度の高い Windows API を観測する API トレースの方が有効である。しかし、API トレースにおいて記録される Windows API は、無数に存在しているが、エントリポイントが統一されていない。そのため、すべてをフックすることが困難である。また、フック対象より下位の API やシステムコールをマルウェアが直接呼び出した場合、フックを回避されてしまう。

一方、システムコールのエントリポイントは、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できない。また、エントリポイントを経由せずに呼び出すこともできないため、マルウェアが意図的にシステムコールのフックを回避することはできない。マルウェアがユーザモードで動作している限り、ファイル操作や通信などを行うことで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、システムコールをトレースすることによって、低オーバーヘッドでマルウェアの挙動を確実に観測可能である。したがって、ユーザモードで動作するマルウェアの解析ログを取得するにあたり、システムコールトレースはマルウェアの挙動の把握に有効である。

2.1.3 スレッド

一般的に、マルウェアはプロセスとして動作するため、マルウェア検知などの研究においてプロセスに関する情報は有用である。既存研究においても、プロセスに着目し、システムコールや API コールの情報を利用してマルウェア検知を行っている研究がある。しかし、その中でもスレッドに着目したものは少ない。Windows における最小の実行単位は「スレッド」であり、各プログラムはスレッド単位で動作する。他のプロセスへ悪意あるコードを挿入する機能を持つようなマルウェアを解析する場合に、マルウェアの挙動に関するログを確実に取得するためにも、通常プロセス内に作成された悪意あるスレッドの挙動と他スレッドの挙動を区別できることが望ましい。

したがって、検体実行時に発行されたシステムコールをスレッドレベルで区別することで、マルウェアを特徴付け

る挙動をより明確に取得できると考えられる。

2.2 Alkanet によるログ取得

本論文では、システムコールの発行者をスレッド単位で識別し、プロセス生成やスレッド生成などを網羅的に取得可能であることから、Alkanet を利用してシステムコールトレースログの取得を行う。Alkanet[4] は、本研究室で開発されている動的解析環境である。マルウェアを動作させてシステムコールトレースを行うマルウェア観測用端末と、ログの収集および解析を行うロギング用端末の 2 台の端末を用いてマルウェア解析を行い、ログを取得する。

Alkanet は、仮想計算機モニタ (VMM: Virtual Machine Monitor) である BitVisor[5] の拡張機能として動作し、高速に、ゲスト OS である Windows 上で動作する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モードのメモリ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ができないことを利用し、発行する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。また、その際、システムコールの発行者をスレッド単位で識別する。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、実行した検体の挙動を確実に観測可能であるため、挙動の情報をより正確に取得可能である。さらに、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取得可能である。したがって、Alkanet を利用して得られたログは、マルウェア検知に有効であるといえる。

Alkanet は、ゲスト OS で発行されるすべてのシステムコールをトレース可能であるが、マルウェアの挙動を効率的に観測するため、典型的なマルウェアの機能の挙動に対応するシステムコールをトレース対象としている。本論文では、トレース対象の挙動以外にもマルウェアを特徴付ける挙動が現れる可能性を考慮し、トレース対象をすべてのシステムコールとして利用する。また、Alkanet は、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、本論文では、sysenter 時のシステムコールのみを抽出するようにフィルタリングを行い、使用する。さらに、本論文では、多数の検体に対してログ取得を行う必要があるため、Alkanet 自動観測システム [6] を利用する。Alkanet 自動観測システムは、Alkanet を構成する 2 つの端末に加え、外部との通信を制限するファイアウォール端末から構成されており、検体の実行からロギングまでを自動で行う。

2.3 学習データの作成

本論文では、機械学習で学習させるためのデータとして、

各検体の実行中に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、マルウェアの挙動を表すものとして使用する。以下、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ものを「システムコール列」と呼称する。

Alkanet を用いて取得したマルウェアおよびクリーンウェア検体のシステムコールトレースログから学習データを作成する。マルウェアを特徴付ける情報としてスレッドについて検証した先行研究 [7] では、検体実行時に生成されたスレッドの扱いを以下の 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。

