Computer Security Symposium 2019
21 - 24 October 2019

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

宇田 隆哉[1,a)]

概要：本論文では，N-gram を利用した抽出法で亜種マルウェアを検出する研究について，想定される攻撃と対応策について述べる．以前の論文で，N-gram を利用して亜種マルウェアから特徴を抽出し，畳み込みニューラルネットワークを用いた深層学習を利用することでバイナリコードの異なる亜種を検出できることを述べた．しかし，それは既存のマルウェアに対して検出可能であることを示したのみであり，この検出手法が攻撃者に既知であった場合に，攻撃を無効化できるかどうかについては検討していなかった．そこで，本論文では，この手法に対して想定される攻撃を列挙し，攻撃が可能かどうか考察する．さらに，攻撃が可能である場合に，その攻撃を無効にする改善方法を提案する．

キーワード：マルウェア検出，機械学習，深層学習

Detection of Malware Species by N-gram Extraction Method and Consideration of Attack Tolerance

Ryuya Uda[1,a)]

Abstract: N-gram extraction method for the detection of malware species and its attack tolerance are described in this paper. The N-gram extraction method for malware species was proposed in the past paper by my laboratory. Malware species with different binary codes were able to be detected by the method with deep learning of convolutional neural network. However, it was only shown that existing malware species could be detected and it was not considered whether it was still effective when the detection method was known to attackers. Therefore, in this paper, I assume attacks against the method and consider the availability of the attacks. Moreover, I also propose improved methods to disable the attacks.

Keywords: Malware Detection, Machine Learning, Deep Learning

1. はじめに

マルウェアの一種に亜種マルウェアというものが存在する．亜種マルウェアが登場する以前，マルウェアというものは特定のバイナリ列で構成されており，このバイナリ列が一度リストに登録されれば，パターンマッチングによりこれを発見するのは容易であった．これに対して，亜種マルウェアは亜種ごとにバイナリ列が異なる．例えば，感染して自分自身を複製するたびにプログラムの一部を変更し，異なるバイナリ列を作成するものが挙げられる．また，一部が異なるプログラムとなるように大量にプログラムを作成し，異なる攻撃先に送信する方法も挙げられる．

パターンマッチング方式は未知のマルウェアに対して無力である一方，既知のマルウェアは誤って見逃すことがなく，完全に有効である．しかし，亜種マルウェアの場合，同様の動作をするものであってもバイナリ列が異なってしまうため，パターンマッチング方式は適用できない．このような亜種マルウェアに対しては，同様の動きをする命令を含んでいるかどうかを解析したり，仮想環境上に構築されたサンドボックス上で動作させて振る舞いを見たりする

[1] 東京工科大学
Tokyo University of Technology
[a)] uda@stf.teu.ac.jp

方法があるが，いずれにの場合においても瞬間的に判断することは困難である．

そこで，著者の研究室において，機械学習を使って亜種マルウェアを検知する研究を行い，論文として発表した [1]．以下，本論文中ではこの論文を EMM 論文と呼称する．亜種マルウェアは，主となる命令群の途中に関係のない命令やデータなどを挿入することで作成されるが，主となる命令群自体は同様のものが存在する．もちろん，参照するメモリ上のアドレスも変化するため，完全に同一のコードとはならないが，同一の振る舞いであれば基本的なコードは同一となる．そこで，EMM 論文では，最近の深層学習で画像の判別によく利用される，畳み込みニューラルネットワーク (以下 CNN) を用いた．これは，画像の特徴を捉えて，表示位置や大きさが異なっていても，畳み込みにより画像を分類できることによる．つまり，亜種マルウェアの構造もこれと類似しているため，効果的に分類できると考えたためである．

亜種マルウェアに CNN を適用するにあたり，一番の問題はファイルサイズであった．亜種マルウェアがどのような形態のマルウェアかにも依存するが，ワームである場合には，大きい分には任意のファイルサイズの亜種マルウェアを自由に作成できる．CNN を効果的に使用するには，ある程度のサンプル数を訓練に使用する必要があるが，大きいマルウェアの場合には訓練時にコンピュータのメモリに入りきらないか，入るとしても非常に訓練に時間がかかってしまう．そこで著者らは，EMM 論文において，亜種マルウェアを圧縮して，効率よく CNN で訓練を行う方法を提案した．

