Computer Security Symposium 2021
26 - 29 October 2021

n-gramによるマルウェア検出における機械学習を騙す良性ソフトウェア汚染

宇田 隆哉[1,a)]

概要：n-gram によるマルウェア検出手法は動的解析を必要とせず，OS や命令に関する知識も必要としない優れたものである．とくに，n-gram と機械学習を用いた手法は，既存研究において良好なマルウェア検出精度を示している．しかし，既存研究においては，既知のデータセットを使用して，新たなマルウェア検出手法を提案している．一方，実際には，攻撃者に既知のマルウェア検出手法が存在している際に，攻撃者はその対策をした攻撃が可能である．そこで，本論文では，機械学習における良性データセットを汚染する手法を提案する．これは，実行されない悪性コードの断片を持つ良性ソフトウェアを故意に大量に作成し，フリーソフトウェアライブラリに登録するというものである．我々が発表した n-gram 抽出手法は n-gram の出現順序と共起度を特徴とするため，一般的な n-gram を用いたマルウェア検出手法よりも汚染の影響を受けにくいが，評価において一定数の誤検出が確認された．この結果は，本論文の汚染手法が n-gram を用いたマルウェア検出手法に有効であることを示すものであり，攻撃者による対策を考慮する必要性を問うものである．

キーワード：マルウェア検出，n-gram，機械学習，人工知能

Benign Software Pollution for Machine Learning Deception in Malware Detection by n-grams

Ryuya Uda[1,a)]

Abstract: Malware detection methods by n-grams are excellent since dynamic analysis and knowledge about OS etc. are not required. Especially, the methods by n-grams and machine learning mark great scores in existing researches. However, existing datasets are used for evaluating new detection methods in the researches. On the other hand, attackers can take measures to cope with existing malware detection methods in the real world. Therefore, in this paper, I propose a benign software pollution method to deceive machine learning algorithms. Pieces of non-executable malicious codes are added to benign software samples which are registered to free software libraries. Our n-gram extraction method is more resistant from the pollution than usual existing malware detection methods with n-gram since n-gram orders and co-occurrence are contained in characteristics, but some samples were misclassified in evaluation. The result shows the effectiveness of the pollution and requires consideration for thoughts of attackers against malware detection.

Keywords: malware detection, n-gram, machine learning, artificial intelligence

1. はじめに

マルウェアを検出する研究は以前より行われてきた．単純なパターンマッチングでは対応しきれなくなった状況に対して，マルウェアの特徴を捉えて検出するという手法が考案されている．しかし，ほとんどの論文において，新規に提案した手法で既知のマルウェアを検出したという評価しか示されておらず，そのマルウェア検出手法が攻撃者に既知であった場合に，攻撃者がその手法を回避可能かとい

[1] 東京工科大学 コンピュータサイエンス学部
School of Computer Science, Tokyo University of Technology
[a)] uda@stf.teu.ac.jp

う点については論じられていない．

もちろん，マルウェア検出手法が既知であるとした上で，攻撃の有無について論じている論文もある．とくに，深層学習を利用した人工知能が注目を集めている現在，この人工知能を騙すような Adversarial Examples や Adversarial Patch などの研究も行われている．マルウェア検出の分野においては，攻撃者が検出手法を回避するマルウェアを作成できるかという点でこの問題が論じられている．

一方，現実の世界においては，攻撃者が行うのはマルウェア作成だけとは限らない．そこで本論文では，攻撃者が良性ソフトウェアを汚染することを想定し，その影響について論じる．これは，研究環境で行われていた内容が，実世界で利用されることになった際，どのような方法でサンプルの収集や評価が行われるかという点にも踏み込んでいる．この攻撃について論じている論文は，筆者が探した範囲においては存在しなく，新しい着眼点の提案であるといえる．

2. 関連研究

2.1 特定領域のバイナリ n-gram を使用したマルウェア検出手法

バイナリの n-gram を使用してマルウェアを検出する，最も古い手法のひとつに Kephart らによるものがある [1]．Kephart らはウイルスのブートセクタから trigram を作成して使用しており，ソフトウェア全体を使用してはいない．なお，この手法ではブートセクタに特徴が現れないマルウェアは検出できない．

