IoTマルウェアの系統樹クラスタリングにおける機能面に踏み込んだ解析のための距離定義の改良

二川 功佑[1,a)]　韓 燦洙[2]　田中 智[2]　岩本 一樹[3]　高橋 健志[2]　竹内 純一[1]

概要：IoT 機器をターゲットとしたマルウェアによる攻撃が増加している．IoT マルウェアは攻撃者のソースコード改変により多様に進化しており，大量の検体に対して詳細な機能情報を効率的に解析することが求められる．機能面に踏み込んだ解析のためには，検体の実行可能ファイル内から有用な情報を取り出して用いることが必要である．本研究はマルウェアの系統樹クラスタリングにおける，詳細な機能情報の解析のための検体間距離定義に焦点を当てている．実行ファイルに記述されているコードはライブラリ関数と攻撃者が独自に記述したと考えられるユーザーコードに大別でき，検体の詳細な機能の差異を表す情報はユーザーコードのみに含まれていると推測できる．そこで，ユーザーコードやライブラリ関数等の情報の有無が機能ラベルに対するクラスタリング精度にどの程度影響するのかを調査するため，IoT マルウェア 4801 検体及び 816 検体の実データセットからこれらの情報を選択的に組み合わせたファイルデータセットを 6 つ作成し距離定義，系統樹クラスタリングをする実験を行った．クラスタリング精度を評価する機能ラベルには無害化情報と Exploit 情報を用いた．その結果，ユーザーコード部のオペコードの情報のみを用いることによる機能ラベルに対するクラスタリング精度の向上を確認した．

キーワード：IoT マルウェア，系統樹，クラスタリング，正規化圧縮距離

Improvement of Distance Definition for In-Depth Functional Analysis in Phylogenetic Clustering of IoT Malware

Kosuke Nikawa[1,a)]　Chansu Han[2]　Akira Tanaka[2]　Kazuki Iwamoto[3]　Takeshi Takahashi[2]
Jun'ichi Takeuchi[1]

Abstract: IoT malware evolves diversely through the modification of source code by attackers, necessitating the efficient analysis of detailed functional information across a large number of samples. For a detailed functional analysis, it is necessary to extract and utilize relevant information from the executable files of the samples. This study focuses on defining inter-sample distances for the analysis of detailed functional information in the context of malware phylogenetic clustering. The code described in an executable file can generally be classified into library functions and user code, which is presumed to be independently written by the attacker. It is hypothesized that the information reflecting detailed functional differences among samples is contained only in the user code. We conducted experiments in which we investigated the change in clustering accuracy based on functional labels, when user code and library functions are present or absent. The results confirmed an improvement in clustering accuracy concerning functional labels when using only the opcode information from the user code sections.

Keywords: IoT Malware, Phylogenetic Tree, Clustering, Normalized Compression Distance

[1] 九州大学
Kyushu University
[2] 国立研究開発法人情報通信研究機構
National Institute of Information and Communications Technology
[3] 株式会社日立システムズ
Hitachi Systems, Ltd.
[a)] nikawa@me.inf.kyushu-u.ac.jp

1. はじめに

近年，Internet of Things（IoT）デバイスの普及が急速に進んでおり，IoT デバイスを標的としたマルウェアによる攻撃が増加している．IoT マルウェアの特徴として，攻撃者による改変が挙げられる．攻撃者は目的を達成するため，Mirai [5] や Bashlite [12] といったソースコードが一般に公開されているマルウェアを部分的に改変することで新たなマルウェアを作成し攻撃に用いる．多様に進化した大量のマルウェアの機能を効率的に解析することが求められており，系統樹を利用したクラスタリングの研究が進められている．系統樹は生物進化学の分野で，生物種間の進化関係を表現するために使用される木構造のグラフである．ソースコードの改変によるマルウェアの進化関係を系統樹に表し，系統樹を適切に分割することで類似した検体のクラスタリングが可能である．

本研究は，系統樹を作成するための検体間の距離定義に焦点を当てており，何ら [2] の研究の発展研究に位置づけられる．何ら [2] による先行研究では，マルウェアの実行可能ファイルをそのまま用いて正規化圧縮距離（Normalized Compression Distance, NCD）[1] により距離を定義し，系統樹を作成していた．NCD はその直感的な原理と，ファイル形式を問わない点においてマルウェア間距離定義に適している．しかし，NCD を用いる上でマルウェアをより詳細な機能ラベルについて踏み込んでクラスタリングするためには，検体間距離定義に用いる情報をマルウェアの機能を表すものに絞り込み，冗長な情報を取り除く必要があると考える．**本研究では（1）機能の差異を表す情報がユーザーコードに含まれる（2）データやオペランド等の不要な情報を除去することでクラスタリング精度を向上させることができるという 2 つの仮説を検証する実験を行った．**