EMM 論文では，亜種マルウェアが高い精度で分類可能であることが示されたが，著者らの研究のほうが実験に使用された亜種マルウェアが作成された日時よりも後で行われているため，亜種マルウェアの作成者にとって作成時点では著者らの分類方法が未知であることになる．そこで，本論文では，亜種マルウェアの検出手法が攻撃者に既知であることを想定し，著者らの手法に有効な攻撃が行えるかどうかの考察を行う．さらに，攻撃が有効である場合に，その攻撃を無効にする追加の手法を提案する．

なお，EMM 論文の提案手法の説明に，一部致命的な誤りがあり，解説した手法と実際の手法が異なっていた．この誤りについても本論文中で説明する．

2. 関連研究

2.1 CNN と注意機構による画像化されたマルウェアの解析手法

矢倉らは，CNN に注意機構を組み合わせることによって，亜種マルウェアに含まれるバイナリ列の中で，そのファミリに特有の領域を検出し，人手による解析を効率化する手法を提案している [2]．彼らの評価実験によると，彼らの手法は GIST 特徴量に基づく k-近傍法によるマルウェア分類手法よりも高い分類精度を示すが，それでも 50%となっている．彼らも，CNN に入力する前にマルウェアを圧縮しなければならないことは理解しており，彼らの手法では，マルウェアを画像にして縦方向に圧縮し，一般的な画像の分類を行っている．しかし，マルウェアのコードをそのまま画像にしているため，縦方向に画像を圧縮するとマルウェアの命令そのものが他の値と合算され，特徴が消えてしまう．評価実験において，マルウェアのデータサイズが大きいと分類精度が著しく低下していることから，この分類手法が攻撃者に既知である場合，故意に大きなサイズのワームを作成されると検出を回避されてしまう．

2.2 Paragraph Vector を用いたマルウェアの亜種推定

佐藤らは，マルウェアから抽出した API コール群を Paragraph Vector によりベクトル化し，機械学習を用いてマルウェアの亜種を推定する研究を行っている [3]．彼らの実験では，21 種類のマルウェアを，教師あり学習器の Support Vector Machine(SVM) または Random Forest(RF) を用いることで，ファミリごとに自動分類している．この手法の分類精度はファミリによって異なるが，SVM を用いた場合に平均 81.69%，最大 96.43%であった．

分類精度の平均が 81.69%であるということは，平均的に 5 個に 1 個は亜種マルウェアを見逃す可能性があるということも問題であるが，この分類手法が攻撃者に既知である場合，動作することのないデッドコードとして無関係の API コールをランダムに大量に挿入された場合に，ベクトル化したデータが機械学習時にメモリに入りきらなくなるか，大量の無関係の API コールによって分類精度がさらに低下する可能性がある．

2.3 ランサムウェアの暗号化処理について

重田らは，ランサムウェアによる暗号化処理に着目したランサムウェアの検知手法を提案している [4]．彼らは，あるランサムウェアが CryptoAPI や OpenSSL を利用していることに着目し，CryptoAPI や OpenSSL の利用を検知したプロセスのうち，ホワイトリストにないものをマルウェアと判断している．

まず，ランサムウェアがワームである場合，ファイルサイズは任意に拡大できるため，CryptoAPI も OpenSSL も使用せずに，これらを独自実装したランサムウェアを作成することが可能である．彼らの手法が攻撃者に既知であれば，攻撃者がこの方法を採用することが想定される．また，ランサムウェアが狭義のウイルスである場合，良性のファイルからなるプロセスの一部としてランサムウェアが動作する．この場合，このランサムウェアのプロセスはホワイトリストに含まれてしまうため，検知対象から除外されてしまう．

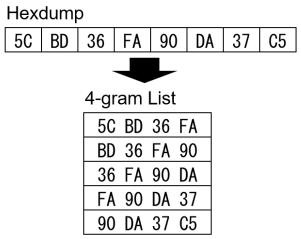

図 1　N-gram リストの作成方法
Fig. 1　Creation of N-gram list.

