Computer Security Symposium 2018
22 - 25 October 2018

異常検知器を用いた未知の悪性マクロの検知手法

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による情報流出が問題視されている．先行研究では，悪性マクロと良性マクロのソースコードから抽出した単語を分類器に学習させ，未知の悪性マクロを検知する手法が提案された．しかしながら，この手法では，良性マクロと一般的に入手しにくい悪性マクロの両方が必要であるため，運用上の課題がある．そこで本稿では，入手が容易な良性マクロのみを用い，悪性マクロを異常検知する手法を提案する．検証実験では，2533 個の良性マクロを学習させ，収集元が異なる 3370 個のテストサンプルを分類させた．その結果，F 値が 0.99 となり，先行研究を上回った．さらに新種のマルウェアファミリを検知できることを確認した．

キーワード：異常検知，悪性マクロ，OneClassSVM，LOF，自然言語処理技術

Detecting Unseen Malicious Macros with Anomary Detection

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

Abstract: Recent days, information leakages caused by targeted email attacks using malicious macros are considered to be problematic. Previous work proposed a method which detects unseen malicious macros by learning extracted words from macros. However, the proposed method requires malicious macros and benign macros for training. Therefore, the previous work has operational issues. Hence, this paper proposes a method which requires only benign macros to detect unseen malicious macros. In verification experiments, benign macros of 2533 files are trained, and test samples of 3370 files in which dataset source is different. As the results, the proposed method obtained 0.99 of F-measure, and this result exceeded the previous work. Moreover, we confirmed that our method can detect unseen malware families.

Keywords: Anomaly detection, malicious macro, OneClassSVM, LOF, Natural Language Processing Technique

1. はじめに

近年，標的型メール攻撃による情報漏洩が問題視されている．アンチウイルスベンダーの Sophos のレポートによると，標的型メール攻撃に使用される添付ファイルの８５%がマイクロソフトオフィスの文書ファイル (MS 文書ファイル) であることが報告されている [1]．その文書ファイルに埋め込まれるマルウェアの多くが，Visual Basic for Application (以下 VBA) による悪性マクロであることが報告されている．クライアントが MS 文書ファイルを開封し，マクロの使用を許可したならば，悪性マクロは実行される．悪性マクロに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２種類が存在し，その挙動は異なる．ダウンローダ型悪性マクロは，被害者の端末を強制的に外部のサーバと通信させる．外部サーバと被害者の端末の通信が確立されたならば，ダウンローダ型悪性マクロはマルウェアを被害者の端末にダウンロードさせる．これに対してドロッパ型悪性マクロは，マクロそのものにマルウェアを埋め込み，実行時にそのマルウェアを展開して実行する．そのため，ドロッパ型悪性マクロは，ダウンローダ型悪性マクロと異なり，外部との通信を必要としない．

[1] 防衛大学校
NDA, Yokosuka city, Kanagawa 239–0811, Japan
[†1] 現在，防衛大学校
Presently with National Defence Academy in Japan
[a)] em56030@nda.ac.jp
[b)] mim@nda.ac.jp
[c)] hidema@nda.ac.jp

Nir らは，悪性ＭＳ文書ファイルのファイル構造に基づいて，悪性なＭＳ文書ファイルを検知する手法を提案した [4]．悪性マクロが埋め込まれるＭＳ文書ファイルの構造が良性マクロのものと類似する場合，この手法では良性マクロを悪性マクロとして誤検知する可能性がある．そのため，悪性マクロそのものを検知する手法は重要なタスクであると考えられる．そこで，われわれは，マクロに出現する単語の出現頻度をベクトルに変換し，そのベクトルを機械学習の分類器に入力することで悪性マクロを検知する手法を提案した [3]．この手法では，十分な数の悪性マクロと良性マクロの両方を学習に必要とするが，悪性マクロは一般的に入手し難い．そのため，この手法は運用上の課題がある．そこで本稿では，入手容易な良性マクロのみを分類器に学習させ，悪性マクロを異常検知する手法を提案し，先行研究の運用上の課題を解決する．

