Vol.2018-DPS-174 No.32
Vol.2018-CSEC-80 No.32
2018/3/6

Bag-of-wordsを用いた悪性マクロの検知手法の提案

三浦　紘弥[1,a)]　三村　守[1,b)]　田中　秀磨[1,c)]

概要：近年，悪性マクロを用いた標的型メール攻撃が増加している．これまでに，様々な悪性文書ファイルの検知手法が提案されているが，悪性マクロそのものを検知する手法は知り得る限りには存在しない．そこで本稿では，マクロのソースコードに注目し，特定の語彙の出現頻度から未知の悪性マクロを検知する手法を提案する．悪性マクロのほとんどは難読化されており，ＡＳＣＩＩコードの 16 進数，ランダムな変数名，文字列を操作する関数が使用される．提案手法では，ランダムな変数等の同一の特徴を示す語彙は同一の語彙に置換してコーパスを作成し，Bag-of-words を用いて特徴ベクトルを作成する．その後，作成した特徴ベクトルとラベルをサポートベクターマシンに入力し，悪性マクロと良性マクロを分類する．検証実験では，提案手法を Virus Total から取得した検体に適用し，未知の悪性マクロに対して効果があることを確認した．

キーワード：VBA，マクロ，サポートベクターマシン，自然言語処理技術，Bag-of-words

Detecting Malicious Macros with Bag-of-words

Hiroya Miura[1,a)]　Mamoru Mimura[1,b)]　Hidema Tanaka[1,c)]

Abstract: In recent years, targeted e-mail attacks using malicious macros are increasing. Previously, detection methods for malicious document files have been proposed. However, to the best of our knowledge, there is no method to detect malicious macros. In this paper, we focus on the source code and propose a method to detect malicious macros from the frequency of the specific vocabulary. Most malicious macros are obfuscated by the functions that convert a character string to hexadecimal numbers of ASCII codes and replace function names with random names. In our method, the vocabularies with the same features are replaced with a vocabulary to create a corpus. Thereafter, Bag-of-words represents the feature vectors. This paper inputs the feature vectors and labels into Support Vector Machine and classifies unseen macros into malicious macros and benign macros. In the verification experiment, we applied the proposed method to the specimen downloaded from Virus Total. As a result, our method is efficient to detect unseen malicious macros.

Keywords: VBA,macro，Support Vector Machine，natural language processing technique，Bag-of-words

1. はじめに

インターネットの普及に伴い，電子メールは連絡手段として一般的なものとなった．こうした中，標的型メール攻撃は，社会に大きな脅威となっている．

標的型メール攻撃とは，添付ファイルやメール本文に記載したリンクをクリックさせることで，メールの受信者の端末等にマルウェアを感染させる攻撃である．標的型メール攻撃の攻撃手法は，添付ファイル型と URL リンク型に区分される [1]．添付ファイル型の標的型メール攻撃は，悪性なファイルを添付し，受信者に添付ファイルを開かせることでマルウェアに感染させる手法である．URL リンク型の標的型メール攻撃は，悪意あるＵＲＬをメール本文に記載し，受信者に記載したＵＲＬに接続させることでマルウェアに感染させる手法である．

情報処理推進機構の標的型メール攻撃に関するレポート [1] によると，これら２つの標的型メール攻撃の手法のうち，９割以上は添付ファイル型の標的型メール攻撃が占める．そのため，添付ファイル型の標的型メール攻撃の対

[1] 防衛大学校理工学研究科
National Defense Academy, Hashirimizu 1-10-20, Yokosuka city, Kanagawa 239–8686, Japan
[a)] em56030@nda.ac.jp
[b)] mim@nda.ac.jp
[c)] hidema@nda.ac.jp

策は特に重要である．

近年の調査 [2] では，標的型メール攻撃で使用する添付ファイルのうち，８５％がＭＳ文書ファイルによるものであることが知られている．この調査 [2] では，悪性なＭＳ文書ファイルのほとんどはマクロによって作られていると報告されている．しかしながら，悪性なＭＳ文書ファイルの検知手法に関する研究の数は，我々の知る限り多くない．そのため，悪性なＭＳ文書ファイルの検知は研究の余地があり，より効果的な検知手法を検討する必要があるものと考えられる．

悪性なＭＳ文書ファイルに埋め込まれるマクロ（以下悪性マクロ）のソースコードは難読化されている場合が多い．難読化の手法は，ソースコード中の文字列をランダムな文字列に置換するものや，ＡＳＣＩＩコードの１６進数を文字列に変換するものなどがある．このように，難読化された悪性マクロのソースコードには，文字列を置換する関数や，１６進数が多く出現するなど，良性マクロには見られない特徴がある．そのため，悪性マクロの特徴に注目することによって，効果的な悪性マクロの検知が可能であると考えられる．そこで本稿では，前述のような悪性マクロのソースコードの特徴に注目し，未知の悪性マクロを検知する手法を提案する．提案手法では，１６進数等の同一の特徴を示す語彙を１語に置換してコーパスを作成し，自然言語処理技術の一つである Bag-of-words を用いて特徴ベクトルを作成する．その後，作成した特徴ベクトルとラベルをサポートベクターマシン (以下ＳＶＭ) に入力し，悪性マクロと良性マクロに分類する．さらに，提案手法を Virus Total から取得した検体に適用し，未知の検体に対して効果があるかを検証する．