そこで，ユーザーコードやライブラリ関数，データやオペランド等の情報の有無が機能ラベルに対するクラスタリング精度にどの程度影響するのかを調査することを目的として，IoT マルウェアデータセットから各情報を選択的に組み合わせたファイルデータセットを 6 パターン作成した．その後，何ら [2] の系統樹クラスタリングアルゴリズムを使用して各ファイルデータセットごとにクラスタリングし，機能ラベルである無害化情報 [14] と Exploit 情報を用いて精度を評価，比較する実験を行った．無害化情報は IoT マルウェアの動作を終了させる等 IoT マルウェアの無害化に関する情報であり，実験では IoT マルウェア 4801 検体から抽出された 175 種類の無害化情報を評価に用いた．Exploit 情報は検体がどのような脆弱性に対して攻撃に用いられたかを表す情報であり，横浜国立大学の吉岡らによって IoT マルウェア 816 検体に付与された 35 種類の Exploit 情報を評価に用いた．また評価指標には Homogeneity [9] を用いた．その結果，ユーザーコードのオペコードのみの情報を用いることで，先行研究に比べ Homogeneity [9] による評価値が無害化情報で 0.1 ほど上昇することを確認した．Homogeneity はクラスタ内のデータがどれだけ同じラベルを持つかを測定する指標であり，この値が 1 に近いほど純粋なクラスタリング結果であることを表す．さらにユーザーコードのみを用いることで無害化情報に対するクラスタリング精度が向上することを確認し，ユーザーコードが無害化情報と密接に関連していることを明らかにした．加えてオペランドやデータセクション等の不要な情報の削減によるクラスタリング精度の向上を確認した．

2. 先行研究

本研究は何ら [2] の研究の発展研究に位置づけており，本章では先行研究である何ら [2] の系統樹クラスタリングアルゴリズムを紹介する．本アルゴリズムはマルウェア検体のバイナリファイルを用いて検体データセットから系統樹を作成し，情報量基準に基づいて系統樹を分割することでクラスタを作成する．高速に系統樹を形成する手法により，スケーラブルな解析を可能としている．大きく，検体間距離定義，系統樹の作成，クラスタリングの 3 ステップに分けられる．

2.1 検体間距離定義

二つのマルウェアの距離を NCD [1] を用いて定義する．NCD はデータの圧縮列長を利用した，0 から 1 の数値で表される距離である．以下にその表式を示す．

$$\mathrm{NCD}(x_i, x_j) = \frac{C(x_i x_j) - \min\{C(x_i), C(x_j)\}}{\max\{C(x_i), C(x_j)\}} \quad (1)$$

$x_i x_j$ は二つのデータ x_i と x_j を繋げた結合データである．$C(x_i)$ はデータ x_i を圧縮したときの圧縮列長を表す．圧縮アルゴリズムの性質により x_i と x_j が類似しているほど $C(x_i x_j)$ は $C(x_i)$ や $C(x_j)$ に近い値となり，NCD は 0 に近づく．データセット $\{x_1, x_2 \ldots, x_N\}$ 内の二つのデータ x_i, x_j に対して, データの類似度を表す距離を (i, j) 要素にもつ行列を距離行列という．何ら [2] はマルウェア検体のバイナリデータを用いて NCD を算出し，検体データセットの距離行列を作成した．圧縮アルゴリズムには Lempel-Ziv-Markov chain アルゴリズム (LZMA) [11] を使用した．

2.2 系統樹の作成

ここでは代表的な系統樹作成手法である近隣結合法 [10] の改良手法である高速近似法 [2], [13] を紹介する．

高速近似法 [2], [13] はデータセットを適切に分割して近隣結合法による系統樹を近似的に作成するアルゴリズムである．その概略図を**図 1** に示す．まずデータセットから種集合として少数の検体をランダムに選び (1)，近隣結合法で

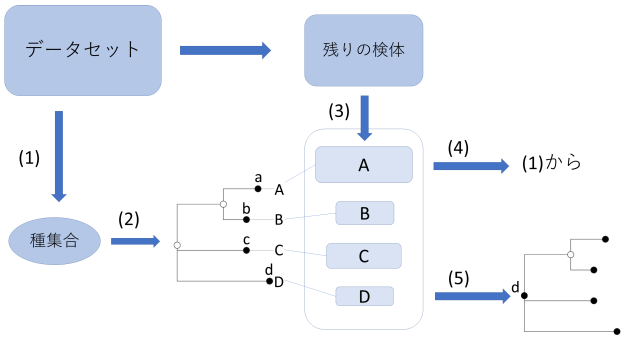


図 1: 高速近似法の全体図

種集合の系統樹を作成する (2)．その後残りの検体を、種集合の各データの内距離が最も近いものに分類する (3)．分類後の種データを含む各データ集合に対して，データ数が閾値より多ければ再度そのデータ集合内で種集合の選択から行い (4)，閾値より少なければ近隣結合法で系統樹を作成する (5)．データ数が N のとき，近隣結合法は距離行列全体を参照するため $O(N^2)$ の距離行列計算コストを要したが，距離行列の一部を参照する高速近似法では $O(N \log N)$ の計算コストで系統樹を作成することが可能である [2]．

