Vol.2020-DPS-182 No.51
Vol.2020-CSEC-88 No.51
2020/3/13

機械学習を悪用する未知のマクロマルウェアの脅威

山本 理紗[1]　三村　守[1]

概要：近年，機械学習を用いた未知のマルウェアの検知手法が研究されている．機械学習ではマルウェアの特徴を抽出し，この特徴によって良性，悪性を分類する．先行研究では，ソースコードの単語を特徴とし，機械学習を用いてマクロマルウェアの分類する手法を提案した．この手法では，言語モデルとして Latent Semantic Indexing（LSI）を使用している．一方で，Android マルウェア等に良性の特徴を付与することによって，機械学習を用いた検知手法を回避されることが検証されており，マクロマルウェアについても同様の課題がある可能性がある．そこで本研究では，マクロマルウェアに対し特徴の変更を行い，マルウェアの機能を維持したまま検知を回避することが可能であるかを検証する．評価対象としては，Bag of Words（BoW）および LSI を用いた検知手法を用いた．これらの検知手法に対し，変数を良性データセットでの出現頻度の高い単語に置換する手法と，良性データセットのみに出現し、悪性データセットには出現しない単語を，動作に影響しない関数の引数として追加する手法を考案する．検証実験では，攻撃者が言語モデルにアクセスできる場合とできない場合において，検知率がどの程度下がるかを評価する．その結果，攻撃者が言語モデルにアクセスできる場合，BoW を用いた分類器では検知率が 1.5%まで低下することを確認した．また，言語モデルにアクセスできない場合においても，LSI を用いた分類器では検知率が 73%低下することを確認した．

キーワード：VBA，マクロ，マルウェア，検知回避

Evaluating the risk of evasion attacks with VBA malware that abuses machine learning

Abstract: To detect unknown malware, several methods with machine learning techniques have been proposed. These methods extract the features of malware, and thereby classify samples as malicious or benign. Our previous method extracts words from source code and detect macro malware. This method constructs a language model Latent Semantic Indexing (LSI) to extract the features. On the other hand, several methods to avoid the detection have been proposed. These methods add benign features to the source code for Android malware. These methods can be applied to the macro malware. In this study, we discuss the risk of macro malware that evades detection. This paper attempts to imitate benign macros by adding benign features to macro malware. First, our method extracts the variables from macro malware and replaces them to frequent words of benign macros. Furthermore, our method adds the frequent words that appear in benign macros or the LSI topics. The words are added as the arguments of some functions, which do not vary main function. The target classifier is our previous method, which detects macro malware with language models: Bag of words (BoW) and LSI. The detection rates are evaluated under two conditions that the attacker can access inside the model or not. As a result, the detection rate with BoW decreases to 1.5% under the situation that the attacker can access inside the model. Even if the attacker cannot access inside the model, the detection rate with LSI decreases by 73%.

Keywords: VBA, macro, malware, evasion

1. はじめに

近年の情報通信技術の発展に伴い，サイバー関連技術の重要度が増加するとともに，サイバー攻撃の脅威も増加し

[1] 防衛大学校情報工学科
Depatment of Computer Science, National Defense Academy, Yokosuka, Kanagawa 239–8686, Japan

ている．キャノンマーケティングジャパン株式会社の 2019 年上半期マルウェアレポートによると，Visual Basic for Applications（VBA）マルウェアは国内で検出されたマルウェアの種類の 9.5%を占めており，JavaScript40.4%，32 ビット windows アプリケーション 26.5%，HTML11.6%についで第 4 位に位置している [*1]．また，現在猛威を振るっているマルウェアである EMOTET にも VBA が利用されている[*2]．このように，VBA マルウェアによるサイバー攻撃は非常に脅威の高いものである．VBA とは Microsoft Office で利用されるインタプリタ型のプログラミング言語であり，ソースコード形式で Microsoft 文書ファイルに埋め込まれる．

現在のウイルス対策ソフトは，主としてパターンマッチングによる検知を行っている [1]．しかしながら，この方法では未知のマルウェアの検知は困難であり，この課題を克服する手段として機械学習を用いた検知手法が研究されている．一方で，機械学習は敵対的攻撃に脆弱であることも指摘されている [2], [3], [4], [5], [6], [7], [8], [9]．このような敵対的攻撃に対しても耐性がある検知手法を開発するためには，具体的な攻撃の可能性を評価し，その対策を検討する必要がある．このような攻撃の可能性とその評価は，様々な研究で実施されている [2], [4], [5], [6], [7], [8], [9]．現実的な状況を考慮すると，攻撃者はマルウェアを対象のコンピュータに送付して遠隔操作することを企図しているため，そのマルウェアは機能を維持している必要がある．しかしながら，多くの研究では機械学習の分類器を回避することに主眼を置いており，マルウェアの機能の維持に注目した研究は少ない [6], [9]．また，知り得る限りでは，VBA マルウェアにおける敵対的攻撃の可能性を検討した研究は存在しない．そこで本研究では，VBA マルウェアの機能を維持したまま，機械学習によるマルウェア検知手法を回避する可能性を実証することを目的とする．検証手法として，マルウェアのソースコード内の変数を置換する手法と，新たなソースコードを追加する手法を用いる．回避の対象は自然言語処理技術である BoW および LSI を用いた検知手法 [10] とし，検証手法によってどの程度検知率が低下するかを評価する．