本稿の構成を以下に示す．第２章では，関連研究について紹介し，本研究との相違点を明らかにする．第３章では，関連技術について説明する．第４章では，提案手法を説明する．第５章では，実験について説明する．第６章では，実験によって得られた実験結果を考察する．第７章では，本研究のまとめを記述する．

2. 関連研究

本章では，関連研究を紹介し，提案手法との相違点を明らかにする．まず，Bow を用いてマルウェアを検知する手法に関する研究を紹介する．次に，悪性ＭＳ文書ファイルを検知する手法に関する研究を紹介する．最後に，標的型メール攻撃に使用される悪性文書ファイルに関連して，悪性ＰＤＦファイルを検知する手法について紹介する．

2.1 Bow を用いたマルウェア検知に関する研究

はじめに，Bow を用いた悪性ファイルの分類手法に関する研究を紹介する．文献 [4] では，悪性実行ファイルのバイナリコードからｎグラムおよび Bow を用いて特徴ベクトルを作成し，ＳＶＭを用いて悪性実行ファイルを分類する手法を提案している．文献 [5] では，Windows の PE ファイル中の文字列から，Bow を用いて特定の文字列の有無を示す１か０の特徴ベクトルを作成し，SVM 等の分類器を用いて悪性な PE ファイルの分類を行っている．これらの関連研究は，Bow およびＳＶＭを用いて悪性ファイルの分類を行う点で，われわれの手法と類似するが，悪性マクロの分類を行うものではない．

2.2 悪性ＭＳ文書ファイルの検知に関する研究

次に，悪性ＭＳ文書ファイルの分類手法に関連する研究を紹介する．文献 [6] は，docx ファイルのパス構造から特徴ベクトルを作成し，ＳＶＭ等を用いて，悪性 docx ファイルを分類する手法を提案している．文献 [7] では，悪性マクロのエミュレーションの実行結果および，メール本文のテキストに基づき，悪性ＭＳ文書ファイルを分類する手法を提案している．これらの関連研究では，悪性ＭＳ文章ファイルを分類する点で，われわれの手法と類似するが，悪性マクロそのものを分類する手法ではない．

2.3 悪性ＰＤＦファイルの検知に関する研究

次に，標的型メール攻撃に頻繁に使用される悪性ＰＤＦファイルの分類手法について説明する．文献 [8] では，JavaScript が参照する，不審なＡＰＩの参照頻度から，悪性ＰＤＦファイルを分類する手法を用いている．文献 [9] では，JavaScript のソースコードの特徴に着目した静的解析と，悪性ＰＤＦファイル実行時の挙動に着目した動的解析を行い，悪性ＰＤＦファイルを分類する手法を提案している．文献 [10] では，JavaScript のソースコードの難読化を解除し，ソースコードに記述される不審なシェルコードを解析し，悪性ＰＤＦファイルを分類する手法を提案している．これらの手法では，ソースコード中の特定の関数等や難読化されたコードから特徴を得る点で，われわれの手法に類似する．しかしながら，これらの手法は，悪性ＰＤＦの検知のみを目的としているため，われわれの手法と目的が異なる．

3. 関連技術

3.1 悪性マクロ

本節では，悪性マクロの挙動と，ソースコードにみられる特徴を説明する．悪性マクロはダウンローダ型悪性マクロとドロッパ型悪性マクロに区分できる．ダウンローダ型悪性マクロとは，文書ファイル開封後，受信者の端末を外部のサーバ等にアクセスさせ，マルウェアをダウンロードし，感染させる悪性マクロである．ドロッパ型悪性マクロとは，マルウェアが悪性マクロに埋め込まれている形式の悪性マクロである．ダウンローダ型悪性マクロとドロッパ型悪性マクロは，挙動が異なるため，使用される関数等の傾向も異なる．表１では，ダウンローダ型悪性マクロと，ドロッパ型悪性マクロに頻繁に使用される関数等を示している．

以下では，ダウンローダ型悪性マクロとドロッパ型悪性マクロの挙動について説明し，それぞれの悪性マクロのソースコードの特徴を説明する．

表 1　悪性マクロに用いられる関数等

ダウンローダ型悪性マクロ	ドロッパ型悪性マクロ
CreateObject 関数 Shell 関数 SendKey ステートメント Declare ステートメント	CustomProperties コレクション

3.1.1　ダウンローダ型悪性マクロ

ダウンローダ型悪性マクロとは，Web ブラウザ等の外部アプリケーションを呼び出し，攻撃者のサーバに不正な通信を行わせ，マルウェアをダウンロード・実行し，感染させる悪性マクロである．そのため，ダウンローダ型悪性マクロは，外部アプリケーション等の呼び出しを行うことが特徴として挙げられる．以下に外部アプリケーション等の呼び出しの例を挙げる．