2.3 クラスタリング

作成した系統樹を記述長最小原理 (MDL 原理) [8] を用いてクラスタリングする [2]．MDL はモデルを与えた時のデータの記述長とモデル自体の記述長の和が最小となるモデルを最良とするモデル選択原理である．クラスタ C に対して記述長は以下で計算される [2]．

$$\begin{aligned} \mathrm{DL}(C) &= -\log L(x^n|\hat{\theta}) + \frac{m_j}{2}\log n \\ &= \frac{n}{2}\log\left(\frac{1}{n}\sum_{i=1}^{n}|x_i - \bar{x}|^2\right) + \frac{m_j}{2}\log n + \frac{n}{2}(\log 2\pi + 1) \end{aligned} \tag{2}$$

x_i は各データ，$\bar{x}$ はデータの平均値を表す．式 (2) において，データの平均値からの距離が正規分布に従うと仮定して変形したものが式 (2) である．m_j はここではモデルの複雑さの影響を調整する自由パラメータであり，このパラメータを変化させることによって最終的に作成されるクラスタ数を制御することができる．

作成した系統樹の各内部ノードは他の 3 つのノードと接続しており，一つの内部ノードを取り除いて木を 3 つの部分木へと分割することができる．分割前のクラスタ記述長と分割後のクラスタ記述長を比較し，分割後の方が記述長が短ければ分割を実行，そうでなければ分割前がより良いクラスタであるとして分割を棄却する．これを再帰的に実行して最終的なクラスタを決定する．

3. 研究目的

本研究の目的は，NCD を用いてより適切に検体間の機能の差異を表す距離を定義し，機能情報に対するクラスタリング精度を向上させることである．次の二つの仮説を基に，実験を行った．

- **仮説 (1)：検体の詳細な機能に関する情報はユーザーコードのみが持つ．**
- **仮説 (2)：データやオペランド等の不要と考えられる情報を除去することで，機能の差異をより良く表現できる距離を定義できる．**

3.1 仮説 (1)

本研究では機能をマルウェアの実行時の挙動と定義する．この定義に基づくと，検体バイナリファイル内のコードが記述されているセクションが機能を表していると考えられる．ユーザーコードはこの内，ライブラリ関数以外のコードのことを指す．IoT マルウェアはファミリによって用いられるツールチェーンがおおよそ決まっており [16]，ライブラリはツールチェーンの一部であるため，ライブラリ関数はファミリの情報を持つ．一方で，ファミリよりも詳細な機能情報や検体の機能の差異が存在するのはユーザーコードに絞られると考えることができる．

3.2 仮説 (2)

NCD はデータ間に共通のパターンが多いほどそれらの結合データは短く圧縮されるという圧縮の性質に基づいた直感的な距離であり，距離定義に用いたデータとそこから得られるクラスタリング結果の関係が理解しやすい．検体間距離において冗長な情報はクラスタリングの精度を低下させる要因になり，一方で機能に関する情報のみを距離定義に用いることでより詳細な機能面に踏み込んだクラスタリングができる可能性がある．バイナリファイルの中で，データセクションとヘッダや，オペランドといった情報は NCD を使用した機能面でのクラスタリングに不要であると考える．以下にその理由を示す．

- データセクションとヘッダ

機能はコード部が表現しており，変数などを保持する領域であるデータセクションはマルウェアの機能に関するものではないと考えられる．同様にヘッダについても挙動を直接表現する情報ではないため，不要であると考える．

- オペランド

プログラムの各命令に対するメモリやレジスタの割り当ては，それらをコンパイル時に決定する静的割り当てと実行時に決定する動的割り当ての二種類が存在する．IoT デバイスはリソースに制約がある場合が多く，IoT マルウェアはほとんどがシンプルかつ軽量なプログラムであるため，動的なメモリ管理を必要とせず，基本的に静的割り当てによってメモリやレジスタは決定されている．こうした

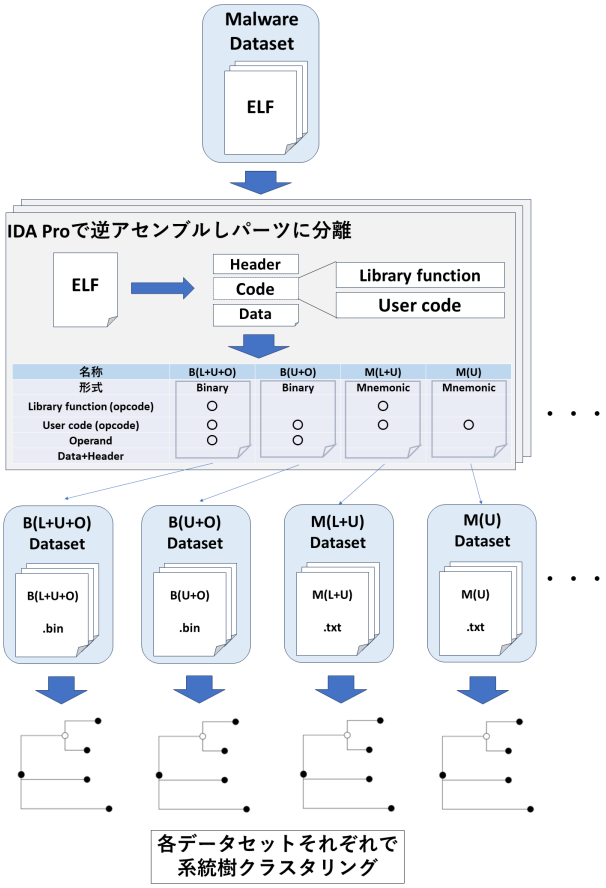


図 2: 提案手法の全体図

メモリやレジスタ等を表現するオペランドはコンパイラによって最適化処理等を通して決定されるため，機能が類似した検体同士であってもその値は大きく異なる可能性があり，機能を直接表現するものではないと考えられる．