本研究の貢献は次に示すとおりである．

1．VBA マルウェアを対象に，機能を維持したまま検知を回避することが可能であるという脅威を実証した．

2．BoW を用いた分類器では，検知回避処理により検知率が 1.5%に低下することを実証した．

3．LSI を用いた分類器では，実践的な条件下でも，検知回避処理により検知率が 73%低下することを実証した．

[*1] https://eset-info.canon-its.jp/files/user/malware_info/images/ranking/pdf/MalwareReport_2019FirstHalf.pdf
[*2] https://blog.trendmicro.co.jp/archives/23648

2. 関連技術

自然言語処理の 1 つである Bag of Words(BoW) は，文書中のユニークな単語の出現数をベクトルの要素の値に変換する手法である．この手法では，文書中に出現する単語の種類が多いほど，特徴ベクトルの次元数は増加する．したがって，BoW を用いて分類処理を行う場合，次元数が多すぎると処理速度が低下してしまう．そのため，BoW では特徴ベクトルの次元数を適切に設定する必要がある．このような BoW の課題を解決する手法の 1 つとして，Latent Semantic Indexing（LSI）がある．LSI は特異値分解を用いた次元圧縮手法であり，ある文書群に含まれる単語において，ある単語に関連する概念の生成を試みる．LSI では任意の次元数を設定することが可能であるため，次元圧縮の手法として活用し，計算量を減らすことができる．次に，LSI を用いたベクトルの作成手法について説明する．まず文書数 n，文書全体に含まれるユニークな単語数 m の文書群に対して，Term Frequency-Inverse Document Frequency (TF-IDF）を用いて行列 T を求める．TF-IDF は TF と IDF を乗算したものであり，文書に含まれる単語の重要度を評価する指標である．TF はある文書中における単語の出現頻度であり，IDF は逆文書頻度を示す．TF-IDF は，ある文書内での出現回数が多いが，文書群の中の他の文書には出現しない単語の場合に高い値となる．

次に，この TF-IDF を用いた行列 T に対して特異値分解を行う．

$$T = USV^{\mathrm{T}} \tag{1.1}$$

$$T = \begin{bmatrix} u_{1,1} & \cdots & u_{1,m} \\ \vdots & \ddots & \vdots \\ u_{m,1} & \cdots & u_{m,m} \end{bmatrix} \begin{bmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_r \end{bmatrix} \begin{bmatrix} v_{1,1} & \cdots & v_{1,n} \\ \vdots & \ddots & \vdots \\ v_{n,1} & \cdots & v_{n,n} \end{bmatrix} \tag{1.2}$$

行列 U は m × m の正方行列であり，各列は TT^{T} と直交する固有ベクトル，行列 V は n × n の正方行列であり，各列は $T^{\mathrm{T}}T$ と直交する固有ベクトル，行列 S は n × m の対角行列であり，対角成分 σ は行列 T の特異値を示す．行列 U は単語ごとのトピックの重みを，行列 V は文書ごとのトピックの重みを示している．

式（1.1）において行列 S は対角行列のため，対角成分 σ が小さければ T に与える影響も小さくなる．任意の次元数 k を設定すると，上位 k 個の σ を残し，それ以外を 0 とする T' に近似した値となる．これによって次元数を圧縮することが可能となり，小さな特異値を 0 にすることによって圧縮前後で T-T' の各要素の 2 乗誤差が最小となる．つまり，元の文書の特徴を残したまま，次元圧縮をすることが可能となる．

3. 関連研究

3.1 機械学習によるマルウェア検知手法

本研究では，簡易で高速な静的分析に機械学習を用いたマルウェア検知手法に焦点をあてる．

Jeong らは PDF ファイルからバイトシーケンスを抽出し，畳み込みニューラルネットワーク（CNN) を用いて PDF のマルウェアを検知する手法を提案した [11]．この研究では，CNN の他に Support Vector Machine（SVM)，ナイーブベイズ，決定木およびランダムフォレストを用いた検知手法についても検証した．検証の結果，CNN が精度および処理速度で優れており，SVM は CNN の次に精度が高いことが確認された．

Amin らは Android マルウェアの APK からバイトコードを抽出して特徴ベクトルを作成し，モデルに制限ボルツマンマシンを使用したディープニューラルネットワークを用いて，マルウェアを検知する手法を提案した [12].

Vidyarthi らは静的分析と動的分析を組み合わせた PE ファイルマルウェアの検知手法を提案した [13]．この研究では，ソースコードにおける特徴を静的分析し，テキストマイニングを用いて exe ファイルの動的分析を実施している．この手法は既知のマルウェアでけでなく，難読化されたファイル，未知のマルウェアおよびサンドボックスなどを回避するマルウェアも検知できることを確認している．Kozachok らは PE ファイルのサイズや暗号化の有無，特定の API 呼び出しの有無，デジタル証明書の有無等，ソースコード以外の部分も特徴とし，マルウェアを検知する手法を提案した [14].