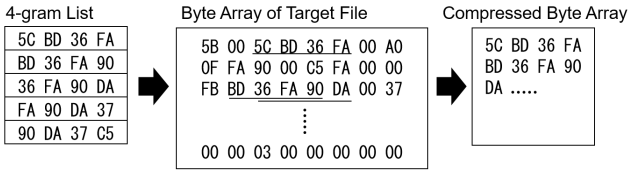

図 2　N-gram リストによる目標ファイルの圧縮方法
Fig. 2　Compression method for target files by N-gram list.

3. 提案手法

EMM 論文において，解説した提案手法と評価に使用した実際のプログラムとの間に差異があったことに気づいたため，この場で訂正するとともに説明したい．

図 1 に N-gram リストの作成方法を示す．

これは EMM 論文でも解説したものであるが，N-gram リストの作成方法に訂正はない．図に示すように，例えば 4-gram であれば，4 バイトずつの組をリストに登録していく．このとき，図のように 1 バイトずつずらしながらリストを作成するため，リストの次の項目に登録されるバイト列の先頭の 3 バイトは，その項目の最後の 3 バイトと同一となる．

次に，EMM 論文で解説した圧縮方法について説明する．図 2 に N-gram リストによる，目標ファイルの圧縮方法を示す．

図のように，作成した N-gram リストにあるいずれか 1 つの項目と，目標ファイル中のバイト列が一致した場合には，そのバイト列の位置にマークを付け，最終的にマークされなかった部分を削除することで圧縮を行うと EMM 論文で述べた．しかし，プログラムを再度確認したところ，実際の実装はこの通りに行われていなかった．図中の Byte Array of Target File において，3 箇所に下線が付けられている．このうち，「5C BD 36 FA」と「BD 36 FA 90」の箇所にマークを付けるというのは正しいが，先に「BD 36 FA 90」の箇所にマークが付けられていた場合，実際の実装においては「36 FA 90 DA」にはマークが付けられない．実際の実装においてマークを付けていた箇所は，その時点でどのバイトにもマークが付けられていない箇所のみであった．例えば，16-gram 圧縮の場合，目標ファイルの中で，作成した 16-gram リストのある項目と一致する箇所を発見した場合，その 16 バイトすべてが未マークの状態でなければ，その 16 バイトのうちどのバイトにもマークを付けていなかった．

EMM 論文で述べた提案手法に合致するように，N-gram 圧縮のプログラムを正しく修正したところ，圧縮率が悪く，ファイルサイズが大きすぎて CNN に入力して訓練を行うことができなかった．そこで，EMM 論文にて実際に評価を行ったほうの圧縮方法を，本論文では N-gram 抽出 (N-gram Extraction) と呼称することにし，この方法にて評価を行うこととする．亜種マルウェアの検出率という点では，この N-gram 抽出手法でも関連研究よりも分類精度が勝っているためである．なお，もとの N-gram 圧縮手法は著者らの機械学習の実装環境では動作させることはできないが，実装を変えれば動作させることは理論上は可能であり，N-gram 抽出手法で抽出されるファイルに含まれる情報は N-gram 圧縮手法で圧縮されるファイルにも必ず含まれるため，N-gram 圧縮手法の分類精度が N-gram 抽出手法に劣ることは考えられない．ただし，ハイパーパラメータのチューニングを行う計算回数を同一にした場合，著しく計算時間が長くなり，現実的ではない．