2.2 ソフトウェア全体のバイナリ n-gram を使用したマルウェア検出手法

ソフトウェア全体のバイナリ n-gram を使用してマルウェアを検出する，最も古い手法のひとつに Abou-Assaleh らによるものがある [2]．対象は亜種マルウェアではなくマルウェア全体であるが，最も頻繁に出現する L 個の n-gram を使用している点で，同時期に特定の亜種マルウェアが猛威を振るえば，この L 個にその亜種マルウェアが共通して持つ n-gram が多く含まれるのは必然である．

同様の考えで，情報利得を用いて上位 L 個の n-gram を用いる手法がある．Reddy らの手法 [3]，Boujnouni らの手法 [4]，Kolter らの手法 [5] の手法がこれに該当する．

Raff らは，バイナリの n-gram を使ったいくつかの手法を評価している [6]．Raff らは，評価のために情報利得の高い最初の 20 万個の n-gram を選択しており，その点では Reddy ら，Boujnouni ら，Kolter らと同様である．

Fuyong らも n-gram を用いる手法を提案している [7]．ただし，彼らは個々の n-gram の情報利得を求めて降順にソートし，すべての n-gram を分類に使用している．

なお，ソフトウェア全体のバイナリ n-gram を使用したマルウェア検出手法には問題点が大きく 2 つある．まず，すべての n-gram を機械学習に使用するのは，非常に計算コストが高いということである．Fuyong らは平均ベクトルから値が高いか低いかを基準としているためこれに該当しないが，単純に機械学習をすべての n-gram に適用することは現実的ではない．次に，バイナリ n-gram から上位のものを選択する方法の場合，前処理に膨大な時間が掛かるという問題がある．Raff らは，Hash-Grams という手法でこの問題を解決しており，その論文中で，サンプル数が多いときに，バイナリ n-gram のトップ k を選択する方法はパフォーマンスのボトルネックになっていると述べている [8]．

Hash-Grams とは異なる方法で，n-gram の前処理に膨大な時間が掛かる問題を解決した我々の手法もある [9]．これは，接尾辞配列を用いたもので，Raff らの Hash-Grams において 4-gram などの短い n-gram では効果を発揮できない問題を解決している．

本節で挙げた研究おいて共通するのは，ある n-gram が含まれているかいないかを特徴としている点である．その n-gram が，どの部分にどのような形で含まれているかは考慮されないため，本論文で提案する汚染手法の影響を受けやすいと考えられる．これは，良性ソフトウェアを n-gram で比較すると，そこに問題となる特徴が含まれる可能性があるというものであり，Raff らの論文中でも指摘されている [6]．

一方で，我々が EMM 研究会で発表した論文（以下 EMM 論文）で述べた n-gram 圧縮手法 [10]，および，CSS2019 で発表した論文（以下 CSS2019 論文）で述べた n-gram 抽出手法 [11] においては，n-gram の出現順序や共起度が特徴として保存されるため，本論文で提案する汚染による影響を受けにくいと考えられる．

2.3 コードセクションのみのバイナリ n-gram を使用したマルウェア検出手法

本論文で提案する汚染手法は，特定の亜種マルウェアに共通して出現する n-gram を，良性ソフトウェアのデータセクションに追加するものである．O'Kane らは，命令セットをオプコードとオペランドに分け，オプコードのみの n-gram を作成してマルウェアを検出する手法を提案している [12]．この手法であれば，本論文で提案する汚染の影響は受けない．しかし，オプコードのみの n-gram を作成するには，OS とソフトウェアエンジニアリングに関する知識が必要であり，容易ではない．

また，コードセクションのみを使用してデータセクションを無視する手法は，ソフトウェア全体のバイナリを使用する手法よりも分類精度が落ちる可能性がある．この点については，n-gram を用いた手法ではないが，Yakura らの論文中で検証されている [13]．Yakura らは，バイナリデー

タを画像に変換し，CNN (Convolutional Neural Network) に入力してマルウェアを検出する手法を提案している．その論文中において，McLaughlin らによる 1 次元 CNN[14] を再現したものと，Yakura らによる 2 次元 CNN との比較実験が行われており，2 次元 CNN のほうが良い結果となっている．その理由について，1 次元 CNN ではコードセクションしか用いていないが，2 次元 CNN ではデータセクションも用いたためであると述べられている．

3. 提案手法

本章で提案するものは，データセットの良性ソフトウェアを汚染する手法である．まず，現実世界に既知のマルウェア検出手法が適用される際，どのようにサンプルが選択されてデータセットが作成されるのかについて論じる．次に，良性ソフトウェアの具体的な汚染手法について説明する．最後に，汚染された良性ソフトウェアを攻撃者が登録する方法について言及する．