先行研究 [3] では，悪性マクロと良性マクロの数の比が概ね均衡のデータセットを用いて評価を実施している．しかしながら，現実的には受信するメールに含まれる悪性マクロの割合は少ないものと考えられる．そのため，悪性マクロの割合が少ない不均衡なデータセットを用いた方が，より実践的な精度を評価することが可能であると考えられる．また，先行研究では同一の収集元から作成したデータセット用いて評価を実施している．そのため，テストサンプルに出現する単語が訓練サンプルと類似しており，同じ特徴を示す可能性がある．しかしながら，この収集元から得られたマクロがすべてのマクロの特徴を集約しているとは限らない．したがって，複数の収集元から作成したデータセットを用いて評価を実施する必要がある．そこで本稿では，実際の運用を想定した不均衡なデータセットを作成し，提案手法の精度を評価する．また，複数の収集元から作成したデータセットを用い，提案手法の汎化性を検証する．

本稿の貢献は次に示すとおりである．

1　良性マクロのみを用いて悪性マクロを検知する手法を提案する．

2　不均衡データに対して，提案手法が高い検知率で悪性マクロを検知できることを示す．

3　複数の収集元から作成したデータセットを用い，提案手法に汎化性があることを示す．

4　提案手法が先行研究 [3] を上回る検知率で，悪性マクロを検知できることを示す．

5　提案手法が新種のマルウェアファミリを検知できることを示す．

本稿の構成を以下に示す．第２章は，関連研究を紹介する．第３章は，関連技術について紹介する．第４章は，提案手法について説明する．第５章は，検証実験の手法およびその結果について説明する．第６章は，検証実験の結果と提案手法に関する考察を行う．最後に第７章は本稿のまとめを述べる．

2. 関連研究

本章では悪性 MS 文書の検知手法に関連する研究を紹介し，本稿の新規性を明らかにする．

Nir らは，悪性な docx ファイルを検知するフレームワークである ALDOCX を提案した [4]．ALDOCX は，docx ファイルの構造と，docx ファイルを構成する xml ファイルの構造から特徴を抽出する．抽出された特徴を分類器に学習させ，学習済の分類器を用いて悪性な docx ファイルを検知する．Naser らは，docx ファイルの構造を解析し，不審なキーワードの有無によって悪性な docx ファイルを検知する手法を提案した [5]．これらの関連研究は，docx ファイルを対象とした研究である．そのため，ppt(pptx) ファイルおよび xls(xlsx) ファイルに対応することができない．本稿は文書ファイルに含まれるマクロを分類の対象としているため，ファイルタイプに関係なく分類を行うことが可能である．

大坪らは，悪性な文書ファイルを検知するツールとして，O-checker を提案した [2]．O-checker は，文書ファイル (MS 文書ファイル，pdf ファイルおよび jtd ファイル) に埋め込まれる実行ファイルの構造や，ファイルフォーマットの構造を解析し，悪性文書ファイルを検知する．Boldewin は悪性な実行ファイルやシェルコードを含む MS 文書ファイルを検知するツール (OfficeMalScanner) を実装した [6]．OfficeMalScanner は，MS 文書ファイルの構造をスキャンし，含まれる Windows API 名や，シェルコードのパターン，OLE データなどの文字列から，定める基準に従って不審度を採点する．OfficeMalScanner は，不審度が閾値を超えたならば当該ファイルを悪性と判断する．三村らは，doc ファイル，rtf ファイル，xls ファイルおよび pdf ファイルに埋め込まれる悪性実行ファイルの難読化を解除してこれを検知する手法を提案した [7]．三村らの手法を実際の検体に対して検証した結果，OfficeMalScanner の検知率を上回った．これらの関連研究は，文書ファイルに埋め込まれる悪性な実行ファイルや，シェルコードを検知するが，提案手法は悪性マクロの検知を行う．

われわれは，マクロのソースコードに出現する単語の出現頻度に基づいて，悪性マクロを検知する手法を提案した [3]．この手法は，良性マクロと悪性マクロの両方を必要とするが，一般的に悪性マクロは入手し難いため，この手法は運用上の課題がある．提案手法では，悪性マクロを必要とせず，入手容易な良性マクロのみを必要とする．

3. 関連技術

3.1 悪性マクロ

攻撃者が悪性マクロを攻撃対象に実行させる手口について説明する．悪性マクロを実行させるためには，MS 文書

ファイルの開封およびマクロの有効化を攻撃対象に行わせる必要がある．多くの場合，攻撃者は攻撃対象を信頼させるような巧みな文章をメール本文に記述する．攻撃者は，信憑性のあるメール本文を記述するために，事前に攻撃対象の業務内容等を入念に調査することがある．そのため，攻撃者から送付されるメールを通常のメールと勘違いして悪性マクロを実行してしまうと，大規模な情報漏洩といった大きなインシデントにつながる可能性がある．