4. 提案手法の攻撃耐性

本章では，EMM 論文では論じなかった攻撃耐性とその対応策について述べる．

4.1 ランダムな値を持つバイナリデータの付加

亜種マルウェアを作成する際に，ランダムな値を持つバイナリデータを付加する方法について考える．亜種マルウェアは，そもそも何らかのバイナリデータを付加することで作成されており，パターンマッチングによる検出を避けるためそれは一般的にランダムな値である．しかし，ランダムな値は，提案手法の N-gram 抽出によって除去されるため，この攻撃は無意味である．

4.2 特定の値を持つバイナリデータの付加

亜種マルウェアを作成する際に，特定の値を持つバイナリデータを付加する方法について考える．同一のバイナリ列をすべての亜種マルウェアに付加することは可能であるが，これは単純にパターンマッチングにより検出される可能性を高めるだけで無意味である．また，提案手法においても，N-gram 抽出時に付加されたバイナリが残り，CNN において亜種マルウェアであることを判断する要素の 1 つに使われるため，検出率が上がるだけである．

4.3 任意の良性ファイルと同一のバイナリデータの付加

任意の良性ファイルから一部のバイナリコードを抜き出し，マルウェアの内部に挿入する方法について論じる．図 3 に示すように，良性ファイルを 1 つ選び，ランダムに選

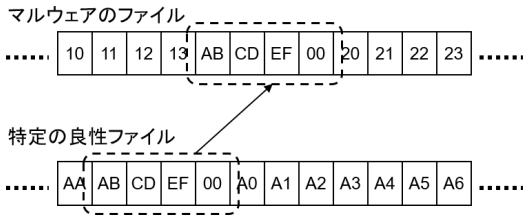

図 3　任意の良性ファイルと同一のバイナリデータの付加方法
Fig. 3　Method for adding the same binary data as those in each benign file.

択された箇所から特定のバイト数をマルウェア内部に挿入するのが最も単純な方法と推測される．

このとき，連続して何バイト選択するかについては，4-gram の抽出に対応するのであれば最低 4 バイト，8-gram の抽出であれば最低 8 バイト，16-gram の抽出であれば最低 16 バイトが必要となる．1 箇所のみ選択して複写し，挿入しても意味はなく，マルウェアを良性ファイルに偽装するためにはある程度のバイト数を選択して挿入する必要がある．このとき，4-gram，8-gram，16-gram の抽出すべてに対応しようと考えれば，16 バイトずつ複数箇所から選択して複写し，挿入するのが効率的である．

任意の良性ファイルからバイナリデータを選択して亜種マルウェアに挿入する方法として，次の 2 通りが考えられる．1 つは，良性ファイルを 1 つ，もしくは 2〜10 などの少ない数選択し，対象としたすべてのファイルの同一箇所からすべての亜種マルウェアにバイナリを複写，挿入する方法である．

この場合，亜種マルウェア自体は，どの個体でもマルウェアとして分類されることになる．なぜなら，複写して挿入されたバイナリは，その亜種マルウェアのすべての個体に共通して含まれており，一方で，良性ファイルの非常に少ない個体数にしかそのバイナリは存在しないからである．なお，機械学習の訓練において，訓練がうまく行われれば，この複写して挿入されたバイナリは，マルウェアの特徴としては無視されて使用されなくなり，それ以外のバイナリからマルウェアかどうかの判断が行われるようになるため，機械学習においては必ずしも致命的な攻撃とはいえない．

仮に，バイナリの複写元がとても有名なソフトウェアで，多くの PC，タブレット，スマートフォンなどで使用されているものであった場合，その良性ファイルは，機械学習においてマルウェアに分類されてしまう可能性があるため，ハッシュ値を別途確認し，ホワイトリストに登録しておく必要がある．既知の良性ソフトウェアでなければ，マルウェア偽装の対象として使用できないため，未知のソフトウェアに対応する必要はない．ただし，この対策を怠ると，特定の有名ソフトウェアをマルウェアとして誤検知させる攻撃が可能となり，サイバーテロなどに使用される恐れもある．機械学習の訓練がうまく行われれば，この攻撃は無効となる可能性については先述したが，必ずしもそうなる保証はないため，ホワイトリストの使用は必須といえる．