このように，静的分析を用いたマルウェア検知手法は広く研究されており，様々な種類のファイルを対象としている．

また VBA マルウェアに関しては，Kim らは難読化されている可能性が高い点に注目し，機械学習を用いて VBA の難読化を検出する手法を提案した [15]．この手法では，難読化されていない VBA マルウェアの検知は難しい．したがって本研究では，難読化されていない VBA マルウェアも考慮した検知手法に着目する．

3.2 機械学習による検知を回避する手法

Ehteshamifar らは，PDF ファイルにおいて静的および動的分析による検知を回避する手法を提案した [2]．この研究では，静的分析による検知を回避するために，悪意あるコードを実行時のみ読み込んで難読化している．Wang らは敵対的生成ネットワーク (GAN) を用いて，機械学習を用いた PDF マルウェアの検知を回避する手法を提案した [3]．この研究では，対象とした検知手法に対し，高い検知回避率を示すことが確認された．

Khormali らは Ramnit 等の Windows マルウェアや Mirai などの IoT マルウェアで敵対的学習を行い，その機能を維持したまま，ディープラーニングを用いた検知モデルを回避する手法を提案した [9]．Huang らは実行ファイルの API 呼び出しを追加し，マルウェアのソースコードを変更することによって，マルウェアの検知を回避する手法を提案した [7]．Chen らは，畳み込みニューラルネットワークを用いた PE ファイルのマルウェア検知手法を提案した [8]．この研究では，PE ファイルにソースコードを追加することによって検知を回避している．一方で，再訓練や訓練データセットの作成方法の変更によって，この攻撃を防ぐことができることを実証した．

Chen らはさらに Android マルウェアの検知モデルに対し，敵対的サンプルを追加することで検知率を低下させることができることを実証した [5]．Abaid らは Android マルウェアの検知モデルに対して敵対的学習を行い，その影響を評価した [4]．Grosse らは Android マルウェアにおいて，良性アプリケーションの特徴を持つコードを追加することによって，マルウェアの機能を維持したまま動的解析による検知を回避することが可能であることを示した [6].

このように，PDF [2], [3]，PE ファイル [7], [8], [9] および Android [4], [5], [6] のマルウェア検知手法に対する攻撃の脅威はこれまでにも指摘されている．しかしながら，VBA マルウェアの検知モデルに対する攻撃の脅威はこれまでに報告されていない．Android マルウェアのように，良性アプリケーションの特徴をソースコードに付与する攻撃は，VBA マルウェアにおいても可能であると考えられる．したがって本研究では，VBA マルウェアの検知モデルに対し，検知を回避する可能性を検証する．

4. 検証手法

4.1 概要

先行研究で提案された手法 [10] では，ソースコードを単語に分割して BoW または LSI で特徴ベクトルを作成し，SVM で悪性マクロと良性マクロを分類している．この手法はマクロを構成する単語を特徴とするため，難読化されていない悪性マクロにも対応することが可能である．よって本研究では，この検知手法を回避の対象とし，悪性マクロの機能を維持したままソースコードの修正を試みる．修正手段として，ソースコードの一部を置換する手法と，新たなソースコードを挿入する手法を考案した．検証の手順を図 1 に示す．

4.2 訓練プロセス

訓練プロセスでは，先行研究で提案された手法 [10] で分類器を訓練する．まず，訓練データから単語を特徴として抽出し，言語モデルを作成する．言語モデルを作成する際に，BoW および LSI を用いる．分類器の訓練に使用する LSI モデルのトピック数は，先行研究で示された最適値

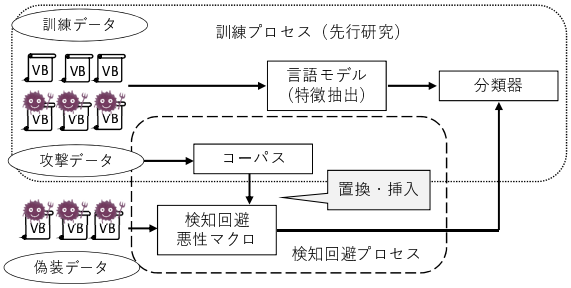

図 1　検証の手順

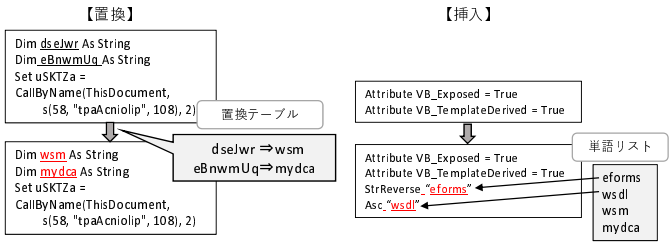

図 2　置換および挿入手法の例

である 400 とする．そして，作成した言語モデルを用いて SVM の分類器を訓練する．ここまでが図 1 の訓練プロセスである．

4.3　検知回避プロセス

訓練プロセスの後，偽装データに対して検知回避プロセスを実行する．偽装データは，分類器による検知を回避するために良性の特徴を付与する悪性マクロである．検知回避プロセスでは，悪性マクロのソースコードの置換または挿入を実施する．この時，置換および挿入する単語の参照元となるデータを攻撃データとする．置換および挿入手法の具体例を図 2 に示す．