次に，悪性マクロの挙動について説明する．悪性マクロには，ダウンローダ型悪性マクロとドロッパ型悪性マクロの２種類がある．ダウンローダ型悪性マクロは，攻撃対象の端末を強制的に外部のサーバに接続させ，マルウェアをダウンロードおよびインストールさせる悪性マクロである．これに対し，ドロッパ型悪性マクロでは，マクロそのものに不正なコードが埋め込まれている．ドロッパ型悪性マクロが実行されたならば，埋め込まれた不正なコードが実行され，攻撃対象の端末はマルウェアに感染する．ドロッパ型悪性マクロはマルウェア感染の過程で外部のサーバに接続する必要がないため，ダウンローダ型悪性マクロと挙動が異なる．

3.2 潜在意味インデキシング

本節では，潜在意味インデキシング (LSI) の概要と，LSI を用いたトピックベクトルの作成手法を説明する．LSI とは，文書群とそれに含まれる単語群について，それらに関連した概念 (トピック) の行列を生成する自然言語処理技術である．LSI は，文書分類や，データクラスタリング，文書間の類似度の計算等に用いられる．Bag-of-Words 等の自然言語処理技術で作成するベクトルは，次元数が固定長であり，かつベクトル変換する文書群によっては冗長な次元数となるため，計算量が著しく多くなることがある．これに対して，LSI の作成するベクトルは可変長であり，ユーザが任意に次元数を設定できるため，次元圧縮の手段として用いられる．

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，文書全体におけるユニークな単語数 m の文書群に対し，各文書の単語の出現頻度 tf をベクトルの要素とし，その要素が単語に対応する行列 $\mathbf{T}$ を作成する．

$$\mathbf{T} = \begin{bmatrix} tf_{11} & \cdots & tf_{1m} \\ \vdots & \ddots & \vdots \\ tf_{n1} & \cdots & tf_{nm} \end{bmatrix}$$

行列 $\mathbf{T}$ に対して Term Frequency-Inverse Document Frequrncy (TFIDF) を用いて行列 $\mathbf{T}$’ を求める．TFIDF とは，文書を代表する単語に重みづけする自然言語処理技術であり，行列 $\mathbf{T}$’ の各要素 t'_{ij} は，$0 < i < m+1$ および $0 < j < n+1$ の範囲内において，$tf_{ij} \times \log(df_i/n)$ の式で求める．このとき，df は文書頻度のことであり，ある単語が出現する文書数を意味する．次に，行列 $\mathbf{T}$’ に対して特異値分解を行う．

$$\mathbf{T}' = \mathbf{DVW}$$

$$= \begin{bmatrix} d_{11} & \cdots & d_{1r} \\ \vdots & \ddots & \vdots \\ d_{n1} & \cdots & d_{nr} \end{bmatrix} \begin{bmatrix} v_1 & & 0 \\ & \ddots & \\ 0 & & v_r \end{bmatrix} \begin{bmatrix} w_{11} & \cdots & w_{1m} \\ \vdots & \ddots & \vdots \\ w_{r1} & \cdots & w_{rm} \end{bmatrix}$$

行列 $\mathbf{V}$ は対角行列であり，その要素は $r_k < r_{k+1}$ の関係が成立する．行列 $\mathbf{V}$ の要素 r_k は文書群全体に存在するトピックを意味する．行列 $\mathbf{D}$ は行が各トピックに対応し，列が各文書に対応する行列である．行列 $\mathbf{W}$ は行が各単語に対応し，列が各トピックに対応する行列である．本稿では，行列 $\mathbf{D}$ と行列 $\mathbf{W}$ をそれぞれ文書トピックベクトル，単語トピックベクトルと呼ぶ．各トピックベクトルは，文書毎または単語毎の各トピックの軽重を表現する．提案手法では，文書トピックベクトルを用いる．

次に，訓練サンプルおよびテストサンプルから文書トピックベクトルを抽出する手法について説明する．まず，訓練サンプルに出現するすべての単語を抽出し，これらのユニークな単語の集合を作る．このユニークな集合を辞書と呼ぶ．次に，文書トピックベクトルの次元数を設定する．この次元数とは，トピック数のことである．辞書に登録されている単語と，設定した次元数に基づいて，訓練サンプルのトピックベクトルを抽出する．訓練サンプルから作成した辞書と，設定した文書トピックベクトルの次元数で文書トピックベクトルを抽出するモデルを，ＬＳＩモデルという．ＬＳＩモデルは，サンプル中の辞書に登録された単語のみを対象に，ＬＳＩモデルの次元数の文書トピックベクトルを抽出する．テストサンプルの文書トピックベクトルは，各テストサンプルをＬＳＩモデルに入力することによって抽出される．