- パターン 1: プライマリスレッドにおけるシステムコール列
- パターン 2: 全スレッドで出現したシステムコールを 1 つに繋げたシステムコール列
- 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 1 つに繋げたシステムコール列

その結果、プライマリスレッドの情報がマルウェア検知に有効であることが分かった。しかし、ネット接続の有無において、同検体を使用する場合、どのスレッドにおけるデータが検知精度に影響を与えているか明らかになっていない。そのため、本論文では、ネットワーク切断環境および接続環境において、パターン 2 の全スレッドを対象とする場合のマルウェア検知を行い、その検知結果を比較する。

2.4 特徴量および機械学習手法の検討

本論文では、各検体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によるマルウェア検知を行う。

マルウェアがユーザモードで動作している限り、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。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。以上より、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、Support Vector Machine (SVM)、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。システムコールの遷移情報を用

いることで高い精度で検知可能であり、かつランダムフォレストで検知を行うことで安定して高い精度を得られたことから、本論文ではこの組み合わせを採用している。特徴量生成において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。ランダムフォレストは、教師あり学習の 1 つである決定木を複数用いた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、各決定木においてマルウェアか否かの予測結果が出た後、多数決によって最終的な予測結果を出力する。

2.5 評価方法

本論文では、層化 10 分割交差検証を利用して 2 値 (クラス) 分類の機械学習を行う。また、混合行列および混合行列を基に算出した精度指標を用いてマルウェア検知結果を評価する。

2.5.1 層化 k 分割交差検証

層化 k 分割交差検証は、クラス分類を行う機械学習においてモデルの評価に用いられる手法であり、交差検証手法の 1 つである。交差検証は、学習モデルの汎化性能を評価する手法であり、その 1 つに k 分割交差検証がある。k 分割交差検証は、データセットを k 分割し、 $(k - 1)$ 個を学習データ、1 個を検証データとして性能を計測することを k 回繰り返して各分割ごとの精度を算出する。最後に k 個の性能の平均値をとることで、最終的な性能とし、モデルがデータセットに対してどの程度汎化しているかを評価する。しかし、k 分割交差検証におけるデータ分割は、データセットの先頭から $\frac{1}{k}$ ずつ取られるため、データセット内のデータが各クラスごとにまとまっていた場合、学習に偏りが生じる可能性がある。そのため、クラス分類を行う機械学習において、層化 k 分割交差検証が有用である。層化 k 分割交差検証では、各データ分割内でのクラスの比率が全体における比率と同等になるように分割される。本論文では、 $k=10$ として層化 10 分割交差検証を用いる。

2.5.2 混合行列

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。混合行列の主対角成分の要素は、正しく分類されたデータの個数を表し、それ以外の要素は誤分類されたデータの個数を表す。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の各要素を以下に示す。また、各要素をまとめて表 1 に示す。

True Positive (TP)

実際にマルウェアであるデータが、マルウェアであると正しく分類されたデータ数

表 1 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	True Positive	False Negative
クリーンウェア	False Positive	True Negative

True Negative (TN)

実際にクリーンウェアであるデータが、クリーンウェアであると正しく分類されたデータ数

False Positive (FP)

実際にクリーンウェアであるデータが、マルウェアであると誤って分類されたデータ数

False Negative (FN)

実際にマルウェアであるデータが、クリーンウェアであると誤って分類されたデータ数

2.5.3 精度指標

分類の精度指標として、混合行列を基に、精度 (Accuracy)、適合率 (Precision)、再現率 (Recall)、F 値 (F-measure) を算出する。精度は、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。

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

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F-measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