１つ目の例は CreateObject 関数である．CreateObject 関数は，他のアプリケーションの機能の一部を一時的なオブジェクトとして返す関数である．

２つ目の例は Shell 関数である．Shell 関数は引数で指定した実行ファイルを起動することができる．

３つ目の例は SendKey ステートメントである．SendKey ステートメントとは，指定したキーを押下する命令である．SendKey ステートメントの機能から，Shell 関数によって起動された実行ファイルに，何らかの操作を自動で行うことができる．そのため，悪性マクロでは，起動した実行ファイルを操作するために，Shell 関数と SendKey ステートメントを併用する場合が多い．

４つ目の例は Declare ステートメントである．Declear ステートメントは，Windows API を呼び出すことができる．Windows API とは，Windows に用意されている各種機能を利用するための，システム関数のことである．Declare ステートメントを使用することにより，Windows が提供する各機能にアクセスすることができる．ダウンローダ型悪性マクロのソースコードには，これらの関数等が頻出することが特徴として挙げられる．

3.1.2　ドロッパ型悪性マクロ

ドロッパ型悪性マクロとは，マクロのソースコードにマルウェアのコードが埋め込まれている悪性マクロのことである．ダウンローダ型悪性マクロとは違い，ドロッパ型悪性マクロは，マクロそのものが不正な挙動を起こす特徴がある．

以下にドロッパ型悪性マクロにマルウェアのコードを埋め込む手法を説明する．

マクロには，CustomProperties コレクションというものが用意されている．CustomProperties コレクションは Excel のメタデータ（作成者，表題，タイトル等の情報）を保存することができるものである．この機能を悪用して，CustomProperties コレクションに悪性な実行ファイルのバイナリ等を埋め込む．マクロのソースコードに実行ファイルのバイナリは直接記述しないため，ソースコードからは確認することはできない．ドロッパ型悪性マクロのソースコードには，CustomProperties コレクションの記述が多く登場する点が特徴として挙げられる．

3.2　難読化手法

悪性マクロは，ソースコードが難読化されている傾向がある．そのため，難読化されたソースコードの特徴を捕捉することにより，効果的に悪性マクロを検知することが可能になると考えられる．以下に，悪性マクロに施される難読化手法を説明する．

典型的な悪性マクロの難読化手法は，表 2 に示す５つに分類できる．

１つ目の手法は，クラス名や関数名等を別の文字に置換する手法である．この手法には，クラス名や関数名を，他意のトークンに置換する手法とランダムな文字列に置換する手法がある．後者の手法では，ランダムに置換された文字列が２０字を超えることが多い．

２つ目の手法は，文字列を ASCII コードにエンコード・デコードする手法である．マクロにはＡｓｃＢ関数と，ＣｈｒＢ関数が用意されている．ＡｓｃＢ関数とは，文字列をＡＳＣＩＩコードに変換する関数である．ＣｈｒＢ関数とは，ＡＳＣＩＩコードに対応する文字列を返す関数である．これら２つの関数を併用することによって，文字列の可読性を下げることができる．悪性マクロの多くは，ＡＳＣＩＩコードが配列に格納されていることが多く見られる．そのため，悪性マクロのソースコードには要素数の多い配列が多数出現する．

３つ目の手法は，ある文字列に対して変数 Key で排他的論理和を行い，エンコードを行う手法である．変数 Key の多くは，具体的な値が分からないように複雑な計算もしくは論理演算が行われていることが多い．

４つ目の手法は，文字列を分割する手法である．文字列を細かく分けたものを変数にそれぞれ代入する．分割された文字列が代入された変数を結合することにより，もとの文字列を復元することができる．この手法は，２つ目の手法を用いて，ＡＳＣＩＩコードに変換された文字列に対して行うなど，他の難読化手法と複合することによって，より可読性を下げることができる．

５つ目の手法は，リフレクション機能を用いるものである．リフレクション機能とは，マクロの実行時に，引数として渡した文字列を命令として実行させる機能である．リフレクション機能は，動的に命令を処理することができるため，処理の内容を静的に解析することは困難である．マクロでは，CallByName 関数がリフレクション機能を持つ．CallByName 関数に，関数名等を引数として渡すことによって，引数の文字列を命令として実行させることができる．CallByName 関数に渡す引数を難読化することによって，実行する関数等が何であるかを秘匿することができる．このように，リフレクション機能を用いることによって，ソースコードの可読性を下げることができる．

3.3　Bag-of-words

本節では，提案手法で使用する自然言語処理技術の一つ

表 2　難読化手法

連番	摘　　要
1	クラス名等の置換による命令等の難読化
2	ASCII コードのエンコード・デコードによる文字列の難読化
3	排他的論理和を用いたエンコード変換による文字列の難読化
4	文字列の分割による文字列の難読化
5	リフレクション機能の使用による命令等の難読化