なお，使用する良性ファイルが 1 つであっても，亜種マルウェアごとに任意の箇所からバイナリを選択してしまうと，N-gram 抽出時に同一バイナリでないものは削除されてしまうため，意味がなくなる．

2 つ目の方法として，亜種マルウェアごとに使用する良性ファイルを変え，任意の位置からバイナリを複写してマルウェアに挿入するものが考えられる．しかし，この方法では，N-gram 抽出時に同一バイナリでないものは削除されてしまうため意味がない．機械学習による訓練が完了し，フィルタが作成された時刻より後に，それまでに使用していなかった良性ファイルから任意のバイナリデータを複写して挿入した亜種マルウェアを新たに作成することも無意味である．これも，既存の亜種マルウェアによる N-gram 抽出時に，同一バイナリでないものは削除されてしまうからである．

4.4　良性ファイルに多く含まれるバイナリデータの付加

本節では，良性ファイルに多く含まれるバイナリデータを亜種マルウェアに付加することで，機械学習を騙せるかどうかについて考察する．良性ファイルに多く含まれるバイナリデータであるが，変数の初期値として一般的に考えらる値が 0 か最大値であるため，1 バイトの値が 16 進表記で 00 もしくは FF となるものに注目した．まず，この 00 もしくは FF が，良性ファイルおよび亜種マルウェアにどの程度含まれているか調べた．

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 データセットの CCC Dataset 2012 および 2013 に含まれる 4 種のファミリのものである．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 [7].

表 1 に，1 バイトの値が 16 進表記で 00 となるバイトが含まれる場合の評価を，表 2 に，1 バイトの値が 16 進表記で FF となるバイトが含まれる場合の評価を示す．表中の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，本文中でも同様に表記を省略する．Ave は平均値，STD は標準偏差，Original は元のファイルのバイト数，00 は 1 バイトの値が 16 進表記で 00 であったバイト数，FF は 1 バイトの値が 16 進表記で FF であったバイト数，Rate は元のファイルのバイト数に対する割合を示している．

表 1　00 が含まれるバイト数と割合
Table 1　Number of bytes and rate of 00.

Name of Data	Original	00	Rate
Ave Benign	549509.4	68600.4	0.125
STD Benign	990533.9	159357.8	0.161
Ave Family a	68424.8	1649.5	0.024
STD Family a	14156.5	167.1	0.012
Ave Family b	60192.1	1611.9	0.027
STD Family b	2740.5	170.1	0.062
Ave Family e	87533.3	3442.0	0.039
STD Family e	13784.3	5540.1	0.402
Ave Family ih	200543.5	17165.1	0.086
STD Family ih	200423.5	92150.1	0.460

表 2　FF が含まれるバイト数と割合
Table 2　Number of bytes and rate of FF.

Name of Data	Original	FF	Rate
Ave Benign	549509.4	20947.4	0.038
STD Benign	990533.9	70025.2	0.071
Ave Family a	68424.8	256.6	0.004
STD Family a	14156.5	66.0	0.005
Ave Family b	60192.1	236.6	0.004
STD Family b	2740.5	23.0	0.008
Ave Family e	87533.3	327.7	0.004
STD Family e	13784.3	47.5	0.003
Ave Family ih	200543.5	5291.0	0.026
STD Family ih	200423.5	7401.4	0.037

表 3　バイト列追加の評価 (4-gram ファミリ a)
Table 3　Evaluation of adding bytes (4-gram 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 4　バイト列追加の評価 (4-gram ファミリ b)
Table 4　Evaluation of adding bytes (4-gram 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

検体のサンプル数は EMM 論文と同じく，a が 900，b が 1500，e が 1499，ih が 670 である．EMM 論文と比べて e の検体数が 1 つ少ない理由は，N-gram 抽出において，比較用の検体と同一の検体が評価用の検体の中に含まれていたことに気づいたためであり，今回はこれを除去している．