3.1 サンプルの選択方法とデータセットの汚染

既存研究においては，データセットが先に存在し，そのデータセットを高精度で良性と悪性に分類可能な技術の開発にのみ注力している．本節では，攻撃者に亜種マルウェア分類手法が既知である場合に，その精度を落とす方法を攻撃者が考案可能であるかどうかに着目している．本節で提案するものは，攻撃者による良性データの汚染である．

マルウェアの場合，何らかの方法でマルウェアと判断され，集められたものが悪性データとしてデータセットに組み入れられる．ここで，悪性と良性の区別を，SVM や CNN などの一般的な機械学習アルゴリズムを用いて行うには，当然ながら良性ソフトウェアが必要となる．そして，その良性ソフトウェアのサンプルを，どこからどのように選択して利用するかが問題となる．

まず，良性ソフトウェアのサンプルを得る場所として一般的に考えられるのは，フリーソフトウェアが登録されているサイトである．店頭販売されているパッケージソフトウェアや，固有かつ正規のダウンロードサイトが決まっている商用ソフトウェアの場合，そこから入手する限りマルウェアである可能性はないため，これを機械学習のデータセットに加えることは不適切といえる．

次に，良性ソフトウェアのサンプルを選別する方法について考える．マルウェア検出手法によっては全サンプルを使うことも可能であり，サンプル数が多いほうが一般的に検出精度が高くなるため，理想的には全サンプルを使用することが望ましい．しかし，一部の手法では全サンプルを用いることは現実的に不可能である．Raff らが論文中で指摘しているように，n-gram を用いた手法では前処理に膨大な時間が掛かる．サンプル数を N とする場合，とくに N × N の処理が発生するような場合には，サンプル数を一定程度にする必要がある．このとき，良性ソフトウェアのサンプルを選別する方法に次のものが考えられる．

- 最も使用されているソフトウェアの上位から選択する
- 最新のソフトウェアの上位から選択する
- ランダムに選択する
- その他ファイルサイズなど

最も使用されているソフトウェアの上位から選択する方法は好ましいように見えるが，限定された用途のソフトウェアで，独特なコーディングをしている場合，そのような情報が機械学習に反映されなくなるという問題がある．マイナーなソフトウェアはマルウェアと誤分類されてもよいという判断ともとられ，批判は免れない．最新のソフトウェアの上位から選択する方法は好ましい方法のひとつである．新規的な技術が日々開発され，それらがソフトウェアに採用されていくのは事実である．新規のマルウェアも同様に新しい技術を採用する可能性があるため，未知のマルウェア検出という目的であればこの選択は適切といえる．ランダムな選択方式は，問題もなければ利点もない．強いていえば，古い技術でコーディングされたソフトウェアの特徴が，新しいマルウェアの検出に採用される一方，代わりに新しい技術でコーディングされたソフトウェアの一部が，サンプルとして選択されなくなってしまう点に問題がある．その他の方式は，データに何らかの偏りが生じるため危険である．例えば，ファイルサイズが小さいものを選択したほうが，n-gram の前処理にかかる時間が早くなる．一方で，大きいファイルサイズのソフトウェアが持つ特徴は無視されることになる．同様に，特定のバージョンの OS 用であったり，特定の言語で作成されているものであったり，様々な区分が考えられるが，それ以外のものが持つ特徴が機械学習から排除される点では同様であり，このような選択は適切とはいえない．

本論文では，最新のソフトウェアの上位から選択する方法で良性ソフトウェアのサンプルを得ており，マルウェアの検出手法が攻撃者に既知である場合に，攻撃者が良性ソフトウェアを汚染することでマルウェア検出にどの程度の影響があるかについて扱う．

3.2 良性ソフトウェアの汚染手法

現在，オープンソースソフトウェアとして様々なソフトウェアのコードが公開されている．本論文では，攻撃者がこれらのコードを利用してソフトウェアを作成することを想定している．とくに，ゲームの場合，ゲームの動作に必要なソースコードがオープンソースであれば，グラフィックとストーリーを変えれば大量に亜種を作成できる．また，グラフィックについても，フリー素材を提供しているサイトが多数存在している．つまり，類似の良性ソフトウェアを大量に短期間に生み出すことは，ある程度の労力を掛ければ不可能とはいえない．人数が複数いる大規模な犯罪組

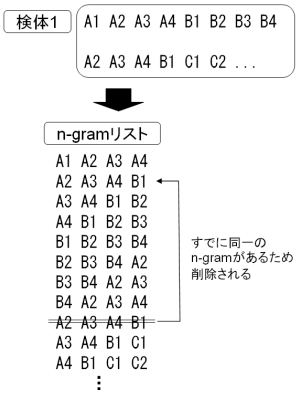

図 1 1 検体からの n-gram リストの作成
Fig. 1 Making procedure of n-gram lists from 1st sample.