3. ネットワーク接続の有無における検知精度の比較

一般に、マルウェアには、実行中に外部と通信を行うものが多く存在する。そのため、ログ取得時のネットワーク接続の有無によって取得されるログの内容が異なり、最終的に得られる検知精度に影響を与える可能性がある。特にネットワークに接続する場合、外部との通信に関する挙動なども含んだマルウェア本来の挙動に近いシステムコールトレースログを取得することで、マルウェアをより正確に検知できると考えられる。本章では、ネットワーク切断環境および接続環境それぞれで取得したログを用いたシステムコールベースのマルウェア検知精度の比較について述べる。

表 2 検知結果 (ネット切断環境)

実際 \ 予測	マルウェア	クリーンウェア
マルウェア	91	15
クリーンウェア	12	94

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、FFRI Dataset 2019[9] に記録されている Windows 実行ファイル 106 検体、クリーンウェアは「窓の杜 [10]」から収集した Windows 実行ファイル 106 検体である。クリーンウェアの実行ファイルの収集にあたり、以下の要件を満たすものを対象とした。

- 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

ネット切断環境下およびネット接続環境下において特徴量の個数を統一するため、対象システムコールを各環境下で検体実行時に発行されたシステムコールの種類の和集合とし、特徴量を生成する。ネット切断環境下で検体実行時に発行されたシステムコールは 136 種類、ネット接続環境下では 233 種類であり、全部で 236 種類のシステムコールが発行された。そのため、特徴量であるシステムコールの全遷移パターンは $55696 (=236 \times 236)$ パターンとなる。実験手順は、以下の通りである。

- (1) Alkanet を利用してシステムコールトレースログの取得
- (2) システムコール列の抽出
- (3) 特徴量 (システムコールの遷移回数) の算出および学習データの作成
- (4) ランダムフォレストによるマルウェア検知
- (5) 検知結果の評価

3.2 検知結果

全スレッドが対象である学習データを用いたマルウェア検知によって得られた検知結果をそれぞれ示す。学習データ数は、マルウェアとクリーンウェアそれぞれ 106 個である。以下、全スレッドを対象として作成した学習データを「全スレッド」と表記する。

3.2.1 ネット切断環境

「全スレッド」を用いたマルウェア検知結果を混合行列で示したものを表 2 に示す。この検知結果における Accuracy は、約 87.3% であり、その他の各精度指標の値は表 3 に示す結果になった。

3.2.2 ネット接続環境

「全スレッド」を用いたマルウェア検知結果を混合行列で

表 3 精度指標（ネット切断環境）

	適合率	再現率	F 値
マルウェア	0.88	0.86	0.87
クリーンウェア	0.86	0.89	0.87

表 4 検知結果（ネット接続環境）

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	14	92

表 5 精度指標（ネット接続環境）

	適合率	再現率	F 値
マルウェア	0.87	0.87	0.87
クリーンウェア	0.87	0.87	0.87

示したものを表 4 に示す。この検知結果における Accuracy は、約 86.8%であり、その他の各精度指標の値は表 5 に示す結果になった。

3.3 評価

3.3.1 ネット切断環境

ネット切断環境下で取得したログを用いた場合、表 2 より、クリーンウェア検体の方が、マルウェア検体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。また、表 3 より、マルウェアにおいては適合率の方が高く、クリーンウェアにおいては再現率の方が高いという結果になった。これらの結果から、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 87%の割合で正しく検知可能であるといえる。

3.3.2 ネット接続環境

ネット接続環境下で取得したログを用いた場合、表 4 より、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った。また、表 5 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。これらの結果から、今回使用したデータセットに関して、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。

3.3.3 各環境における誤検知

表 2 と表 4 より、各接続環境における誤検知数に大きな差はない。しかし、誤検知された検体をそれぞれ比較したところ、共通して誤検知した検体数にはマルウェアとクリーンウェアで差があった。ネット切断環境でもネット接続環境でも誤検知した検体数、および一方の環境でのみ誤検知した検体数をそれぞれまとめたものを表 6 に示す。表 6 より、マルウェアにおいては、ネット切断環境下において誤検知された 15 検体のうちの 13 検体はネット切断環境下でのみ誤検知し、ネット接続環境下では正しく検知され