表 1 および表 2 より，Rate の値を見ると，良性ファイルおよび亜種マルウェアともに，00 の値を持つバイトのほうが FF のものよりも多く含まれていることがわかった．もちろん，これ以外の値のバイトをより多く含む可能性は否定できないが，プログラムの仕組み上，常識的に考えてありえそうもないため，本論文では 00 の値を持つバイトを亜種マルウェアに追加することで偽装を行った．

00 の値を持つバイトを追加するルールは次のとおりとした．表 1 より，良性ファイルのほうが 00 の値を持つバイト数が多いため，N-gram 圧縮された亜種マルウェアの 16 バイトごとに 1/10 の確率で 16 バイト連続した 00 の値を持つバイト列を挿入した．これにより，亜種マルウェアのファイルサイズは平均 1 割増加することになる．なお，正確に実験を行うには，N-gram 圧縮前の亜種マルウェアにデータを付加する必要があり，付加するデータの間隔や連続するバイト数も良性ソフトウェアのものに合わせる必要がある．さらに，表 1 と表 2 を見ると，良性ソフトウェアの標準偏差の値が平均値を超えていることから，各良性ソフトウェアにおけるデータの分布を 1 つずつ模倣して亜種マルウェアにデータを挿入する必要がある．しかも，実際の亜種マルウェアにデータを挿入する場合，コードの都合により任意の箇所に任意のバイト数を挿入できるわけではない．今回の実験では，良性ファイルに多く含まれるバイナリデータの付加することで，最大でどの程度の影響を与えられるかではなく，影響があるかどうかを調べるため，実験を簡略化した．

実験は EMM 論文と同じ環境で行った．ただし，バージョンは Python 3.6.8, NumPy 1.16.1, TensorFlow 1.12.0, Keras 2.2.4，mtcnn 0.0.8 である．4-gram を用いた a の結果を表 3 に，b の結果を表 4 に，e の結果を表 5 に，ih の結果を表 6 に，8-gram を用いた a の結果を表 7 に，b の結果を表 8 に，e の結果を表 9 に，ih の結果を表 10 に，16-gram を用いた a の結果を表 11 に，b の結果を表 12 に，e の結果を表 13 に，ih の結果を表 14 に示す．EMM 論文と同じく 10-分割交差検証を用いているが，これは検証ではなく事実上のテストである．エポック数は 3 で固定であり，検証用のサンプルはハイパーパラメータの決定に関与していない．訓練におけるステップ数は Keras により決定されており，a が 1728，b が 2160，e が 2159，ih が 1562 である．表中の TP は True Positive，FP は False Positive，TN は True Negative，FN は False Negative を表す．

表 5　バイト列追加の評価 (4-gram ファミリ e)
Table 5　Evaluation of adding bytes (4-gram 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 6　バイト列追加の評価 (4-gram ファミリ ih)
Table 6　Evaluation of adding bytes (4-gram 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

表 7　バイト列追加の評価 (8-gram ファミリ a)
Table 7　Evaluation of adding bytes (8-gram 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 8　バイト列追加の評価 (8-gram ファミリ b)
Table 8　Evaluation of adding bytes (8-gram 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 9　バイト列追加の評価 (8-gram ファミリ e)
Table 9　Evaluation of adding bytes (8-gram 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	0	150	0
9	149	0	150	0

表 10　バイト列追加の評価 (8-gram ファミリ ih)
Table 10　Evaluation of adding bytes (8-gram Family ih).

i	TP	FP	TN	FN
0	67	0	150	0
1	66	0	150	1
2	66	0	150	1
3	67	0	150	0
4	67	0	150	0
5	67	3	147	0
6	67	0	150	0
7	67	0	150	0
8	67	0	150	0
9	67	0	150	0

表 3〜表 14 より，a と b では誤分類は起きていない．4-gram の ih に FP が 3 つ，FN が 1 つ存在するが，これは EMM 論文では存在しなかったものである．8-gram の ih にも FN が新たに 2 つ出現している．以上より，影響は皆無ではないことがわかる．なお，訂正したデータの掲載は割愛するが，EMM 論文の Accuracy，Precision，Recall，F 値には誤りがあることを記しておく．