4.4　置換

ソースコードを置換する処理では，まず偽装データからソースコードを抽出する．そして，この抽出したソースコード中のすべてのユーザが定義した変数を，別の単語に置換する．置換する単語は，攻撃データの良性マクロに頻出する単語とした．図 2 の例では，ソースコード中に含まれる変数は”dseIwr”および”eBnwmUq”の 2 種類である．これらの変数を，良性データから選択した”wsm”および”mydca”にそれぞれ置換している．

4.5　挿入

ソースコードを挿入する処理では，まず偽装データからソースコードを抽出する．そして，攻撃データ中の良性マクロのみに出現する単語や，良性の LSI トピックのみに含まれる単語を追加する．追加する単語は，マクロとしての機能を維持するため，” Asc”，” AscB”，” AscW”，” LCase”，” LTrim”，” Trim”，” StrReverse”，” Ucase”，” Split” のいずれかの関数の引数として追加する．これらの関数は，引数に指定された文字列の順番や，大文字小文字を変換する等の操作しか実施しないため，悪性マクロの機能に対する直接的な影響は及ぼさないものと考えられる．単語を追加する際は，追加する単語のリストの中から 1 つずつ順番に追加する．図 2 の例では，単語リストから”eforms”および”wsdl”を選択し，これらを”StrReverse”および”Asc”の引数として追加している．

表 1　データセットの各サンプルの数

2015 年		2016 年		2017 年	
良性	悪性	良性	悪性	良性	悪性
622	870	1200	1150	2220	719

4.6　実装

検証手法は，CPU：IntelCore i7(3.40GHz)，メモリ：16GB，OS：Windows10 Home を搭載した PC において，Python 3.6 を用いて実装した．TF-IDF および LSI の実装には gensim 3.6.0 [*3]を，SVM の実装には scikit-learn 0.22.1 [*4]を用いた．

5.　検証実験

5.1　データセット

本実験で用いるデータセットの各サンプルの数は表 1 のとおりである．データセットは，2015 年～2017 年の間に初めて Virus Total にアップロードされたものから，ファイルの拡張子が doc，docx，xls，xlsx，ppt，pptx であり，マクロを含むものを収集した．悪性データは，収集したファイルの中で，58 種類のアンチウイルスソフトのうち半数以上が悪性と判定したものを選択した．良性データは，どのアンチウイルスソフトも悪性と判定しなかったものを選択した．なお，これらのデータに重複はない．

5.2　評価指標

本研究で使用する評価指標について説明する．評価指標には正解率（Accuracy），適合率（Precision），再現率（Recall），F 値（F-measure）の 4 種類を用いる．各指標の定義は式 2.1～2.4 に示すとおりである．

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

$$F\text{-}measure = \frac{2Recall \times Precision}{Recall + Precision} \tag{2.4}$$

*3　https://radimrehurek.com/gensim/
*4　https://scikit-learn.org/stable/

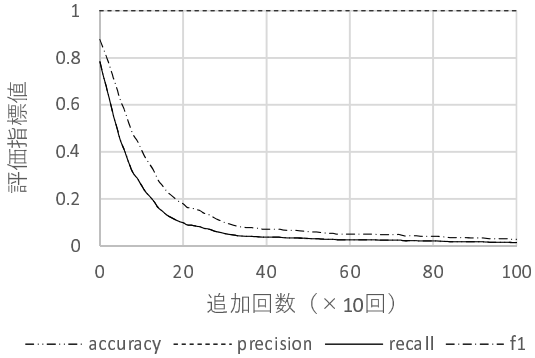

図 3 BoW における追加回数毎の分類精度の変化

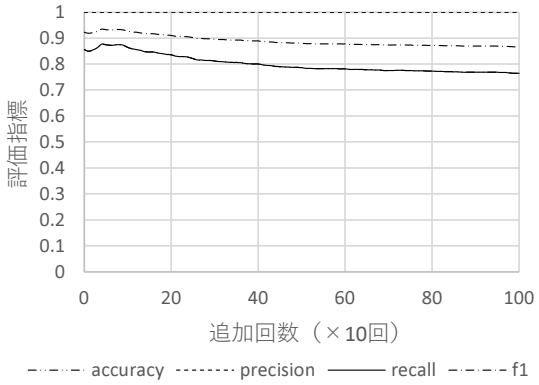

図 4 LSI における追加回数毎の分類精度の変化

5.3 実験内容と結果

各実験の条件を表 2 に示す．

5.3.1 置換処理の効果

まず、置換処理の効果を確認することを目的として実験を実施する．ここでは実用性は考慮せず，訓練データおよび攻撃データは 2016 年の良性および悪性のデータセットとし，偽装データは 2017 年の悪性データセットとする．言語モデルには BoW および LSI を用いる．各モデルにおける置換処理の分類精度を表 3 および表 4 に示す．BoW では Recall の低下は 3%にとどまり，LSI ではほとんど低下していない．したがって，置換処理の分類精度への影響はほとんどないものと考えられる．