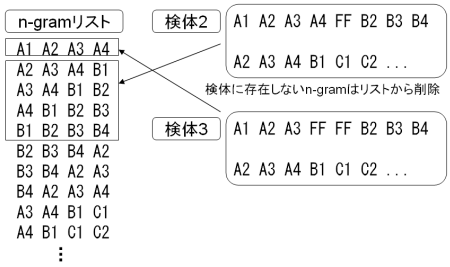

図 2 n-gram リストとすべての検体の比較
Fig. 2 Comparison procedure of n-gram lists with all samples.

織であれば，むしろたやすい．

次に，良性ソフトウェアの具体的な汚染手法について説明する．例えば，マルウェア検出手法に n-gram が使用されている場合，マルウェアに共通して含まれる n-gram を良性ソフトウェアが持てばよい．n-gram は文字列に適用するものであったため，ソフトウェアに適用する場合には様々な解釈がある．ここでは，ソフトウェア中の連続する n バイトを n-gram として扱うものを対象とする．通常，この n-gram はデータセクションではなくコードセクションのものである．マルウェア検出に n-gram を使用する手法には，データセクションやコードセクションの区別をせずにバイナリ全体を扱うものが存在する．これは，OS に依存せず，ソフトウェアエンジニアリングの知識も不要であるため，利点といえる．そこで，本手法では，自作の亜種マルウェアが共通して持つ n-gram を良性ソフトウェアに組み入れる．

本論文で行った手順は次のとおりである．まず，図 1 のように，同一亜種マルウェアファミリの 1 検体から n-gram リストを作成する．この 1 検体は任意のものでよいが，処理時間の都合上，最小ファイルサイズのものが望ましい．図の例は 4-gram であり，連続する 4 バイトの「A1 A2 A3 A4」,「A2 A3 A4 B1」が n-gram リストに登録されていく．このとき，すでに同一の 4-gram がこのリストに存在していれば，重複して登録はしない．

次に，図 2 のように，最初の 1 検体から作成した n-gram リストを，同一亜種マルウェアファミリの残りの検体と順次比較していく．図では，まず，n-gram リストと検体 2 を比較し，検体 2 に含まれない 4-gram を n-gram リストから削除している．次に，n-gram リストと検体 3 を比較し，同様に検体 3 に含まれない 4-gram を n-gram リストから削除している．これを繰り返すことで，すべての検体に共通する n-gram だけが n-gram リストに残る．

なお，この方法では，「すべての検体の 8 割以上に共通して存在する n-gram」や「良性の 2 割未満に存在し，なおかつ，悪性の 8 割以上に存在する n-gram」,「1 検体あたりの出現回数が 2 回以上の n-gram」のようなものを n-gram リストとして抽出することはできない．今回は処理速度を優先したが，様々な方法を検討する必要がある．

このようにして作成した n-gram リストの各 n-gram は，攻撃者が作成する良性ソフトウェアのデータとして組み入れられる．4-gram の場合は 32bit になるため，int 型変数の初期値として代入可能であるし，任意のバイナリ値を取れるものであれば文字列型でも構わない．難読化の意図も含めて，プログラムの動作に影響しない変数を用意することもできる．

3.3 汚染された良性ソフトウェアの登録

故意に汚染した良性ソフトウェアを，フリーウェアやシェアウェアなどとしてサイトに登録できるかどうかについて考える．

まず，特定のバイナリを含んでいるかどうかで登録を拒めるか検討する．4-gram の場合は 4 バイトとなるが，ある 4 バイトを含んでいるかどうかのみでマルウェアと判断することは難しい．偶然そのようなバイナリを含むソフトウェアである可能性が排除できないからである．なお，連続した特定の 100 バイトを含むようなものであれば，マルウェアと判断することは可能かもしれないが，3.3 節で述べたように，本手法では 4 バイトずつ分割して異なる変数に値を代入することも可能であるため，必ずしも連続した 100 バイトのようにはならない．