表 6 ネット接続の有無における誤検知数

	切断 ∩ 接続	切断のみ	接続のみ
マルウェア	2	13	12
クリーンウェア	11	1	3

た。また、ネット接続環境下において誤検知された 14 検体のうちの 12 検体はネット接続環境下でのみ誤検知され、ネット切断環境下では正しく検知された。一方、クリーンウェアにおいては、ネット切断環境下とネット接続環境下の両方で誤検知された検体が多く、一方の環境下でのみ誤検知された検体数は少なかった。

3.3.4 考察

ネット切断環境とネット接続環境それぞれにおいて、検知精度はネット切断環境の方が高かったが、F 値は同じ数値であった。このことから、今回使用したデータセットに関して、ネット接続の有無に関わらず約 87%の精度で検知可能であるといえる。ただし、マルウェアとクリーンウェアそれぞれにおける精度は変動する。また、ネット接続することによって、正しく検知されたマルウェアは増加したが、クリーンウェアは減少した。このことから、今回使用したデータセットに関して、全スレッドを対象とした場合、ネット接続することでマルウェアであると予測される傾向が高くなるといえる。

各環境下において誤検知した検体を比較した結果、表 6 より、共通して誤検知された検体は、マルウェアは少なかったが、クリーンウェアは多かった。このことから、今回使用したクリーンウェアにおいて、ネット接続の有無に関わらず同検体が誤検知される傾向が高く、マルウェアの挙動と類似する部分があると考えられる。しかし、マルウェアにおいては、一方の環境下におけるマルウェア検知で誤検知された検体の多くは、もう一方の環境下では正しく検知される傾向が高かった。したがって、ネット接続することにより、マルウェアを検知するために重視された特徴が変化したと考えられる。

以上より、検体および特徴量を統一した上で、全スレッドを対象とした学習データを用いた場合、ネット接続することにより、マルウェアであると検知される傾向が高くなったことで、検知精度が下がったと考えられる。

4. 議論

ネットワーク接続の有無におけるマルウェア検知精度の比較を行った結果、ネット接続することで検知精度が下がる傾向にあることが分かった。検知精度が下がった原因として、以下のことが考えられる。

- Alkanet を利用してログ取得を行った際の実行環境が限定的である
- クリーンウェアを収集するにあたり、挙動に関する条件がない

表 7 システムコールの種類数の違い

環境	マルウェア / クリーンウェア	マルウェア	クリーンウェア
切断	136 種類	129 種類	104 種類
接続	233 種類	131 種類	215 種類

本論文におけるログ取得にあたり、検体実行時の条件を統一するため、実行開始後、キーボード操作やマウス操作といったユーザ操作を行わない観測環境でログ取得を行った。そのため、実行開始直後にユーザ操作を必要とする検体では、ユーザ操作待ちの状態では停滞したログを取得している可能性が高い。特にクリーンウェアは、ユーザとのやり取りが多く、実行開始後すぐにキーボードによる入力待ちやマウスによる操作待ちの状態になる検体もある。したがって、ユーザ操作が行われない限定的な環境でログ取得を行ったことにより、その特徴がログに現れている可能性が考えられる。また、このような限定的な挙動のシステムコールトレースログから学習データを作成したことにより、実行環境の特徴がそのままマルウェアとクリーンウェアを区別する特徴になっている可能性がある。

本論文では、クリーンウェアを収集するにあたり、3章で述べた条件を設定していた。しかし、検体実行後のユーザ操作の有無など、検体の挙動に関する条件はない。そのため、ユーザとのやり取りが必要な検体において、その検体の特徴付けるために必要十分なログを取得できていなかった可能性がある。今回のように限定的な観測環境下で取得したログを用いる場合、ユーザとのやり取りを必要としない検体を選んで収集する必要がある。また、先述より、一般的にクリーンウェアは、ユーザとのやり取りが多い。一方、マルウェアは、クリーンウェアに比べてユーザとのやり取りが少なく、ユーザから見えないところで実行するものが多い。このような違いがあるため、今回のように限定的な観測環境にしない場合、クリーンウェアにおいて、ユーザとのやり取りを必要とする検体に対し、ユーザ操作を十分に行い、キーボードによる入力や GUI 操作に関する挙動が含まれるログを取得することによって、マルウェアと区別しやすくなると考えられる。