である Bag-of-words(Bow) について説明する．Bow は，文章中のトークンの出現回数を数え，出現回数に応じてそのトークンに相当するベクトルの要素の値に変換する手法である．Bow では，トークンの語順や意味を考慮しない．具体的に例を挙げて Bow について説明する．ここに 3 つの例文がある．

例文 1 **I have a pen**
例文 2 **I have an apple**
例文 3 **Apple-Pen**

これらの例文から重複なくトークンを抽出すると，

I　have a pen an apple Apple-Pen

のようなトークンを得ることができる．重複なく抽出したトークンに対応する出現回数を，各例文毎の特徴ベクトルの要素にする．

例文 1　**[1,1,1,1,0,0,0]**
例文 2　**[1,1,0,0,1,1,0]**
例文 3　**[0,0,0,0,0,0,1]**

このような手法で，例文 1 ～ 3 は 7 次元ベクトルに変換することができる．

Bow では，トークンの種類が多いほど，特徴ベクトルの次元数が増加する．そのため，実運用を想定して Bow を用いる場合は，次元数を調整する必要がある場合がある．

4. 提案手法

4.1 概要

提案手法は，未知の悪性マクロを高い精度で検知することを目的としている．以下に提案手法の概要を説明する．

提案手法の流れを図 1 に示す．提案手法では，訓練データを既知の検体として，テストデータを未知の検体として扱う．提案手法では，訓練データとテストデータのＭＳ文書ファイルからマクロのソースコードを抽出する．抽出したソースコードを分かち書きし，トークンを作成する．分かち書きは特殊文字を空白に置換することによって行う．難読化の特徴が類似する複数のトークンを，特徴毎に 1 つのトークンに置換処理を行う．置換処理によって得られたトークンを，Bow を用いて特徴ベクトルを作成する．訓練データの特徴ベクトルとラベルをＳＶＭに入力して，学習させる．最後に，学習させたＳＶＭにテストデータの特徴ベクトルを入力し，ラベルを得る．

4.2 ソースコードの抽出

提案手法では，olevba[12] を用いてＭＳ文書ファイルからマクロのソースコードを抽出する．olevba は実行対象のＭＳ文書ファイルから，マクロのソースコードを抽出することができるオープンソースソフトウェアである．実行対象がＭＳ文書ファイル（Excel や Word 等）であれば，プラットフォームに関係なく実行することができる．

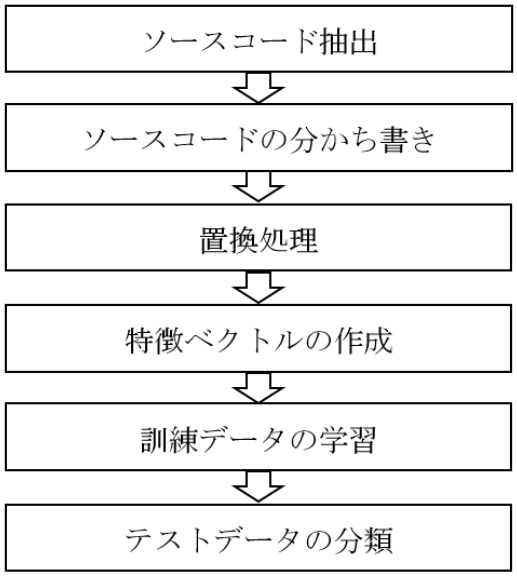

図 1　提案手法の流れ

4.3 分かち書き

検体からマクロのソースコードを抽出した後，分かち書きを行い，コーパスを作成する．分かち書きは，特殊文字を空白に置換し，変数名や関数名，クラス名等をトークンとなるよう行う．これは，悪性マクロと良性マクロに差異が生まれることを狙いとしている．具体的に，置換した特殊文字を表 3 に示す．

表 3　置換した特殊文字

特殊文字	名　称	特殊文字	名　称
“	ダブルクォート	＋	プラス
’	シングルクォート	/	スラッシュ
{	中括弧	&	アンド
(	丸括弧	%	パーセント
,	カンマ	¥	円マーク
.	ピリオド	$	ドルマーク
*	アスタリスク	#	シャープ
-	ハイフン	@	アットマーク

4.4 置換処理

本節では，マクロのソースコードの置換処理の手法を説明する．置換処理の目的は，悪性マクロにおいて同一の特徴を持つ異なるトークンを 1 語に置換することによって，悪性マクロの分類に効果的な特徴ベクトルを作成し，未知の悪性マクロの分類精度を向上させることである．

置換処理の例を示す．たとえば，以下のように１６進数の値を 1 つのトークンに置換する．

置換処理前：0xFF 0x14 0xA2

置換処理後：0xhex 0xhex 0xhex

この例では，置換処理前では各１６進数は異なるトークンとして処理されるが，置換処理後には同一のトークンとして処理される．

Bow の特徴ベクトルの要素は，トークンの出現頻度によって構成されている．そのため，置換処理を行うことによって，悪性マクロの特徴ベクトルの要素の値を大きくすることができ，未知の悪性マクロの分類精度を向上させることが期待される．詳細な置換処理の手法を表 4 に示す．以下に置換処理の手法を説明する．