次に，静的解析やサンドボックスでの動的解析が挙げられる．本手法ではコードではなくデータとして n-gram を挿入しており，振る舞い検知では悪性とは判断されない．

ただし，特定の個人が短期間に類似のソフトウェアを大量に登録することには違和感がある．身分を詐称することは可能かもしれないが，ソフトウェアの類似性から嫌疑を掛けられ，人物を特定される可能性がある．一方，国家ぐるみのサイバーテロや，大規模な犯罪組織が主導するものであれば可能と考えられる．とくに，ソフトウェアの目的がエンターテイメントである場合，短期間に類似のソフトウェアが大量に登録されることは，アフィリエイト目的であれば不自然とはいえない．

4. 評価

4.1 評価方法

本節では，提案手法の評価方法について述べる．

亜種マルウェアに関しては，EMM 論文で用いたものと同じく，MWS データセット [15] の CCC Dataset 2012 および 2013 に含まれる 4 種のファミリのものを用いた．ファミリ名の表記は Microsoft のものとしている．良性ソフトウェアのサンプルは EMM 論文で使用したものと同一であり，Windows にプリインストールされている拡張子が exe のファイルおよび，Download.com Powered by Cnet からダウンロードされた実行ファイルである [16]．良性ソフトウェアのサンプル数は 1500 であり，選択はランダムである．

亜種マルウェアの表記は，Net-Worm.Win32.Allaple.a が a，Net-Worm.Win32.Allaple.b が b，Net-Worm.Win32.Allaple.e が e，Net-Worm.Win32.Kido.ih が ih である．以降，この略称で本文中において表記を省略するものとする．検体のサンプル数は EMM 論文に合わせて，a が 900，b が 1500，e が 1499，ih が 670 とした．e のサンプル数が EMM 論文よりひとつ少ないのは，EMM 論文において比較用検体がこの 1500 サンプルに含まれており，除外したためである．なお，CCC Dataset は 2013 以降更新されておらず，MWS データセットにはこれ以降の亜種マルウェアデータセットが存在していない．

本論文の評価においては，EMM 論文と同じく 10 分割交差検証を用いた．本評価においては，この検証を事実上のテストとして扱う．我々の n-gram 抽出手法を用いて，ある亜種マルウェアファミリに含まれるマルウェアを検出するには，同一ファミリの検体を捕獲してトレーニングを行い，未知の検体がこのファミリに含まれているかどうか評価する．つまり，テストデータは別のデータセットである必要はなく，保有しているデータセットの一部をテスト用としても，実際の環境と同様である．よって，10 分割交差検証を用いているが，これは検証ではなく，分割されたデータの内，トレーニングに使用されないものがテストに相当する．なお，検証用として分割されたデータは，トレーニング時のハイパーパラメータの調整には一切影響を与えていない．

4.2 機械学習による評価

CSS2019 論文で述べた我々の n-gram 抽出手法を用いて，本論文で提案する汚染手法がどの程度有効か評価を行った．

3.2 節で述べたように，本来はマルウェアのコード中で，変数の値として n-gram を代入する必要があるが，それにはマルウェアのソースコードが必要となる．そこで，本評価においては，n-gram 抽出済の良性サンプルの末尾に，図 2 の方法を使って作成した n-gram リストを追加した．

表 1 CNN による良性ソフトウェア汚染の評価結果 (ファミリ a)
Table 1 Evaluation results with benign software pollution by CNN (Family a).

i	TP	FP	TN	FN
0	90	0	150	0
1	90	0	150	0
2	90	0	150	0
3	90	1	149	0
4	90	0	150	0
5	90	0	150	0
6	90	0	150	0
7	90	0	150	0
8	90	0	150	0
9	90	0	150	0

我々の n-gram 抽出手法ではスキップされる箇所があるため，厳密に同一処理とはならないが，これをもって評価を行った．

良性ソフトウェアの汚染率に関しては，3.3 節の記述を参考にした．多くのソフトウェアがサイトに登録される中，怪しまれずに汚染を行うことは容易ではない．本評価においては，良性ソフトウェアのサンプル数は 1500 であるため，国家ぐるみの組織で類似のゲームソフトを作成したとしても 30 種類くらいが限界と予想される．これは，1 日平均 1 つ登録して 1 ヶ月で 30 個という想定である．評価結果には，汚染による影響が強く出たほうが考察しやすいため，この 30 個を上回る，5 %の 75 個を汚染するものとした．つまり，犯罪組織が最大限努力しても到達しない程度の汚染率で評価しているということである．