3.3 難読化手法

悪性マクロのソースコードの多くは，解析を妨害するために難読化されている．本来難読化とは，商用のソフトウェアの著作権の保護や，ソースコードの改竄防止等を目的として開発されてきた．しかしながら，近年では攻撃者がマルウェアのコードの解析を妨害する目的で使用することが多くなってきている．とりわけ，悪性マクロにおいては，そのソースコードを確認することが容易であるため，難読化されていることが多い．本節では，悪性マクロに施される難読化の手法および難読化されたソースコードの特徴について説明する．典型的な悪性マクロの難読化手法は，表１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

２つ目の手法は，文字列をＡＳＣＩＩコードにエンコー

表 1　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	リフレクション関数の使用による命令等の難読化

表 2　置換する特殊文字

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
,	コンマ		円マーク
.	ピリオド	$	ドルマーク
*	asterisk	#	sharp
-	haihun	@	at mark

ド・デコードする手法である．ＡＳＣＩＩコードは 16 進数で記述されている．ＶＢＡ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．悪性マクロでは，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算術演算もしくは論理演算が行われている．

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，リフレクション関数を用いる手法である．リフレクション関数とは，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関数である．よって，リフレクション関数は動的に命令を処理することができる．そのため，分析者が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．

悪性マクロのソースコードには以上で述べたような難読化された単語が多く含まれるが，一般的な良性マクロには多く含まれない傾向がある．そのため，ＬＳＩで文書トピックベクトルを抽出する際，難読化された悪性マクロから良性マクロとは異なる文書トピックベクトルが抽出されるものと考える．

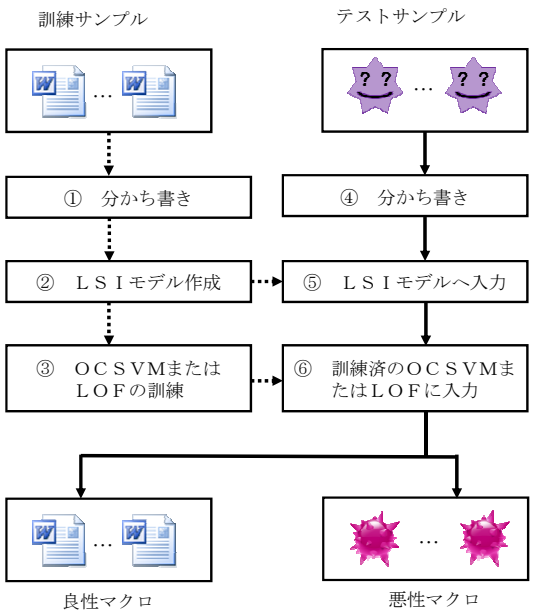


図 1　提案手法

4. 提案手法

先行研究 [3] で提案された手法では，一般的に収集が難しい悪性マクロを必要とするため，運用上の課題がある．良性マクロは，インターネット上に公開されているものが多く存在している．例を挙げるならば，プログラミングに関する質問サイトである teratail[8] 等である．そのため，良性マクロは悪性マクロに比べて入手が容易である．そこで本稿では，先行研究の運用性を向上させるため，良性マクロのみを用いて悪性マクロを検知する手法を提案する．提案手法では，分類器として異常検知器を用いる．異常検知器とは，訓練サンプルに１クラスしか必要としない分類器のことである．異常検知器は，入力されたテストサンプルが訓練サンプルと乖離した特徴を持つ場合，これを異常として検知する分類器である．

提案手法の概要を図１に示す．提案手法は，①から⑥の順序で処理を実行する．訓練サンプルは，良性ＭＳ文書ファイルから構成されている．テストサンプルは，良性なものか悪性なものかが未知であるＭＳ文書ファイルである．

表 3　実験環境

CPU	IntelCorei7(3.30GHz)
メモリ	32GB
OS	Windows8.1Pro

4.1　異常検知器の学習

①では，訓練サンプルからマクロを抽出し，そのソースコードを分かち書きする．一つのＭＳ文書ファイルから複数のマクロが抽出された場合は，これを統合し，一つのマクロとして取り扱う．次に，抽出したマクロのソースコードを分かち書きする．分かち書きは，表 2 に示す特殊文字を空白に置換することによって行う．分かち書きされたソースコードは，単語（変数や関数等）が空白で分割される．本稿では，分かち書きされたソースコードをコーパスと呼ぶ.