4. 提案手法

本研究における提案，比較手法を説明する．その概略図を**図 2** に示す．

4.1 解析対象とする IoT マルウェア

IoT マルウェアの多くは Linux 環境で実行可能な Executable and Linkable Format(ELF) 形式であり，ELF 形式の IoT マルウェアを対象とする．また本研究はユーザーコード部のクラスタリング精度への寄与を調べることを目的としており，そのためにはプログラムからユーザーコードを特定する必要がある．特定には逆アセンブルを使用しているため，扱う検体は逆アセンブル可能であることを前提としており，パッキングと呼ばれる難読化処理のされていない検体やすでにアンパックされた検体を解析対象とする．

4.2 ユーザーコード抽出

まず，IoT マルウェアデータセット中の各検体の実行可能ファイルに対して IDA Pro を使用して逆アセンブルを行い，IDA F.L.I.R.T. [3] により検体に静的リンクされたライブラリ関数を特定する．次にコードセクション中の各コードを，ライブラリ関数とユーザーコードの二つに区別する．具体的には，セグメントと呼ばれるコードとデータが含まれる各ブロックを先頭から末尾まで順に走査し，コードセクション内の各関数がライブラリ関数であるかを判定する．判定の結果ライブラリ関数でないと判別された，以下のものをユーザーコードとして扱う．

(1) 関数でないコード
(2) 関数であるがライブラリ関数ではないもの

4.3 オペランド除去

オペランドは機能を直接表すものではないという仮説の検証のため，命令からオペランドを除去してオペコードのみを取り出す．しかし命令セットアーキテクチャ（Instruction Set Architecture，ISA）によって機械語における命令の記述方法は大きく異なり，命令は各 ISA 固有の方式で機械語にエンコードされているため，命令を機械語のまま扱う場合オペコードをバイナリの状態で取り出すことは容易ではない．加えてオペコードをバイナリで抽出する機能は IDA Pro 等の逆アセンブラでもサポートされていない．そこで本研究ではオペコードをニーモニックとして取り出すことでコードのオペコードシーケンスをテキストデータとして収集し，オペランドを除去する．IDA がサポートしている print_insn_mnem 関数は引数に命令のアドレスを与えることでその命令のオペコードをニーモニック形式で返す．この関数を使用して命令からオペコードを取得し，オペコードシーケンスとしてテキストファイルに記述することでオペランドを取り除いたコードを表現する．

4.4 抽出データセット作成

(L): Library function, **(U):** User function, **(O):** Operand, **(D):** Data + Header の 4 つの項目について，それぞれのパーツを選択的に組み合わせることにより任意の情報を持ったファイルを作成する．

図 2 中の表を用いて説明する．例として名称が B(L+U+O) のファイルは，形式がバイナリであり，オリジナルの実行可能ファイルの情報の内，ライブラリ関数，ユーザーコードの各オペコードとオペランド部分を持っている．同様に，B(U+O) のファイルはオリジナルの実行可能ファイルの内，ユーザーコード部分のオペコードとオペランドを保持している．このように，ファイルの名称はその形式と，4 つの項目の内保持している情報の頭文字をとって決定されている．オペランド部分を持つとする場合，それは保持しているオペコードに対応したものを指してお

```
"dupcmd": "DUP",
"init": "¥u001b[0;32m[CONNECTED] [%s] [%s]",
"prefix": 33,
"termcmd": "LOLNOGTFO"
```


図 3: Bashlite 検体の無害化情報ラベルの例（上からマルウェア重複起動時の終了コマンド，初期通信内容，コマンド接頭辞，キルコマンド）


```
"capture_date":"2020/06/03"
"port":"37215"
"vulnerability:"CVE-2017-17215"
"vulnerability_type":"Remote Code Execution (RCE)"
"device":"Huawei home routers HG532"
```


図 4: Mirai 検体の Exploit 情報ラベルの例（上から検体収集日，ポート番号，脆弱性識別子，脆弱性の種類，ターゲットデバイス）

り，例えば B(U+O) はユーザーコード部のオペランドを保持しているがライブラリ関数部のオペランドに関しては保持していない．またファイルは各項目を単純に結合して作成するのではなく，各コードや関数の順番はオリジナルの実行可能ファイル内での先頭からの出現順に準拠する．本研究ではコード部に関する情報のクラスタリング精度への寄与を調査するため，データ部やヘッダ部といったコード以外のセクションは仮説に基づき一つの項目としている．