CSS2019 論文で述べた n-gram 抽出手法を用いて，機械学習によって評価を行った．なお，CSS2019 論文で述べた n-gram 抽出手法のとおりにトレーニングサンプルを用意すると，テストサンプルと分けるたびに n-gram 抽出からやり直さなければならなくなる．CSS2019 論文と同様の評価方法で，抽出後のサンプルに対して 10 分割交差検証を行っている．

実験は CSS2019 論文と同じ環境および同じ機械学習のネットワークで行った．a の結果を表 1 に，b の結果を表 2 に，e の結果を表 3 に，ih の結果を表 4 に示す．なお，表中の i は 10 分割交差検証の各回の番号，TP は True Positive，FP は False Positive，TN は True Negative，FN は False Negative を表す．また，マルウェアであるほうが Positive である．

5. 考察

本章では，4 章で行った評価の考察を行う．本論文で提案する汚染手法を，我々の亜種マルウェア抽出手法に適用した場合の結果については，表 1〜表 4 に示した．今回は 4-gram における評価しか行っていないが，EMM 論文では

表 2 CNN による良性ソフトウェア汚染の評価結果 (ファミリ b)
Table 2 Evaluation results with benign software pollution by CNN (Family b).

i	TP	FP	TN	FN
0	150	0	150	0
1	150	0	150	0
2	150	0	150	0
3	150	0	150	0
4	150	0	150	0
5	150	0	150	0
6	150	0	150	0
7	150	0	150	0
8	150	0	150	0
9	150	0	150	0

表 3 CNN による良性ソフトウェア汚染の評価結果 (ファミリ e)
Table 3 Evaluation results with benign software pollution by CNN (Family e).

i	TP	FP	TN	FN
0	150	0	150	0
1	150	0	150	0
2	150	1	149	0
3	150	0	150	0
4	150	0	150	0
5	150	0	150	0
6	150	0	150	0
7	150	0	150	0
8	150	0	150	0
9	149	0	150	0

表 4 CNN による良性ソフトウェア汚染の評価結果 (ファミリ ih)
Table 4 Evaluation results with benign software pollution by CNN (Family ih).

i	TP	FP	TN	FN
0	66	0	150	1
1	67	0	150	0
2	65	0	150	2
3	66	0	150	1
4	52	4	146	15
5	67	0	150	0
6	64	0	150	3
7	67	0	150	0
8	67	0	150	0
9	67	0	150	0

亜種マルウェア抽出手法の 4-gram 評価において誤分類はなかった．

参考のため，00 の値を持つバイトを追加した場合の評価を行った，CSS2019 論文の 4-gram 評価を再掲載する．a の結果を表 5 に，b の結果を表 6 に，e の結果を表 7 に，ih の結果を表 8 に示す．

2.2 節にて，ソフトウェア全体のバイナリ n-gram を使用したマルウェア検出手法の関連研究を挙げた．ここで挙げた研究のすべての手法においては，マルウェアが特徴とする n-gram をテスト検体が含むかどうかを分類根拠としている．そうであると，本論文で提案した汚染手法は，良性ソフトウェアがその n-gram を含んでしまうため，影響は大きいと考えられる．本論文では，n-gram をテスト検体が含むかどうかによる評価を行っておらず，どの程度の影響があるかについては不明である．また，同一亜種マルウェアファミリのすべての検体に共通して含まれる n-gram のみを用いたため，80 %や 90 %共通して含まれる n-gram は除外されてしまっている．CNN では，それらの n-gram が

表 5 バイト列追加の評価 (ファミリ a)
Table 5 Evaluation results of adding bytes (Family a).

i	TP	FP	TN	FN
0	90	0	150	0
1	90	0	150	0
2	90	0	150	0
3	90	0	150	0
4	90	0	150	0
5	90	0	150	0
6	90	0	150	0
7	90	0	150	0
8	90	0	150	0
9	90	0	150	0

表 6 バイト列追加の評価 (ファミリ b)
Table 6 Evaluation results of adding bytes (Family b).

i	TP	FP	TN	FN
0	150	0	150	0
1	150	0	150	0
2	150	0	150	0
3	150	0	150	0
4	150	0	150	0
5	150	0	150	0
6	150	0	150	0
7	150	0	150	0
8	150	0	150	0
9	150	0	150	0