5.3.2 挿入処理の効果の検証

次に，挿入処理の効果を確認する．挿入する単語は，攻撃データの良性マクロのみに出現し，悪性マクロには出現しない単語を選択する．訓練データおよび攻撃データは同様に 2016 年の良性および悪性のデータセットとし，偽装データは 2017 年の悪性データセットとする．言語モデルには BoW および LSI を用いる．各モデルにおける追加回数毎の分類精度の変化を図 3 および図 4 に示す．BoW では，Precision を除く各評価指標は単語を追加する毎に減少しており，最終的に Recall は 77%低下することが確認できる．しかしながら LSI では，単語を追加しても各評価指標はあまり変化しないことが確認できる．したがって，単純に良性マクロのみに出現する単語を追加しただけでは，BoW のような単純な手法には効果はあるものの，LSI にはあまり効果がないものと考えられる．

5.3.3 LSI に有効な手法の検証

LSI に対しても有効は手法を探るため，LSI のトピックに注目して挿入する単語を選択する．考案した単語の選択手法を図 5 に示す．まず，攻撃データの良性マクロおよび悪性マクロから，それぞれ良性 LSI モデルおよび悪性 LSI モデルを作成する．そして，良性 LSI モデルのトピックおよび悪性 LSI モデルのトピックを比較し，良性トピックのみに出現し，悪性 LSI トピックには出現しいない単語を挿入する単語として選択する．

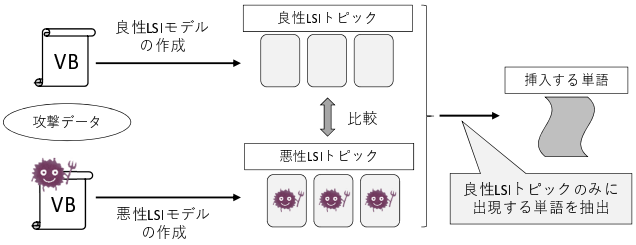

図 5 LSI のトピックに注目した単語の選択手法

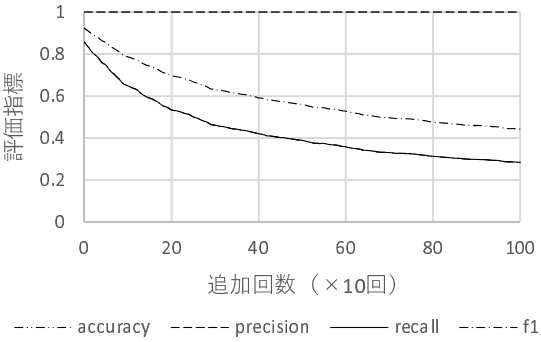

図 6 LSI のトピックに注目した手法による追加回数毎の分類精度の変化

訓練データおよび攻撃データは同様に 2016 年の良性および悪性のデータセットとし，偽装データは 2017 年の悪性データセットとする．攻撃データからトピック数を 400 として良性および悪性 LSI モデルを作成し，良性 LSI モデルのトピックのみに出現する単語を追加する．LSI モデルにおける追加回数毎の分類精度の変化を図 6 に示す．Precision を除く各評価指標は単語を追加する毎に減少しており，最終的に Recall は 27%に低下したことが確認できる．したがって，考案した単語の選択手法は LSI に対しても有効であると考えられる．よって以後の実験ではこの手法を用いて単語を選択する．

5.3.4 LSI のトピック数の影響

LSI のトピック数の影響を評価するために，攻撃データから作成する LSI モデルのトピック数を 100〜1000 の範囲で変更する．訓練データおよび攻撃データは同様に 2016 年の良性および悪性のデータセットとし，偽装データは

表 2　実験条件

No.	目的	手法	言語モデル	訓練データ	攻撃データ	偽装データ	単語の選び方
1	置換処理の効果を検証する.	置換	BoW	2016b,m	2016b,m	2017m	良性データの頻出単語
			LSI				
2	挿入処理の効果を検証する.	挿入	BoW				良性データのみに出現する単語
3	LSI を用いた分類器に対して有効な手法を検証する.		LSI				良性 LSI トピックのみに出現する単語
4	LSI のトピック数による影響を評価する.						
5	実践的な環境下での挿入手法の効果を検証する.			2016b，m(1)	2016b，m(2)	2016m(2)	
				2016b，m	2015b，m	2015m	

表 3　BoW に対する置換処理による分類精度の変化

	元データ	検知回避あり
訓練データ	2016	2016
攻撃データ	×	2016
偽装データ	2017 悪性	2017 悪性
Accuracy	0.78	0.76
Precision	1.00	1.00
Recall	0.78	0.76
F-measure	0.88	0.86