オペランドを除去したファイルについては，先述したようにオペコードがニーモニックとしてテキストデータで表現される．例として M(L+U) はライブラリ関数とユーザーコード部のオペコードシーケンスが記述されたテキストファイルである．

データセットの各検体について同様に，様々な組み合わせのファイルを作成する．その後，同じ項目を保持している同種のファイルを集めてデータセットを作成する．本研究で使用する作成データセットは，**表 1** に示す 7 つである．B(L+U+O+D) はすべての項目を保持しており，すなわちこれは抽出前のオリジナルのデータセットを表す．

4.5 クラスタリング精度の比較

各データセットそれぞれで，系統樹クラスタリングを行う．クラスタリングアルゴリズムは 2 章で紹介した何ら [2] の手法を使用し，機能ラベルを用いてクラスタリング精度を評価する．データセット間の精度を比較することで，ユーザーコード等の各情報の有無が機能面でのクラスタリングにどれだけ影響を与えるかを調査する．

5. 実験

5.1 評価指標

本研究ではクラスタリング精度の評価指標として Homogeneity [9] を用いる．式 3 において，C はクラス集合，K はクラスタ集合を表す．これは，既知であるクラスと取得したクラスタリング結果それぞれのデータ分布によって計算され，クラスタがどれほど同じクラスを持つデータから構成されるかを評価する．Homogeneity は 0 から 1 の値をとり，1 に近いほどクラスタリング結果がクラス構造をよく表現しており精度が高いと評価できる．

$$\text{Homogeneity} = \frac{I(C;K)}{H(C)} = 1 - \frac{H(C|K)}{H(C)} \qquad (3)$$

5.2 無害化情報による評価

無害化情報 [14] には，IoT マルウェアの初期通信に関する情報やキルコマンド [15] が含まれる．IoT マルウェアは実行初期段階において Command and Control サーバ (C&C サーバ) との通信の確立する．初期通信とはこのとき，サーバに最初に送信するペイロードを指す．また IoT マルウェアには攻撃者からの指令によりその動作を自ら停止させる機能を持つものも存在する．実行停止にはキルコマンド [15] と呼ばれるコマンドが用いられる．こうした，初期通信の情報やキルコマンド等はうまく利用すれば IoT マルウェアを無害化することのできる情報であるため，これらをまとめたものを検体の無害化情報ラベルとしている．ある Bashlite 検体から抽出された無害化情報を**図 3** に示す．

5.2.1 データセット

本実験では IoTPOT [6] と X-POT [4] により 2021 年 6 月〜2022 年 3 月の間に収集された IoT マルウェア 5771 検体の内，無害化情報が抽出された 4801 検体を用いた．ISA は全て ARM 及び MIPS の二種類であった．

4801 検体から確認された無害化情報は 175 種類であった．検体と無害化情報ラベルは多対一の関係であり，複数の検体から同一の無害化情報が抽出されたものも存在する．

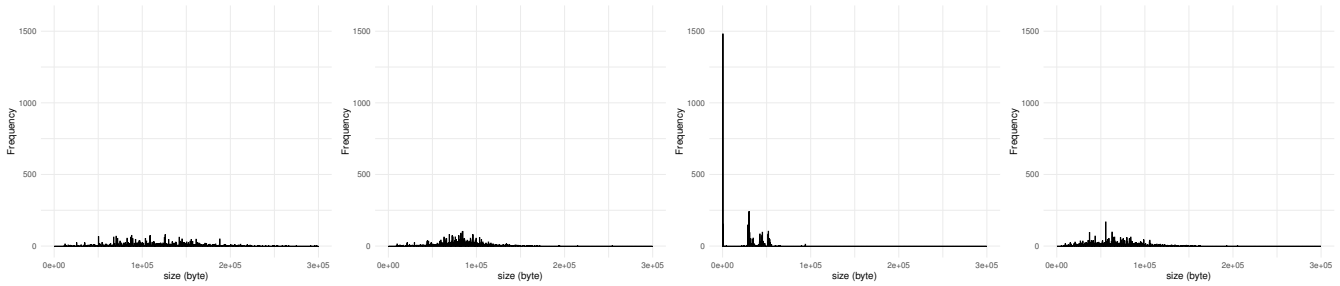
5.2.2 結果：Binary

オリジナルデータセットから作成した，表 1 中の形式が Binary である 4 つのデータセットの結果を比較する．これら 4 つのデータセットのファイルサイズ分布を**図 5** に示す．ただしサイズが極端に大きいものは除いている．B(L+O) はサイズが 0 byte のファイルこそ無いものの，0 byte に近いものが多数存在した．これらはライブラリ関数が動的リンクされている検体や，IDA F.L.I.R.T. によって正しくライブラリが特定されなかった検体から作成されたファイルである．