表 7 バイト列追加の評価 (ファミリ e)
Table 7 Evaluation results of adding bytes (Family e).

i	TP	FP	TN	FN
0	150	0	150	0
1	150	0	150	0
2	150	0	150	0
3	150	0	150	0
4	150	0	150	0
5	150	0	150	0
6	150	0	150	0
7	150	0	150	0
8	150	1	149	0
9	149	0	150	0

表 8　バイト列追加の評価 (ファミリ ih)
Table 8　Evaluation results of adding bytes (Family ih).

i	TP	FP	TN	FN
0	67	2	148	0
1	67	0	150	0
2	67	0	150	0
3	67	0	150	0
4	67	0	150	0
5	67	0	150	0
6	67	0	150	0
7	67	0	150	0
8	66	0	150	1
9	67	1	149	0

分類に大きく依存する特徴になりうるため，それらも含めればさらに汚染による影響を強くできると考えられる．

次に，EMM 論文と CSS2019 論文の 4-gram 評価結果について考察する．EMM 論文では誤分類は存在しなかったが，表 7 より e に FP が 1 つ，表 8 より ih に FP が 3 つと FN が 1 つ存在していることが確認される．つまり，我々の n-gram 抽出手法も完全ではなく，汚染の影響を受けるということである．

そして，本論文の表 2 では誤分類が存在しなかった．これは，b においては n-gram の出現順序や共起度が適切に機械学習された可能性があることを示唆している．もしくは，b においては，80 %や 90 %共通して含まれる n-gram を使ってうまく分類できた可能性もある．どちらがより強い要因であるかについては，既存研究で用いられている，n-gram を含んでいるかどうかの手法と比較する必要がある．

本論文の表 1 では FP が 1 つ，表 3 でも FP が 1 つ存在した．良性ソフトウェアをマルウェアと誤分類しているものであり，マルウェアの特徴が良性ソフトウェアに付加されたために分類精度が下がったことがわかる．ただし，表 7 にも FP が 1 つ存在しているため，本論文で提案した汚染手法が著しく影響したとはいえない．

注目すべきは，本論文の表 4 である．FP が 4 つあるが，そのすべては i=4 の回のものである．また，FN が合計 22 個あるが，その内の 15 個が i=4 の回のものである．つまり，i=4 の回のみ極端にトレーニングがうまくいかなかったということになる．本評価は 10 分割交差検証であり，5 このときに偏りが生じ，i=4 の回のトレーニングサンプルに汚染された良性サンプルが多く混入した可能性はある．その場合，その亜種マルウェアに共通して含まれる n-gram を分類時の根拠として使えないので，分類がうまくいかなかったということになるが，その理由では i=4 以外の回ではうまく分類できたことが説明できない．5 もうひとつ考えられる理由が，ih は a，b，e に比べて，亜種マルウェアに 80 %や 90 %共通して含まれる n-gram が少ないということである．100 %共通するもの以外の n-gram が少ないと，汚染によりトレーニングがうまくいかなくなる可能性が高くなる．i=4 の回はその影響がトレーニングに強くでたのではないかと推測される．

本来，亜種マルウェアというものは，同一ファミリに完全に共通して存在するバイナリ列が少ないほうが，発見されにくくなると思われるが，表 4 の結果はそれを覆している点が興味深い．同一ファミリに完全に共通して存在するバイナリ列があっても，それを使って良性ソフトウェアを汚染できれば，検出率を下げられるということである．

6. まとめ

本論文では，攻撃者が良性ソフトウェアを汚染する可能性があるという発想に基づき，n-gram を用いたマルウェア検出手法に対する汚染手法を提案し，評価を行った．既存研究では，特定の n-gram を含んでいるかどうかを特徴としているため，この汚染による影響を受けやすい．これらの既存手法における評価が行えなかったのが残念である．一方，我々が提案した n-gram 抽出手法では，n-gram の出現順序と共起度が特徴となるため，本論文の汚染手法による影響を受けにくい．しかし，ih に対する結果では，FP が 4 個，FN が 22 個出現し，誤分類のなかった EMM 論文の結果と比べると，汚染による一定の影響があった．