本節で述べた攻撃方法には簡単に対応可能である．値が 00 となるバイトを，N-gram 抽出を行う前に良性ファイルおよび亜種マルウェアの両方から削除してしまえばよい．もちろん，未知のテストサンプルとなる検体からも同様にこの値のバイトを削除する．この方法により，この攻撃を無効化できる．

4.5　Adversarial Examples の付加

亜種マルウェアに Adversarial Examples を付加する方法について考える．CNN を用いた機械学習を騙す目的に Adversarial Examples が有効であることはよく知られており，著者らの論文においても，視覚的影響が少ない Adversarial Examples を CAPTCHA 画像に重畳する研究

表 11　バイト列追加の評価 (16-gram ファミリ a)
Table 11　Evaluation of adding bytes (16-gram 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 12　バイト列追加の評価 (16-gram ファミリ b)
Table 12　Evaluation of adding bytes (16-gram 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 13　バイト列追加の評価 (16-gram ファミリ e)
Table 13　Evaluation of adding bytes (16-gram Family e).

i	TP	FP	TN	FN
0	149	1	149	1
1	150	0	150	0
2	150	0	150	0
3	150	0	150	0
4	150	1	149	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

について述べた [5][6]．CNN は画像の分類によく用いられているが，画像においては任意のバイナリ列を付加してしまうと，視覚的影響が大きくなる．そこで，Adversarial Examples に小さな係数を乗算したしたものを，オリジナル画像に重畳して使用する．一方，マルウェアにおいては，Adversarial Examples にどれほど小さな係数を乗算しようと，そのバイナリ列をオリジナルのマルウェアに重畳してしまうと，バイナリコードが破壊され，動作しなくなってしまう．しかし，マルウェアは画像とは異なり，プログラムの一部に動作とは無関係のコードが挿入されていても動作

表 14　バイト列追加の評価 (16-gram ファミリ ih)
Table 14　Evaluation of adding bytes (16-gram Family ih).

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

には支障がなく，視覚的影響も関係ないため，Adversarial Examples を単純にマルウェアの一部に連接することは可能である．

まず，Adversarial Examples を単純に作成して使用する方法について考察する．これは，すべての亜種マルウェアに共通の Adversarial Examples を連接する方法と，個別のものを連接する方法とに分類される．前者の場合，N-gram 抽出への耐性があり，抽出後のバイナリに Adversarial Examples が残るが，これ自体がすべての亜種マルウェアに共通のバイナリ列になってしまい，パターンマッチングで容易に検出されてしまう．なお，Adversarial Examples を含んだサンプルが訓練に使用された場合，そのモデルはその Adversarial Examples に高い耐性を持つことも知られており，効果的な攻撃とはいえない．後者の場合，亜種マルウェアごとに付加される Adversarial Examples のバイナリが異なるため，N-gram 抽出によってこれらは除去され，無意味である．

次に，提案手法および実装方法が攻撃者に既知である場合に，Adversarial Examples が攻撃に有効であるか考察する．攻撃者は，ある程度の数の亜種マルウェアを先行して配布し，この亜種マルウェアを検出するフィルタを作成させた後に，これを騙す亜種マルウェアの作成を試みることが可能である．フィルタ作成後に，新たに作成する亜種マルウェアに任意の Adversarial Examples を連接した場合，これは N-gram 抽出によって除去されてしまうため，無意味である．しかし，攻撃者は，ある程度の数の亜種マルウェアを先行して配布する段階で，この中でファイルサイズが一番小さい個体のみに Adversarial Examples を連接しておくことが可能である．図 2 を用いて説明すると，図中の 4-gram List の中に Adversarial Examples は含まれることになり，機械学習による訓練済みモデルが作成された後に，図中の Byte Array of Target File として，Adversarial Examples を含む亜種マルウェアの検体が選択されることになるため，この攻撃は有効である．Adversarial Examples が機能していれば，理論上，新たに作成された亜種マル