今回、ネット切断環境下およびネット接続環境下において特徴量の個数を統一するため、特徴量の対象システムコールを、ネット切断環境下およびネット接続環境下で検体実行時に発行されたシステムコールの種類数の和集合とした。そのため、特徴量が 55696 個と膨大な数であったため、マルウェアを区別する上で重視されていない特徴量も含まれていた可能性がある。また、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いて、ネット切断環境下およびネット接続環境下で検体実行時に発行されたシステムコールの種類には差があった。各環境下で取得したシステムコールログ内に出現したシステムコールの種類数についてまとめたものを表 7 に示す。表 7 より、クリーンウェア

表 8 各環境下においてのみ出現したシステムコール

	種類数	システムコール
切断のみ	3 種類	NtSetEventBoostPriority, NtDelayExecution, NtQueryFullAttributesFile
接続のみ	100 種類	NtCreateTimer, NtUserSetWindowFNID, NtUserGetClassInfo, NtUserGetImeInfoEx, NtUserUpdateInputContext, etc.

において、ネット接続によって出現するシステムコールの種類が約 2 倍増加していることが分かる。マルウェアにおいても、ネット接続によって出現するシステムコールの種類が増加しているが、クリーンウェアほどの差はないことが分かる。また、ネット切断環境下においてのみ出現したシステムコールは 3 種類、ネット接続環境下においてのみ出現したシステムコールは 100 種類存在した。それぞれについてまとめたものを表 8 に示す。ネット接続環境については、数が多いため一部のみまとめる。以上より、今後、どのシステムコールの遷移が重要であるかを明確にした上で特徴量を削減し、マルウェアを区別する上で重視されるシステムコールの遷移のみに絞ることで、検知精度の向上が期待できる。

5. 関連研究

東らの研究 [11] では、性質の異なるデータセットを分布が異なるデータセットとし、仮説検定を用いることで、データセットの分布の違いを数値として表現可能かを検証している。既存研究の提案手法における評価は、あくまで手元にあるデータセット内での評価であり、研究と実環境下では使用時にギャップが生じる。ギャップを埋めるための評価には、性質の異なるデータセットが 2 種類以上必要であるが、データセットの性質を表現することが困難であることから、データセットが異なることを表現する新たな指標の検討が行われている。一方、本論文では、既存研究で使用されているデータセットがバラバラであり、自身の提案手法の客観的な評価が行えないといったことから、評価用のデータセットの作成を目指している。

Xiao らの研究 [12] では、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確率を用いてマルウェアを検出する手法を提案している。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、マルコフ連鎖内の遷移確率を比較することで状態遷移の異常を捉えている。この手法では、マルコフ連鎖内のシステムコールの遷移確率を用いて検知しているが、システムコールシーケンスの遷移情報を用いているという点において類似している。また、本論文では、Android マルウェアではなく、Windows マルウェアを対象として検知

している。

中里らの研究 [13] では、プロセスによって呼び出される個々のスレッドの挙動に焦点を当て、スレッドにおける API 呼出しシーケンスから得られる挙動の特徴を基に、マルウェアを効率的かつ正確にクラスタ化する分類手法を提案している。この手法では、マルウェアの挙動の特徴を表すものとしてスレッドから API 呼出しシーケンスを抽出しているが、本論文ではシステムコールシーケンスを抽出して使用している。また、マルウェアの分類ではなく、検知を行うためにスレッドの挙動に着目している。