考察により，同一ファミリに完全に共通して存在するバイナリ列があっても，それを使って良性ソフトウェアを汚染できれば検出率を下げられるという知見が得られたのはひとつの成果といえる．つまり，機械学習を情報セキュリティ分野に適用する際には，既知のデータをどれだけの精度で分類できたかどうかだけではなく，攻撃者がどのような攻撃を行う可能性があるのかについて，常に念頭に置く必要があるということである．

謝辞　本研究は JSPS 科研費 JP18K11248 の助成を受けたものです．また、本研究にはマルウェア対策のための研究用データセットである MWS データセットを使用しています．

参考文献

[1] Kephart, J. O., Sorkin, G. B., Arnold, W. C., Chess, D. M., Tesauro, G. J. and White, S. R.: Biologically Inspired Defenses Against Computer Viruses, *Proc. the 14th Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, pp.985–996 (1995).

[2] Abou-Assaleh, T., Cercone, N., Keselj, V. and Sweidan, R.: N-gram-based Detection of New Malicious Code, *Proc. the 28th Annual Intl. Computer Software and Applications Conf. (COMPSAC)*, Vol.2, pp.41–42 (2004).

[3] Reddy, D. K. S. and Pujari, A. K.: N-gram Analysis for Computer Virus Detection, *Journal in Computer Virology*, Springer, Vol.2, No.3, pp.231–239 (2006).

[4] Boujnouni, M. E., Jedra, M. and Zahid, N.: New Malware Detection Framework Based on N-grams and Sup-

port Vector Domain Description, *Proc. the 11th Intl. Conf. on Information Assurance and Security (IAS)*, pp.123–128 (2015).

[5] Kolter, J. Z. and Maloof, M. A.: Learning to Detect and Classify Malicious Executables in the Wild, *Journal of Machine Learning Research*, Vol.7, pp.2721–2744 (2006).

[6] Raff, E., Zak, R., Cox, R., Sylvester, J., Yacci, P., Ward, R., Tracy, A., McLean, M. and Nicholas, C.: An Investigation of Byte N-gram Features for Malware Classification, *Journal of Computer Virology and Hacking Techniques* (2016).

[7] Fuyong, Z. and Tiezhu, Z.: Malware Detection and Classification Based on n-grams Attribute Similarity, *Proc. 2017 IEEE Intl. Conf. on Computational Science and Engineering (CSE) and IEEE Intl. Conf. on Embedded and Ubiquitous Computing (EUC)*, pp.793–796 (2017).

[8] Raff, E. and Nicholas, C.: Hash-Grams: Faster N-Gram Features for Classification and Malware Detection, *Proc. ACM Symposium on Document Engineering (DocEng) 2018* (2018).

[9] Kita, K. and Uda, R.: Malware Subspecies Detection Method by Suffix Arrays and Machine Learning, *Proc. the 55th Annual Conf. on Information Sciences and Systems (CISS)* (2021).

[10] 瀧口翔貴, 宇田隆哉: N-gram 圧縮と深層学習を用いたマルウェア分類手法の提案, 電子情報通信学会技術研究報告, Vol.118, No.494, EMM2018-112, pp.111–116 (2019).

[11] 宇田隆哉: N-gram 抽出法による亜種マルウェアの検出と攻撃耐性の考察, 情報処理学会コンピュータセキュリティシンポジウム 2019, 2A4-3 (2019).

[12] O'Kane, P., Sezer, S. and McLaughlin, K.: N-gram Density Based Malware Detection, *Proc. the World Symposium on Computer Applications & Research (WSCAR)* (2014).

[13] Yakura, H., Shinozaki, S., Nishimura, R., Oyama, Y. and Sakuma, J.: Neural Malware Analysis with Attention Mechanism, *Computers & Security*, Vol.87 (2019).

[14] McLaughlin, N., del Rincon, J. M., Kang, B., Yerima, S., Miller, P., Sezer, S., Safaei, Y., Trickel, E., Zhao, Z., Doupe, A. and Ahn, G. J.: Deep Android Malware Detection, *Proc. the 7th ACM Conf. on Data and Application Security and Privacy (CODASPY)*, pp.301–308 (2017).

[15] 高田雄太, 寺田真敏, 村上純一, 笠間貴弘, 吉岡克成, 畑田充弘: マルウェア対策のための研究用データセット〜MWS Datasets 2016〜, 情報処理学会研究報告, Vol.2016-CSEC-74, No.17, pp.1–8 (2016).

[16] "Windows PC Software - Free Downloads and Reviews", `http://download.cnet.com/windows/` (2017 年 8 月参照)