表 4 置換処理

手法	置換する文字パターン	置換後の文字
1	16 進数 (0x ＸＸの形式のもの)	0xhex
2	16 進数 (&H ＸＸの形式のもの)	andhex
3	Asc,AscB,AscW	asc
4	20 文字以上の文字列	longchr
5	20 桁以上の数字	longnum
6	配列の要素	elementofarray

手法１および２は，ＡＳＣＩＩコードを１つのトークンに置換する手法である．難読化された悪性マクロは，ASCII コードを多用する傾向にあるため，ＡＳＣＩＩコードを１つのトークンに置換処理を行うことによって，悪性マクロの分類に有効な特徴を得ることができる．ＡＳＣＩＩコードは，１６進数で記述される．マクロのソースコードでは，１６進数は＆ＨＸＸや 0x ＸＸの形式 (ＸＸには任意の１６進数が入る) で表現されており，これらをそれぞれ１語の語彙に置換する．

手法３は，Ａｓｃ Ｂ関数に類似する関数を１つのトークンに置換する手法である．3.2 節で述べたとおり，ＡＳＣＩＩコードのエンコードは主にＡｓｃＢ関数を用いて行う．ＡｓｃＢ関数はＡｓｃ関数やＡｓｃW関数など類似する関数が存在する．Ａｓｃ関数は引数に渡した文字列を，半角文字に変換する関数である．ＡｓｃW関数は引数に渡した文字列を，ＡＳＣＩＩコードまたはＵＮＩＣＯＤＥとして返す関数である．これらの関数は，ＡｓｃＢ関数に比べて出現頻度は低い．しかしながら，Ａｓｃ関数やＡｓｃW関数の良性マクロにおける出現頻度は低かった．そのため，Ａｓｃ関数，ＡｓｃW関数およびＡｓｃＢ関数を１つのトークンに置換した．

手法４および５は，２０字以上の文字列および２０桁以上の数値を，１つのトークンに置換する手法である．悪性マクロにおいては，ランダムに置換された文字列は，２０字以上の文字列であることが多い．そのため，悪性マクロでは，２０字以上の文字列を１つのトークンに置換することで，悪性マクロの分類に有効な特徴を得ることができる．また，２０桁以上の数値も多く出現するため，これらも１語のトークンに置換した．

手法６は，配列の要素を１つのトークンに置換する手法である．悪性マクロにおいては，配列に無意味な文字列や，数値が格納されていることが多い．そのため，配列の要素を１つのトークンに置換することによって，悪性マクロの分類に有効な特徴を得ることができる．

4.5 特徴ベクトルの作成

置換処理を行ったコーパスに対して，Bow を適用して特徴ベクトルを作成する．Bow の特徴ベクトルの要素は，トークンの出現頻度で表現される．そのため，出現頻度が多いトークンがベクトル変換されることによって，分類に有効な特徴ベクトルを作成することができるものと考えられる．よって，提案手法では，特徴ベクトルに変換するトークンは，出現頻度の高いトークンを順に使用する．

特徴ベクトルに変換する手法は，良性な検体のコーパス，悪性な検体のコーパス，またはその両方のコーパスから作成する手法が考えられる．しかしながら，これらの手法を用いて特徴ベクトルを作成したとき，それぞれの分類精度の優劣は不明確である．そこで，提案手法では予備実験を行い，未知の悪性マクロの分類精度の高い手法を選択する．

4.6 訓練データの学習およびテストデータの分類

訓練データの学習では，訓練データの特徴ベクトルとラベルをＳＶＭに入力し，学習を行う．ＳＶＭのパラメータは，デフォルト値である，Ｃ＝１，gamma=auto およびカーネル関数を RBF(Radial Basis Function) に設定する．

テストデータの分類では，テストデータの特徴ベクトルを学習したＳＶＭに入力し，テストデータのラベルを得る．

5. 実験

5.1 概要

本章では，予備実験および検証実験について説明し，それらの結果について説明する．予備実験では，未知の悪性マクロの分類に効果的なコーパスを調査する．検証実験では，１０分割交差検証を行う．１０分割交差検証では，２０１５年の検体および２０１６年の検体全体に適用する．また，検証実験では，提案手法に基づく試験システムを実装し，未知の悪性マクロの分類精度を評価する．この検証実験では，２０１５年の検体を訓練データとして，２０１６年の検体をテストデータとして使用する．

5.2 実験環境

提案手法を表 5 に示す環境で実装した．実装に使用した言語は Python2.7 である．Bow の実装には gensim(0.10.1)[13] を使用した．gensim は，Bow や Doc2Vec 等の自然言語処理技術に関する機能を多数備えている．SVM の実装には，Python のモジュールの一つである scikit learn(0.19.0)[14] を用いた．scikit learn は機械学習ライブラリであり，多数の分類アルゴリズムを備えている．