クラスタリング結果を**図 6a** に示す．図の横軸はクラスタ数，縦軸は評価値である Homogeneity である．ライブラリ関数とユーザーコードを併せ持つ B(L+U+O) を基準にすると，そこからユーザーコードを除去したデータセットである B(L+O) は評価値が低下しているのに対し，一方でライブラリ関数を除去したデータセットである B(U+O) の評価値は上昇している．また B(L+U+O+D) はマルウェア

表 1: 表の例

名称	B(L+U+O+D)	B(L+U+O)	B(L+O)	B(U+O)	M(L+U)	M(L)	M(U)
形式	Binary	Binary	Binary	Binary	Mnemonic	Mnemonic	Mnemonic
Linrary function	○	○	○		○	○	
User code	○	○		○	○		○
Operand	○	○	○	○			
Data + header	○						



(a) B(L+U+O+D) (b) B(L+U+O) (c) B(L+O) (d) B(U+O)

図 5: 4801 検体のサイズ分布

の実行可能ファイルをそのままクラスタリングに用いた場合の結果であるが，ライブラリ関数のみに絞った B(L+O) と同程度の評価となっていることが分かる．

5.2.3 結果：Mnemonic

次に，表 1 中の形式が Mnemonic である 3 つのデータセットを，参考として B(L+U+O+D) と合わせて比較する．結果を**図 6b** に示す．こちらも同様に，ライブラリ関数とユーザーコードのオペコードを持つ M(L+U) を基準に比較する．ユーザーコードに関する情報を除去した M(L) では評価が大きく下がってしまったが，一方でライブラリ関数の情報を除去した M(U) では，広いクラスタ数の範囲においてわずかに精度が上昇した．

5.2.4 考察：無害化情報

Binary と Mnemonic の両方で，B(L+U+O) や M(L+U) を基準にライブラリ関数を取り除くと精度が向上しユーザーコードを取り除くと精度が低下した結果となった．この対照的な結果は，無害化情報という情報が検体のユーザーコードでのみ詳細に判別できることを示している．このことから，無害化情報については仮説（1）は正しいと言える．また M(L) は精度が大きく低下しているが，先述したようにライブラリ関数はファミリを表しており，ライブラリ関数のオペコードシーケンスの情報のみでは検体間の差異が大きく失われてしまったためであると考えられる．

7 つのデータセットの結果を一つのグラフにプロットしたものが**図 6c** である．ファイル形式が異なるため一概に Binary と Mnemonic を比較することはできないが，最も精度を向上させることができたのは M(U) であり，ライブラリ関数やオペランド等の情報を除去したデータセットであった．このようにユーザーコードのオペコードの情報のみで最も高い精度を達成することができたため，無害化情報に関して仮説（2）についても正しいと言える．

5.3 Exploit 情報による評価

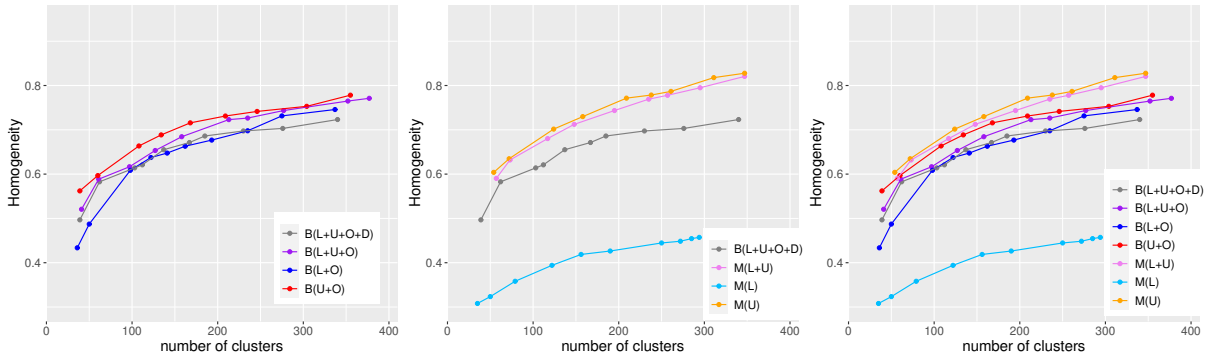
IoT マルウェアが，どのような脆弱性を利用して攻撃に使用されたかをまとめたものを，その検体の Exploit 情報という．Exploit 情報にはその脆弱性情報の識別名や利用されたポート番号，ターゲットのデバイス名などが含まれる．ある Mirai 検体の Exploit 情報の一部を**図 4** に示す．本研究では共通脆弱性識別子（Common Vulnerabilities and Exposures, CVE）番号 [7] などで表される，“vulnerability” をラベルに用いた．

5.3.1 データセット

実験ではラベルの関係上，無害化情報の評価に用いたものとは異なる IoT マルウェア 816 検体を用いた．検体は 2020 年から 2021 年に収集されたもので，Exploit 情報は横浜国立大学の吉岡らによって検体に付与されたものを使用した．816 検体から確認されたラベルの種類数は 35 種類であった．検体のファミリは全て Bashlite であり，ISA は MIPS であった．検体と Exploit 情報は多対多の関係にあり，一つの検体が複数の Exploit 情報を持つ場合や，またある Exploit 情報が複数の検体に対応している場合が存在した．