表 4　LSI に対する置換処理による分類精度の変化

	元データ	検知回避あり
訓練データ	2016	2016
攻撃データ	×	2016
偽装データ	2017 悪性	2017 悪性
Accuracy	0.85	0.85
Precision	1.00	1.00
Recall	0.85	0.85
F-measure	0.92	0.92

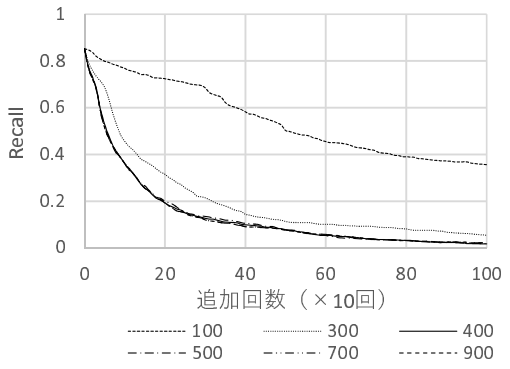

図 7　各トピック数における追加回数毎の Recall の変化

2017 年の悪性データセットとする．攻撃データからトピック数を 100〜1000 として良性および悪性 LSI モデルを作成し，良性 LSI モデルのトピックのみに出現する単語を追加する．各トピック数における追加回数毎の Recall の変化を図 7 に示す．トピック数が 100 の場合には，Recall の低下は 50%程度であったが，その他の場合には 83%低下した．この結果から，LSI モデルのトピック数は Recall の低下にあまり影響しないことが確認された．

5.3.5　実践的な環境での効果の検証

最後に，攻撃者が検知モデルにアクセスできないという

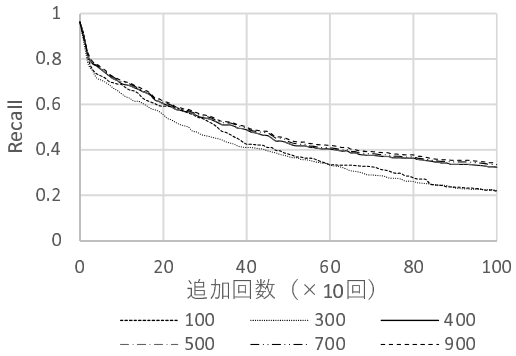

図 8　実践的な環境での各トピック数における追加回数毎の Recall の変化

実践的な状況を想定する．この状況を想定するため，攻撃データには訓練データとはまったく異なるデータセットを使用する．2016 年の良性および悪性データセットをランダムに半分に分割し，片方を訓練データとして，もう片方を攻撃データとする．偽装データは攻撃データに含まれる悪性マクロとする．LSI モデルは 100〜1000 のトピック数で作成する．実践的な環境での各トピック数における追加回数毎の Recall の変化を図 8 に示す．実践的な環境においても，Recall は 75%低下することを確認した．したがって，攻撃者が検知モデルにアクセスできないという実践的な状況であっても，検知を回避することが可能であると考えられる．トピック数毎の Recall の差は 14%程度であり，実践的な環境でも同様に，LSI モデルのトピック数は Recall の低下にあまり影響しないものと考えられる．

次に，攻撃データおよび偽装データをより古い 2015 年の良性および悪性データセットに変更する．この変更の意図は，攻撃者がより古いデータセットを用いたとしても，新しい検知モデルを回避することが可能であるかを検証することにある．LSI モデルのトピック数は 400 とする．攻撃者が不利な環境での LSI モデルにおける追加回数毎の分類精度の変化を図 9 に示す．Precision を除く各評価指標は単語を追加する毎に減少しており，Recall は最終的に 47%低下することを確認した．したがって，攻撃者が古いデータセットを用いる不利な状況であっても，検知を回避することは可能であると考えられる．

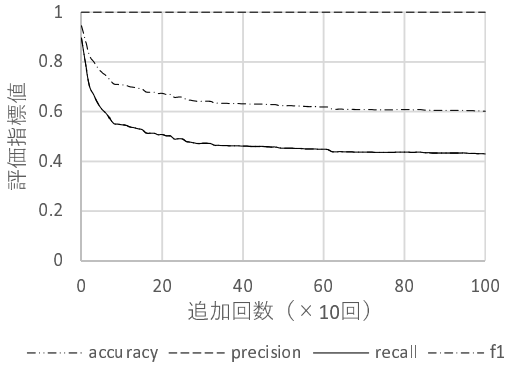

図 9 攻撃者が不利な環境での LSI モデルにおける追加回数毎の分類精度の変化

6. 考察

6.1 検知を回避する VBA マルウェアの脅威

検証実験の結果，置換処理では分類精度はほとんど低下せず，脅威とはならないことが確認できた．しかしながら挿入処理では，BoW および LSI を用いた分類器の検知精度は大きく低下することが確認できた．攻撃データに訓練データとまったく異なるデータセットを使用したとしても，LSI の Recall は 75%も低下することを確認した．この結果から，攻撃者が検知モデルにアクセスできないという実践的な状況であっても，検知を回避することは可能であると考えられる．さらに，攻撃データおよび偽装データを古いデータセットに変更したとしても，Recall は最終的に 47%低下することを確認した．この結果から，攻撃者が古いデータセットを用いる不利な状況であっても，検知を回避することは可能であると考えられる．したがって，VBA マルウェアにおいても良性アプリケーションの特徴をソースコードに付与する攻撃は可能であり，検知を回避する脅威はあり得ると考えられる．