表 5 実験環境

ＣＰＵ	IntelCorei7(3.40GHz)
メモリ	8GB
ＯＳ	Windows10Home

5.3 データセット

検証実験で使用したデータセットの内訳を表 6 に示す．検体は Virus Total で収集した．検体は，２０１５年および２０１６年に初めて Virus Total にアップロードされたものを選んだ．検体の選択条件は以下のとおりである．検体は，ファイルの拡張子が doc，xls，ppt，docx，xlsx，pptx であるもののうち，マクロが埋め込まれているものを選択した．悪性な検体は，５８種類のアンチウイルスソフトのうちの半数以上が悪性ファイルと判定したものを選択した．良性な検体は，５８種類のアンチウイルスソフトが１つも悪性ファイルと判定しなかったものを選択した．なお，これらの検体に重複はない．

表 6　データセットの内訳

2015 年検体		2016 年検体	
良性ファイル	悪性ファイル	良性ファイル	悪性ファイル
622 個	515 個	1200 個	641 個

5.4 評価規準

実験では悪性マクロのラベルを１，良性マクロのラベルを０とし，Precision(P),Recall(R) および F 値 (F) を評価手法として用いた．各評価手法の定義は次の数式のとおりである．

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

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

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

式中の TP(True Positive),FP(False Positive),TN(True Negative) および FN(False Negative) の関係は表 7 に示すとおりである．

表 7　クラス分類の定義

		真の値	
		正	負
予測結果	正	TP	FP
	負	FN	TN

5.5 予備実験

Bow で作成される特徴ベクトルの要素は，コーパスの選択次第で変化するため，未知の悪性マクロの分類精度に差異が出ると考えられる．そのため，予備実験を行い，コーパス毎の未知の悪性マクロの分類精度を調査・比較し，分類に有効なコーパスを採用する．

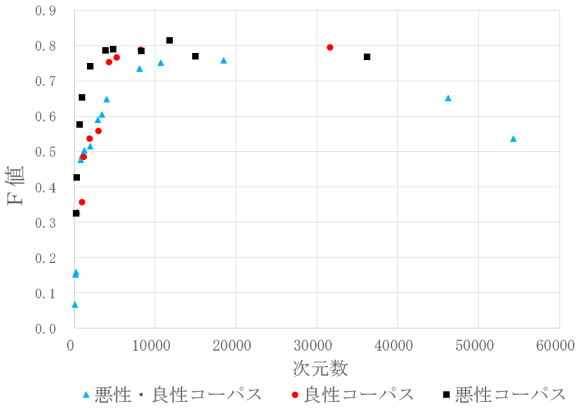

図 2　各特徴ベクトルの作成手法毎の分類精度

5.5.1 実験手法

予備実験は，提案手法に準ずる形で行う．ただし，分かち書きは”（”（丸括弧),’”’（ダブルクォート),”’”（シングルクォート）を空白に置換して行い，かつ置換処理は行わない点が提案手法と異なる．予備実験では，訓練データのうち，良性な検体のコーパス（良性コーパス），悪性な検体のコーパス（悪性コーパス），またその両方（悪性・良性コーパス）の３種類のコーパスから特徴ベクトルを作成する．各コーパスから作成した次元数の異なる特徴ベクトルおよびラベルをＳＶＭに学習させ，テストデータの特徴ベクトルをＳＶＭに入力して未知の検体の分類精度を求める．予備実験の結果は，次元数毎のＦ値をもって総合的に評価する．

5.5.2 実験結果

予備実験の結果を図 2 に示す．図 2 は，横軸を特徴ベクトルの次元数，縦軸をＦ値として，各手法を比較したものである．予備実験の結果から，悪性コーパスから特徴ベクトルを作成する手法では，３０００次元付近でＦ値が上昇し，Ｆ値が高い状態を安定して維持することがわかった．また，悪性コーパスから特徴ベクトルを作成する手法では，１２０００次元付近のＦ値の値が，他の手法よりも一番高い値を示した．そのため，悪性コーパスから特徴ベクトルを作成する手法が未知の悪性マクロの分類に比較的有効であると評価した．よって，検証実験では悪性コーパスを用いて特徴ベクトルを作成する．

5.6 検証実験

検証実験では，１０分割交差検証および，提案手法を実装した試験システムによる未知の悪性マクロの分類を行う．１０分割交差検証では，２０１５年の検体および２０１６年の検体全体に適用して行う．試験システムによる検証実験では，２０１５年の検体を訓練データとし，また２０１６年の検体をテストデータとして，未知の悪性マクロの分類および分類精度の評価を行う．

5.6.1 実験手法

１０分割交差検証の実験手法について示す．１０分割交差検証では，２０１５年の検体および２０１６年の検体全体に，提案手法の分かち書きおよび置換処理を行い，コー

パスを作成する．特徴ベクトルは，２０１５年の検体のうち，悪性な検体のコーパスから作成する．作成した特徴ベクトルおよびラベルを，ＳＶＭに入力し，１０分割交差検証を行う．