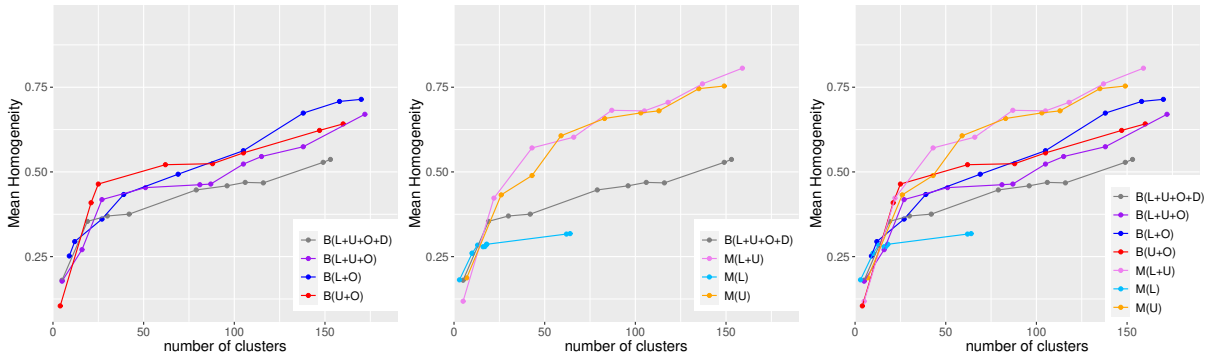
5.3.2 平均 Homogeneity

検体とラベルが多対多の関係の場合，クラス分布において検体が複数のクラスに所属しているケースが存在するため，単純な Homogeneity の算出が不可能である．存在しう



(a) Binary 形式のデータセット　(b) Mnemonic 形式のデータセット　(c) 全データセット

図 6: 無害化情報による評価結果



(a) Binary 形式のデータセット　(b) Mnemonic 形式のデータセット　(c) 全データセット

図 7: Exploit 情報による評価結果

るラベルの各組み合わせをそれぞれクラスとすることも考えられるが，ラベルの種類数が N のときそのクラス数は最大で $2^N - 1$ となり，クラスタ数に対してクラス数が超過となってしまう．そこで，各ラベルについてそのラベルに属するか否かの 2 クラスで Homogeneity を算出し，クラスタリング結果が各ラベルをどれだけ表現できているかを評価する．その後，得られた N 個の Homogeneity の値を平均した平均 Homogeneity によってクラスタリング精度を評価する．

5.3.3 結果：Binary

まず無害化情報の時と同様，表 1 中の形式が Binary である 4 つのデータセットによるクラスタリング結果を比較する．結果を**図 7a** に示す．図の横軸はクラスタ数で，縦軸は評価値である平均 Homogeneity である．B(L+U+O) を基準に比較すると，ライブラリ関数のみの B(L+O) とユーザーコードのみの B(U+O) の両方で精度の向上が見られた．特にクラスタ数 100 付近を基準に，それより少ない範囲ではユーザーコードの方が良い評価であるが，一方で 100 より多い範囲ではライブラリ関数による系統樹の方が良い評価であった．またいずれも B(L+U+O+D) より高い精度となっていた．

5.3.4 結果：Mnemonic

次に，表 1 中の形式が Mnemonic である 3 つのデータセットを，参考として B(L+U+O+D) と合わせて比較する．その結果を**図 7b** に示す．M(L+U) と M(U) が同程度の評価であり，M(L) が著しく低い結果となった．M(L) はクラスタの分割が早期に停止してしまい，最大のクラスタ数が 63 であった．

5.3.5 考察：Exploit 情報

Binary と Mnemonic の両方において，B(L+U+O) や M(L+U) からライブラリ関数部を取り除くことによる精度の向上が見られた．一方でユーザーコードを取り除いた場合，Binary では同様に精度の向上が見られており，この結果はユーザーコードのみに機能情報が存在するとする仮説（1）に反している．Mnemonic では，M(L+U) と比較して M(L) の精度は大きく低下した．これは無害化情報の時と同様，検体間の差異が大きく失われてしまったからであると考えられる．特に今回用いた検体は全てファミリが Mirai であり，早期にクラスタを分割することができなくなったことが分かる．ライブラリ関数部に差異が無いため，M(L+U) と M(U) の結果は同程度であるとも考えられる．最も良い精度であったのは M(L+U) と M(U) であり（**図 7c**），データやオペランド等の不要な情報削減による

評価の向上が見られたため，仮説（2）については正しいと結論づけた．

5.4 異なる圧縮アルゴリズムを用いた NCD

本研究では先行研究と同様，圧縮アルゴリズムに LZMA を使用したが，これは LZMA がバイナリデータにおいて最も効率的にパターンマッチングをすることができるアルゴリズムであったためである [2]．ニーモニックを記述したテキストデータを用いることで先行研究よりも高いクラスタリング精度を達成したが，テキストデータの圧縮においては LZMA に限らず他の圧縮アルゴリズムを用いることができる．そこで M(U) データセットを用いて，Linux コマンドで標準的に使用可能な bzip2，gzip，Compress（LZ78）を使用してクラスタリングを行った．無害化情報を用いて評価した結果，いずれも LZMA でのクラスタリング精度を超えることはできず，バイナリデータ同様，テキストデータでの NCD も LZMA が最適であった．