6.2 攻撃に対するレジリエンス

検証実験では，BoW では単純に良性マクロのみに出現する単語を追加するだけで大幅に検知率が低下したのに対し，LSI では検知率はほとんど低下しなかった．したがって，LSI は BoW よりも攻撃への耐性が高いものと考えられる．しかしながら LSI においても，攻撃データから作成した良性および悪性の LSI モデルのトピックに含まれる単語のうち，良性 LSI モデルのトピックのみに出現する単語を追加することによって検知率が大幅に低下することが確認された．結局のところ，どちらのモデルも最終的に攻撃によって検知率は大幅に低下しており，良性アプリケーションの特徴をソースコードに付与する攻撃に対して脆弱であると言える．次に，LSI では良性マクロのみに出現する単語を追加するだけでは検知率が低下しなかった理由について考察する．良性マクロのみに出現する単語は 2914 語であったのに対し，良性 LSI トピックのみに出現する単語はわずか 164 語であった．これは，LSI の特異値分解によって，マクロの分類に寄与する貢献度の低い単語が消去されたためである．つまり，LSI トピックのみに出現する単語は，分類結果に大きな影響を与える重要な単語である．したがって，LSI トピックから追加する単語を選択することにより，効率的に良性マクロの特徴を偽装データに付与することが可能であると考えられる．そのため，最終的に LSI の検知率も大幅に低下したものと考えられる．良性マクロのみに出現する単語を追加するだけでは，必ずしも分類に寄与するとは限らない単語を大量に追加することになるため，検知率は低下しなかったものと考えられる．

6.3 攻撃への対策

良性アプリケーションの特徴をソースコードに付与する攻撃を防ぐ手法について考察する．

1 つ目の対策は，言語モデルを作成する際に，任意に単語を記述できる部分以外から特徴ベクトルを作成することである．今回検証したモデルでは，ソースコード内で任意に単語を記述できる部分とできない部分を区別せずに特徴ベクトルを作成する．そのため，任意に単語を記述できる部分に，実際には実行されないコードを追加することが可能であった．しかしながら，任意に単語を記述できる部分以外から特徴ベクトルを作成することができれば，良性アプリケーションの特徴をソースコードに付与する攻撃の効果は減少し，このような悪性マクロも検出することが可能となるものと考えられる．

2 つ目の対策は，分類器の再訓練である．検知を回避したマルウェアを悪性データとして分類器を再訓練することにより，検知率を回復させることが可能であると考えられる [8]．しかしながら，この対策は攻撃後に実行することになるため，事前に実施できる 1 つ目の対策と組み合わせる必要があるものと考える．

6.4 本研究の限界

本研究では，偽装データの悪性マクロの機能を維持したまま検知を回避する手法を検証することを目的とした．検証手法では，理論的には元の悪性マクロの機能を維持していると考えられるが，実際にマクロを実行して機能を確認したわけではない．

また，良性アプリケーションの特徴をソースコードに付与する攻撃への対策として，言語モデルの作成方法と分類器の再訓練を提案したが，実際にこれらの手法が対策として有効であるかも確認できていない．

6.5 研究倫理

本研究では，VBA マルウェアにおいても良性アプリケーションの特徴をソースコードに付与する攻撃は可能であり，

検知を回避する脅威はあり得ることを示した．本研究を実施するにあたり，倫理的側面には細心の注意を払った．検証手法は特別な環境を必要とせず，容易に実装することが可能であると考えられる．そのため，攻撃者がこの検証手法を悪用する可能性が考えられる．しかしながら，検証にはモデルを構築するためのデータセットが必要である．さらに，検証手法はウイルス対策ソフトのような伝統的なパターンマッチングによる検知手法を回避することは考慮していない．したがって，攻撃者が検証手法を悪用するのは容易ではないものと考えられる．しかしながら，今後この検証手法を悪用した VBA マルウェアが出現する可能性は否定できないため，本研究で使用したソースコードおよびデータセットは公開しないこととした．本研究の趣旨は，このような未知の脅威に警鐘を鳴らすことである．今後，このような攻撃への耐性を持つ検知手法を開発することで，レジリエンスの向上に貢献することができるものと考える．