次に，提案手法に基づく試験システムを用いた検証実験の実験手法を示す．本検証実験では，２０１５年の検体を訓練データとして，２０１６年の検体をテストデータとして扱う．コーパスは，訓練データおよびテストデータ全体に，提案手法の分かち書きおよび置換処理を行って作成する．特徴ベクトルは，悪性コーパスから作成する．

実験結果は，特徴ベクトルの次元数毎のＦ値を評価する．また，特徴ベクトルの次元数毎の，悪性マクロの分類に要した実行時間を評価する．

5.6.2 実験結果

１０分割交差検証の結果を表８に示す．１０分割交差検証の結果では，高い精度で悪性マクロを分類できることが示された．１０分割交差検証で発生した誤検知においては，False Negative は発生せず，False Positive がいくつか発生した．F alse N egative が発生しなかった要因は，２０１５年の検体の悪性マクロは，２０１６年の検体の悪性マクロに類似する特徴があるためであると考えられる．また，False Positive の要因は，良性マクロに類似する悪性マクロが，いくつか存在するためであると考えられる．

次に，試験システムを用いて，未知の悪性マクロの分類を行った結果を図３に示す．図３は，予備実験で，悪性コーパスを用いた手法の結果をベースラインとして，試験システムのＦ値と比較したものである．全体として，試験システムのＦ値はベースラインのＦ値より高い結果を示した．試験システムのＦ値は，8000 次元に近づくにつれて上昇し，9000 次元以降緩やかに下降する傾向がみられた．

表９では，試験システムを用いた検証実験の詳細な結果を示している．8880 次元の時，Ｆ値は最大値をとった．21506 次元の時，Ｒの値が著しく下がり，併せてＦ値の値も下降した．この結果から，単に次元数が増加することによって分類精度が向上するわけではなく，ある次元数を発端に分類精度が下降する傾向があることが分かった．また，試験システムが分類に要した時間と，次元数は概ね比例の関係となっていることが確認できた．１つのマクロ当たりに要する実行時間は概ね 0.002 秒であった．

表 8　１０分割交差検証の結果

評価手法	値
P	0.871
R	1.000
F	0.931

6. 考察

6.1 誤検知の分析

本節では，誤検知の原因を分析する．False Positive が発生したマクロは，20 字以上の変数名等を多用していたものが多かった．悪性マクロの長い変数名を持つものは，ランダムにアルファベットを組み合わせていたものが多数であった．しかしながら，良性マクロの長い変数名を持つものは，複数のトークンを組み合わせてできていた．したがって，英語辞書を活用する等，ランダムな文字列と有意な文字列を識別するアルゴリズムを実装することによって，False Positive の数を減少させることができると考えられる．

次に，False Negative であったマクロの原因を分析する．False Negative は，難読化されていない悪性マクロに多くみられた．特徴ベクトルは，難読化されたトークンの特徴に偏って構成されている．そのため，難読化されていないマクロを悪性と分類できなかったものと考えられる．しかしながら，良性マクロのソースコードと比較して，False Negative と判断されたマクロのソースコードには，攻撃者のサーバの URL と思われる文字列や，データベースに何らかの操作を行う命令文が多くみられた．よって，URL やデータベースへの命令文を特徴として捉えられるよう，コーパスに置換処理を行うことによって，False Negative の数を減少させることができると考えられる．

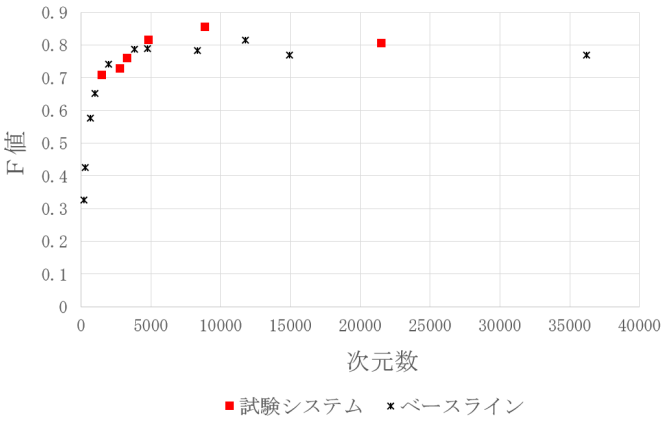

図 3　次元数毎の未知の悪性マクロの分類精度

表 9　次元数毎の分類精度と実行時間

次元数	P	R	F	実行時間 (秒)
1515	0.801	0.634	0.708	3.2
2792	0.773	0.686	0.727	5.6
3313	0.785	0.734	0.759	6.6
4831	0.814	0.815	0.815	9.9
8880	0.876	0.833	0.854	18.7
21506	0.986	0.68	0.805	44.7

6.2 置換処理の効果

未知の悪性マクロの分類に貢献した特徴的なトークンについて考察する．特徴的なトークンとは，悪性マクロでの出現率が高く，かつ良性マクロでの出現率の低いトークンである．この際，出現率とは，「あるトークンが出現するファイル数/ファイルの総数」である．検証実験で得られた，特徴的なトークンのうち，代表的なものを表 10 に示す．CreateObject 関数や CallByName 関数などが，特