②では，訓練サンプルからＬＳＩモデルを作成する．そして，作成されたＬＳＩモデルに訓練サンプルのコーパスを入力して，文書トピックベクトルを抽出する．次に，③では，抽出された文書トピックベクトルを異常検知器に入力して訓練する．先行研究 [3] では，ここで悪性マクロと良性マクロから抽出した特徴ベクトルを分類器に入力するが，提案手法は良性マクロから抽出した文書トピックベクトルのみ入力する．異常検知器はＯＣＳＶＭとＬＯＦを用いる．ＯＣＳＶＭ [10] とは，単一のクラスのみ学習するＳＶＭである．ＬＯＦ[11] とは，密度ベースの異常検知器である．

4.2　テストサンプルの分類

④では，テストサンプルからマクロを抽出し，①と同じ手法で分かち書きを行ってコーパスを作成する．次に，⑤では，②で作成したＬＳＩモデルにテストサンプルのコーパスを入力して, 文書トピックベクトルを抽出する．最後に，⑥では，テストサンプルの文書トピックベクトルを訓練済みの異常検知器に入力し，テストサンプルを悪性マクロと良性マクロに分類する．尚，訓練済みの異常検知器は再利用が可能であるため，テストサンプルの分類の度に訓練を行う必要はない．

4.3　実装

提案手法を表 3 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．マクロの抽出には olevba[9] と呼ばれるツールを用いる．olevba とは，オープンソースのマクロ抽出ツールのことであり，Microsoft が提供する各種文書ファイルからマクロを抽出することができるツールである．OCSVM および LOF の実装には，Python のモジュールの一つである scikit learn（0.19.1）を用いた [12]. 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提供する．LSI の実装には，gensim（2.0.0）を用いた [13]．gensim は自然言語処理技術に関する機能を多数提供するモジュールである．

表 4　Virus Total から収集したサンプルの内訳

時期	2016 年度		2017 年度	
区分	良性	悪性	良性	悪性
サンプル数	1200	641	2220	1150

表 5　stackoverflow から収集した良性サンプルの内訳

時期	2015 年度	2016 年度	2017 年度
サンプル数	2533	10215	11702

5.　検証実験

提案手法の検知率を検証するため，検証実験を行う．提案手法では，入手容易な良性マクロのみを訓練に用る．そのため，検証実験では一般的に公開されているウェブサイトから，訓練サンプルを収集した．

5.1　データセット

本検証実験で用いるデータセットは Virus Total (VT)[14] および stackoverflow(SO)[15] から収集した．VT とは，マルウェア等の検体を提供するウェブサービスである．また，VT は著名なアンチウイルスベンダーと提携し，各社の製品によるマルウェア判定等のサービスを提供する．表 4 に VT から収集したサンプル数を示す．表 4 における時期とは，サンプルが VT に初めてアップロードされた時期のことである．サンプル全体の収集条件は，doc，docx，xls，xlsx，ppt，pptx ファイルのうち，マクロが埋め込まれているものとした．悪性サンプルの収集条件は，全 58 社のアンチウイルスベンダーのうち，半数以上が悪性であると判定したものとした．良性サンプルの収集条件は，全ベンダーが良性であると判定したものとした．尚，ＶＴから収集したサンプルに重複はない．

本稿では，容易に入手可能な良性マクロを用いて分類器を学習させることで，未知の悪性マクロを検知する点が主な特徴である．訓練に用いる良性サンプルを，世界的なプログラミングに関する質問サイトであるＳＯから収集した．2015 年度から 2017 年度の間の，マクロに関連する投稿のうち，ソースコードをクローラを用いて収集した．収集した各ソースコードのハッシュ値を求め，重複するものを削除した．これらのソースコードがマクロであるかを確認するため，各年度のソースコードからそれぞれ 500 個ずつランダムに抽出し，目視で点検した．収集した良性マクロの数を表 5 に示す．表 5 の時期とは，対象のマクロがＳＯに投稿された時期を意味する．

5.2　実験手法

本稿では，二つの検証実験を行う．先行研究 [3] では，悪性マクロと良性マクロの数が概ね均衡のデータセットを分類した．しかしながら，実際の運用を想定した場合，悪性マクロの数は少なく，テストサンプルの比率は不均衡になると考えられる．そこで一つ目の検証実験では，不均衡