6. おわりに

本論文では、動的解析システムのネットワーク接続の有無が与える影響に焦点をあて、マルウェアを特徴付ける情報としてスレッドに着目し、スレッド情報に基づいた特徴量を生成することでシステムコールベースのマルウェア検知を行った。具体的には、ネット切断環境およびネット接続環境において Alkanet を利用し、取得したシステムコールトレースログを用いてスレッド情報に基づいた特徴量を生成し、学習データを作成することでシステムコールベースのマルウェア検知を行った。その結果、ネット接続することで検知精度が下がる傾向にあることが分かった。この原因として、4 章で述べたことが考えられる。

今後は、4 章で述べた事柄に加え、ネット接続によって精度が下がった原因として考えられる事柄を解消した上で再度取得したログを用いたマルウェア検知について検討するなど、ネット接続することで精度が下がった原因の調査をさらに進める必要がある。また、本論文では、マルウェアを特徴付ける情報としてスレッドを使用した。他の情報については検討できていない。そのため、スレッド以外の情報についても検討する必要がある。

参考文献

- [1] キヤノンマーケティングジャパン株式会社サイバーセキュリティラボ: 2020 年上半期のマルウェアレポートを公開, <https://eset-info.canon-its.jp/malware.info/trend/detail/200917.html> (2021.1.14 アクセス).
- [2] Suci, O., Coull, S. E. and Johns, J., Exploring Adversarial Examples in Malware Detection, 2019 IEEE Security and Privacy Workshops (SPW), pp. 8–14, 2019.
- [3] Jung, J., Kim, H., je Cho, S., Han, S. and Suh, K., Efficient Android Malware Detection Using API Rank and Machine Learning, J. Internet Serv. Inf. Secur., Vol. 9, pp. 48–59, 2019.
- [4] 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, Vol. 55, No. 9, pp. 2034–2046, 2014.
- [5] Shinagawa, T., Eiraku, H., Tanimoto, K., Omote, K., Hasegawa, S., Horie, T., Hirano, M., Kourai, K., Oyama, Y., Kawai, E., Kono, K., Chiba, S., Shinjo, Y. and Kato, K., BitVisor: a thin hypervisor for enforcing

- i/o device security, Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, pp. 121–130, 2009.
- [6] 森本 康太, 鄭 俊俊, 瀧本 栄二, 齋藤 彰一, 毛利 公一, マルウェアの動的解析におけるログ出力が停止する現象の実態調査, 情報処理学会研究報告, Vol. 2020-SPT-36, No. 25, pp. 1–8, 2020.
 - [7] 梶原友希, 鄭俊俊, 毛利公一, 対象スレッドの違いによるマルウェア検知精度の比較, 情報処理学会研究報告, Vol. 2020-CSEC-90, No. 8, pp. 1–7, 2020.
 - [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 回科学技術フォーラム (FIT2019), Vol. 4, pp. 195–198 (L-033), 2019.
 - [9] 荒木粧子, 笠間貴弘, 押場博光, 千葉大紀, 畑田充弘, 寺田真敏, マルウェア対策のための研究用データセット～MWS Datasets 2018～, 研究報告コンピュータセキュリティ (CSEC), Vol. 2018-CSEC-82, No. 38, pp. 1–8, 2018.
 - [10] 株式会社インプレス: 窓の杜, <https://forest.watch.impress.co.jp/> (2021.1.14 アクセス).
 - [11] 東結香, 津田侑, データセットの分布の違いを表現する指標の検討と予測結果の関係性分析, コンピュータセキュリティシンポジウム 2019 論文集, Vol. 2019, pp. 1059–1066, 2019.
 - [12] 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, Vol. 11 Iss. 1, pp. 8–15, 2017.
 - [13] NAKAZATO, J., SONG, J., ETO, M., INOUE, D. and NAKAO, K., A Novel Malware Clustering Method Using Frequency of Function Call Traces in Parallel Threads, IEICE Transactions on Information and Systems, Vol. E94.D, No. 11, pp. 2150–2158, 2011.