徴的なトークンとして観測された．これらの難読化によく使用される関数等の他，elementofarray(配列の要素) や，longchr(２０字以上のトークン) 等の置換処理によって作られたトークンも多くみられた．これらの結果から，置換処理を行うことによって，分類精度を向上させることができたと考えられる．

表 10 特徴的なトークン

トークン	悪性マクロでの出現率	良性マクロでの出現率
elementofarray	99%	43%
andchr	93.9%	28%
next	90.9%	27.9%
function	85.1%	18.3%
string	83.3%	25.7%
len	79%	14.7%
public	77.5%	17.7%
longchr	73.7%	19.7%
createobject	73%	6.6%
error	73%	20.7%
byte	56.1%	1.5%
callbyname	51.3%	0.1%

6.3 実用性

表 9 から，１８００個程度のテストデータの分類に要する時間は数十秒程度であることが分かった．つまり，この結果は数個程度のマクロであれば極めて短時間で分類できることを示している．提案手法は，学習を事前に済ませることができるため，受信メールのマクロの分類に適用することが可能である．マクロを付したＭＳ文書ファイルのメールやりとりは，一般的に，一度に大量に送受信する可能性は低い．そのため，提案手法を応用することによって，リアルタイムな悪性マクロ検知システムを構築することが可能である．

検証実験の結果では，試験システムは，１０分割交差検証の結果に近似できる可能性を示した．例として，過去に受信したマクロを訓練データとして学習させることによって，提案手法の実運用を想定したとき，訓練データを月毎に学習させることで，１０分割交差検証の結果に近づくことが可能であると考えられる．つまり，より高い分類精度での悪性マクロの検知が可能であると考えられる．

提案手法では，リアルタイムな悪性マクロ検知システムの構築や，高分類精度への発展が可能であるため，実用性が高いと考えられる．

7. まとめ

本稿では，マクロのソースコードから Bow を用いて特徴ベクトルを作成するモデルを構築し，ＳＶＭによって未知の悪性マクロを検知する手法を提案した．予備実験では，特徴ベクトルの作成手法を複数試行し，未知の検体の分類に効果的な手法を調査した．検証実験では，１０分割交差検証を行い，高精度の分類が可能であることを示した．また，検証実験では提案手法に基づく試験システムを実装し，平易な置換処理を行ったものと，試験システムとをＦ値をもって比較した．その結果，試験システムでは，0.854 のＦ値を得ることができた．よって，提案手法は未知の検体の分類に効果的であると考えられる．

検証実験では，Virus Total から入手した２０１５年の検体と２０１６年の検体を用いた．そのため，実際に運用されるメールを利用するネットワークに提案手法を適用した場合の分類精度には検証の余地がある．よって，今後の課題は，メールを利用する実際のネットワークへの適用が挙げられる．

参考文献

[1] 標的型メール攻撃の傾向と事例分析 (https://www.ipa.go.jp/files/000036584.pdf)
[2] Wolf in sheep’s clothing: a SophosLabs investigation into delivering malware via VBA(https://nakedsecurity.sophos.com/2017/05/31/wolf-in-sheeps-clothing-a-sophoslabs-investigation-into-delivering-malware-via-vba/)
[3] TREND MICRO セキュリティマガジン (http://sp.trendmicro.co.jp/jp/trendpark/apt/201507-01/20150728065240.html)
[4] Robert Moskovitch, Malicious Code Detection Using Active Learning (https://www.researc hgate.net/publication/221654483_Malicious_Code_Detection_Using_Active_Learning)
[5] Maryann Gong, Classifying Windows Malware with Static Analysis
[6] Nir Nissim, ALDOCX: Detection of Unknown Malicious Microsoft Office Documents Using Designated Active Learning Methods Based on New Structural Feature Extraction Methodology (http://ieeexplore.ieee.org/document/7762928/)
[7] 碓井 利宣，幾世 知範，岩村 誠，矢田 健：悪性マクロ付き文書ファイルの解析効率化のための分類手法 (http://ci.nii.ac.jp/naid/40021162910/)
[8] Igino Corona, Lux0R: Detection of Malicious PDF-embedded JavaScript code through Discriminant Analysis of API References(https://dl.acm.org/citation.cfm?id=2666657)
[9] Daiping Liu, Detecting Malicious Javascript in PDF through Document Instrumentation(http://ieeexplore.ieee.org/document/6903571/)
[10] Xun Lu, De-obfuscation and Detection of Malicious PDF Files with High Accuracy
[11] Proactive defense against malicious documents: formalization, implementation and case studies(https://link.springer.com/article/10.1007/s11416-015-0259-6)
[12] olevba (https://github.com/decalage2/oletools/wiki/olevba)
[13] python package index gensim 0.10.1(https://pypi.python.org/pypi/gensim/0.10.1)
[14] python package index scikit learn 0.19.0(https://pypi.python.org/pypi/scikit-learn/0.19.0)