7. おわりに

本研究では，VBA マルウェアにおいても良性アプリケーションの特徴をソースコードに付与する攻撃は可能であり，検知を回避する脅威はあり得ることを示した．検証実験では，BoW を用いた分類器では検知率が 1.5%まで低下し，LSI を用いた分類器では検知率が 73%低下することを確認した．この攻撃は，攻撃者が検知モデルにアクセスできないという実践的な状況であっても可能であり，攻撃者が古いデータセットを用いる不利な状況であっても有効であることを確認した．したがって，検証手法は機械学習を用いたマルウェア検知システムに対して大きな脅威になり得るものと考えられる．このような攻撃への対策としては，特徴ベクトルの作成範囲の変更や，分類器の再訓練が挙げられる．

今後の課題としては，これらの対策手法の実装やその評価が挙げられる．また，敵対的学習モデルを用いた回避手法の影響や脅威の有無についても検討する必要があるものと考えられる．

参考文献

[1] F. Biondi, T. Given-Wilson, A. Legay, C. Puodzius, and J. Quilbeuf, “Tutorial: An overview of malware detection and evasion techniques,” in *Leveraging Applications of Formal Methods, Verification and Validation. Modeling - 8th International Symposium, ISoLA 2018, Limassol, Cyprus, November 5-9, 2018, Proceedings, Part I*, 2018, pp. 565–586. [Online]. Available: https://doi.org/10.1007/978-3-030-03418-4_34

[2] S. Ehteshamifar, A. Barresi, T. R. Gross, and M. Pradel, “Easy to fool? testing the anti-evasion capabilities of PDF malware scanners,” *CoRR*, vol. abs/1901.05674, 2019. [Online]. Available: http://arxiv.org/abs/1901.05674

[3] D. Maiorca, B. Biggio, and G. Giacinto, “Towards adversarial malware detection: Lessons learned from pdf-based attacks,” *ACM Comput. Surv.*, vol. 52, no. 4, pp. 78:1–78:36, 2019. [Online]. Available: https://doi.org/10.1145/3332184

[4] Z. Abaid, M. A. Kâafar, and S. Jha, “Quantifying the impact of adversarial evasion attacks on machine learning based android malware classifiers,” in *16th IEEE International Symposium on Network Computing and Applications, NCA 2017, Cambridge, MA, USA, October 30 - November 1, 2017*, 2017, pp. 375–384. [Online]. Available: https://doi.org/10.1109/NCA.2017.8171381

[5] S. Chen, M. Xue, L. Fan, S. Hao, L. Xu, and H. Zhu, “Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach,” *Computers & Security*, vol. 73, 12 2017.

[6] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, “Adversarial examples for malware detection,” in *Computer Security – ESORICS 2017*, S. N. Foley, D. Gollmann, and E. Snekkenes, Eds. Cham: Springer International Publishing, 2017, pp. 62–79.

[7] Y. Huang, U. Verma, C. Fralick, G. Infantec-Lopez, B. Kumar, and C. Woodward, “Malware evasion attack and defense,” in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, June 2019, pp. 34–38.

[8] B. Chen, Z. Ren, C. Yu, I. Hussain, and J. Liu, “Adversarial examples for cnn-based malware detectors,” *IEEE Access*, vol. 7, pp. 54 360–54 371, 2019.

[9] A. Khormali, A. Abusnaina, S. Chen, D. Nyang, and A. Mohaisen, “Copycat: Practical adversarial attacks on visualization-based malware detection,” 2019.

[10] M. Mimura and T. Ohminami, “Towards efficient detection of malicious VBA macros with LSI,” in *Advances in Information and Computer Security - 14th International Workshop on Security, IWSEC 2019, Tokyo, Japan, August 28-30, 2019, Proceedings*, 2019, pp. 168–185. [Online]. Available: https://doi.org/10.1007/978-3-030-26834-3_10

[11] Y. Jeong, J. Woo, and A. R. Kang, “Malware detection on byte streams of PDF files using convolutional neural networks,” *Security and Communication Networks*, vol. 2019, pp. 8 485 365:1–8 485 365:9, 2019. [Online]. Available: https://doi.org/10.1155/2019/8485365

[12] M. Amin, T. A. Tanveer, M. Tehseen, M. Khan, F. A. Khan, and S. Anwar, “Static malware detection and attribution in android byte-code through an end-to-end deep system,” *Future Generation Comp. Syst.*, vol. 102, pp. 112–126, 2020. [Online]. Available: https://doi.org/10.1016/j.future.2019.07.070

[13] D. Vidyarthi, S. P. Choudhary, S. Rakshit, and C. R. S. Kumar, “Malware detection by static checking and dynamic analysis of executables,” *IJISP*, vol. 11, no. 3, pp. 29–41, 2017. [Online]. Available: https://doi.org/10.4018/IJISP.2017070103

[14] A. V. Kozachok and V. I. Kozachok, “Construction and evaluation of the new heuristic malware detection mechanism based on executable files static analysis,” *J. Computer Virology and Hacking Techniques*, vol. 14, no. 3, pp. 225–231, 2018. [Online]. Available: https://doi.org/10.1007/s11416-017-0309-3

[15] S. Kim, S. Hong, J. Oh, and H. Lee, “Obfuscated vba macro detection using machine learning,” in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2018, pp. 490–501.