6. おわりに

本研究では系統樹クラスタリングにおいて，機能面に踏み込んだ解析のための NCD を用いた新たな検体間距離定義方法を提案し，2 つの仮説を提案した．実験では無害化情報と Exploit 情報について，ユーザーコードやライブラリ関数などの情報を取捨選択することによるクラスタリング精度の変化を調査した．結果として，無害化情報についてユーザーコードのみを用いることでクラスタリング精度を向上させることができた．またオペランドやデータ等の不要と考えられる情報を除去することで NCD による検体間距離定義を改善することができた．（1）機能の差異を表す情報がユーザーコードに含まれる（2）データやオペランド等の不要な情報を除去することでクラスタリング精度を向上させることができるという二つの仮説について，無害化情報による評価では仮説（1），（2）共に正しいことを確認したが，Exploit 情報による評価では仮説（2）のみが正しく，仮説（1）については正しいとは言えない結果が得られた．

残された課題としては，Exploit 情報に関してはライブラリ関数のみでも高いクラスタリング精度を得ており，その理由を明らかにできていないこと，およびオペランドを除去する為にニーモニックを使用したが，テキストファイルというファイル形式の違いが NCD やクラスタリング精度に及ぼす影響を分析できていないことなどがある．その他課題として，ユーザーコードのみでクラスタリングをするためには全ての検体についてライブラリ関数除去をする必要があり，NCD のメリットの一つである簡便さが失われていることが挙げられる．

謝辞 本研究は総務省の「電波資源拡大のための研究開発 (JPJ000254)」における委託研究「電波の有効利用のための IoT マルウェア無害化／無機能化技術等に関する研究開発」によって実施した成果を含む．また，JSPS 科研費 JP23H05492 の助成を受けた．データを提供してくださいました横浜国立大学の吉岡克成先生に深く感謝する．

参考文献

[1] Cebrian, M., Alfonseca, M. and Ortega, A.: The Normalized Compression Distance Is Resistant to Noise, *IEEE Transactions on Information Theory*, Vol. 53, No. 5, pp. 1895–1900 (2007).

[2] He, T., Han, C., Isawa, R., Takahashi, T., Kijima, S. and Takeuchi, J.: Scalable and Fast Algorithm for Constructing Phylogenetic Trees With Application to IoT Malware Clustering, *IEEE Access*, Vol. 11, pp. 8240–8253 (2023).

[3] Hex-Rays: IDA F.L.I.R.T. technology: in-depth, `https://www.hex-rays.com/products/ida/tech/flirt/in\_depth/`.

[4] Kato, S., Tanabe, R., Yoshioka, K. and Matsumoto, T.: Adaptive Observation of Emerging Cyber Attacks targeting Various IoT Devices, *IFIP/IEEE International Symposium on Integrated Network Management* (2021).

[5] NEC: Mirai（マルウェア）, `https://www.nec-solutioninnovators.co.jp/ss/insider/security-words/40.html`.

[6] Pa, Y., Suzuki, S., Yoshioka, K., Matsumoto, T., Kasama, T. and Rossow, C.: IoTPOT: A Novel Honeypot for Revealing Current IoT Threats,, *Journal of Information Processing* (2016).

[7] Red Hat: CVE とは，`https://www.redhat.com/ja/topics/security/what-is-cve`.

[8] Rissanen, J.: Modeling by shortest data description, *Automatica* (1978).

[9] Rosenberg, A. and Hirschberg, J.: V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure., pp. 410–420 (2007).

[10] Saitou, N. and Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees., *Molecular Biology and Evolution*, Vol. 4, No. 4, pp. 406–425 (1987).

[11] Salomon, D.: *Data Compression: The Complete Reference Fourth Edition*, Springer (2007).

[12] threat post: BASHLITE Family Of Malware Infects 1 Million IoT Devices, `https://threatpost.com/bashlite-family-of-malware-infects-1-million-iot-devices/120230/`.

[13] Yone, T.: Phylogenetic tree estimation for large-scale malware datasets, Master's thesis, Kyushu University (2016).

[14] 三須剛史，岩本一樹，奥村吉生，高田一樹，吉岡克成：無害化情報の自動抽出を目的とした IoT マルウェアの静的解析，暗号と情報セキュリティシンポジウム (2021).

[15] 三須剛史，岩本一樹，高田一樹，吉岡克成：IoT マルウェア駆除のためのキルコマンド等の自動抽出，コンピュータセキュリティシンポジウム (2019).

[16] 赤羽秀，岡本剛：シンボル情報が消去された IoT マルウェアに静的結合されたライブラリ関数の特定，コンピュータセキュリティシンポジウム (2020).