ウェアは，提案手法により検出されることを完全に回避できる．

この攻撃に対する対策について検討する．まず，Adversarial Examples に耐性を持たせる手法について説明する．図 2 を用いて説明すると，図中の Compressed Bite Array（実際の提案手法では，3 章にて説明した抽出になる）に，別途作成した Adversarial Examples を連接しておくというのが 1 つの方法である．Adversarial Examples の作成方法は，攻撃者だけではなく防御者側にも既知であるため，それまでに収集された亜種マルウェアの検体から Adversarial Examples を作成することは可能である．作成する際に使用するモデルやネットワークが異なっても，Adversarial Examples への耐性をある程度有することも知られているため，完全な対応策ではないが，有効であるとはいえる．

次に，N-gram 圧縮の際に，比較に使用する検体を変更し，複数のフィルタを作成する方法について説明する．図 1 に示す Hexdump は，あるファミリにおいて，収集された亜種マルウェアの検体から一番小さいものを選択して作成している．よって，攻撃者は，この Hexdump に使用される検体を事前に指定することが可能である．そこで，この Hexdump に使用される検体の選択を，一番小さいものではなく，任意のものに変更する．亜種マルウェアを検出するフィルタの作成には，1 つのマルウェアファミリにつき約 700〜1500 個の検体が使用されているが，任意のものから選択すれば，攻撃者が Adversarial Examples を，この Hexdump に使用される検体に含ませられる確率は約 700〜1500 分の 1 になる．当然，攻撃者は複数の検体に Adversarial Examples を連接してくることが可能であるが，このフィルタを 1 つのマルウェアファミリにつき複数個作成し，論理和をとって亜種マルウェアを検出するようにすればよい．

5. まとめ

本論文では，EMM 論文にて提案した，N-gram 抽出を用いた亜種マルウェアの検出手法が，攻撃者に既知である場合にも有効であるかどうかの考察を行った．結果として，いくつかの攻撃方法は提案手法に対して有効であり，その攻撃方法は提案手法以外のパターンマッチングやその他の機械学習によるマルウェア検出方式にも有効であることが確認された．一方，これらの攻撃を無効化する対策方法についても言及し，適切に対策を行えば提案手法の有効性に変わりはないことも示した．

現在，情報セキュリティに関する様々な攻撃に対して，機械学習を使用して防御する研究が行われているが，既知の攻撃サンプルを用いて評価を行うのみで，防御方法が攻撃者に既知である場合の考察が行われていないものが散見される．実運用の際には，事前に十分な検討が必要であることが，本論文によって示された．

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] 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.
[2] 矢倉大夢, 篠崎慎之介, 西村礼恩, 大山恵弘, 佐久間淳: CNN と注意機構による画像化されたマルウェアの解析手法, 情報処理学会コンピュータセキュリティシンポジウム 2017 論文集, No.2, pp.1381–1388, 2017.
[3] 佐藤拓未, 後藤滋樹, 武部嵩礼: Paragraph Vector を用いたマルウェアの亜種推定法, 情報処理学会コンピュータセキュリティシンポジウム 2016 論文集, No.2, pp.298–304, 2016.
[4] 重田貴成, 森井昌克, 長谷川智久, 池上雅人, 石川堤一: 暗号化処理に着目したランサムウェア検知, 情報処理学会研究報告, Vol.2017-SPT-22, No.30, pp.1–5, 2017.
[5] 阿座上知香, 柴田千尋, 宇田隆哉: Adversarial CAPTCHA : 畳込みニューラルネットワークに耐性のある CAPTCHA の提案と評価, 情報処理学会論文誌, Vol.60, No.2, pp.680–695, 2019.
[6] T. Azakami, C. Shibata, R. Uda and T. Kinoshita: Creation of Adversarial Examples with Keeping High Visual Performance, Proc. of IEEE 2nd International Conference on Information and Computer Technologies (ICICT), 2019.
[7] “Windows PC Software - Free Downloads and Reviews”. http://download.cnet.com/windows/ (2017 年 8 月参照)