表 6　不均衡データの分類における実験条件

区分	訓練サンプル	テストサンプル
年度	2015 年度	2016 年度
良性サンプル数	2533 (SO)	11415 (SO+VT)
悪性サンプル数	-	641 (VT)
年度	2015 年度	2017 年度
良性サンプル数	2533 (SO)	12902 (SO+VT)
悪性サンプル数	-	1150 (VT)

表 7　異なるデータセットを対象とした分類における実験条件

区分	訓練サンプル	テストサンプル
年度	2015 年度	2016 年度
良性サンプル数	2533 (SO)	**1200 (VT)**
悪性サンプル数	-	641 (VT)
年度	2015 年度	2017 年度
良性サンプル数	2533 (SO)	**2220 (VT)**
悪性サンプル数	-	1150 (VT)

データをテストサンプルとし，その検知率を検証する．

先行研究では，テストサンプルは訓練サンプルのデータセットと同一のデータセットから収集された．同一のデータセットでは，マクロに出現する単語が類似するなど，ある特定のマクロの分類に特化している場合がある．そこで二つ目の検証実験では，提案手法の汎化性を検証するために，訓練サンプルと異なるデータセットでテストサンプルを構成する．そして，そのテストサンプルを分類して検知率を検証する．

5.2.1　不均衡データを対象とした分類

一つ目の検証実験では，不均衡データであるテストサンプルに対して提案手法を用いて分類を行い，その検知率を検証する．本検証実験の条件を表 6 に示す．訓練サンプルは 2015 年度にアップロードされたＳＯのマクロとする．テストサンプルの良性サンプルは，SO および VT から収集した良性サンプルを合わせたものとする．テストサンプルの悪性サンプルは，ＶＴから収集した悪性サンプルとする．テストサンプルの良性サンプル数が 11415 個に対して，悪性サンプル数は 641 個であり，良性サンプルの方が極めて数が多い．不均衡データを用いることにより，実践的な環境での精度を評価する．また，テストサンプルよりも過去にアップロードされた検体を訓練サンプルとすることにより，未知のマクロに対する効果を検証する．文書トピックベクトルの次元数を 100 ずつ増加させて分類を行い，Ｆ値で検知率を評価する．

5.2.2　異なるデータセットを対象とした分類

次に，訓練サンプルと異なる収集元のテストサンプルのみを用いて精度を評価する．この実験では，テストサンプルに出現する単語の傾向は，訓練サンプルとは異なる可能性がある．本検証実験の条件を表 7 に示す．不均衡データに対して行った検証実験では，テストサンプルがＳＯとＶＴの２つの収集元から構成されていた．これに対し本実験では，テストサンプルに VT にアップロードされたサンプルのみを用いる．訓練サンプルは，2015 年度にアップロードされたＳＯのサンプルを使用する．尚，表 7 では，表 6 と異なる部分が太字で示されている．文書トピックベクトルの次元数を不均衡データに対して行った検証実験と同様の手法で変化させる．テストサンプルの分類後，検知率をＦ値で評価する．

5.3　実験結果

不均衡データを対象とした検証実験の結果を図 2 に示す．(a) と (b) は，それぞれ 2016 年度のテストサンプルを分類した場合の実験結果と，2017 年度のテストサンプルを分類した場合の実験結果を示している．図 2 の縦軸はＦ値を示し，横軸は文書トピックベクトルの次元数を示している．ＯＣＳＶＭでは，次元数が大きくなるにつれてＦ値が上昇する傾向が (a) と (b) において確認できた．ＬＯＦでは，次元数に関係なく，Ｆ値が高い状態を維持することが (a) と (b) において確認できた．この結果から，ＬＯＦはＯＣＳＶＭよりも安定的に高い検知率で悪性マクロを検知できるといえる．また，ＬＯＦはテストサンプルがアップロードされた期間に関係なく，高い検知率で検知できた．よって，この実験結果から，提案手法は不均衡データにおいて悪性マクロを極めて正確に検知できることが示された．

異なるデータセットを対象とした検証実験の結果を図 3 に示す．(a) と (b) は，それぞれ 2016 年度のテストサンプルと 2017 年度のテストサンプルを分類した場合の結果を示している．図 3 の縦軸はＦ値を示し，横軸は文書トピックベクトルの次元数を示している．(a) および (b) から，ＯＣＳＶＭでは，文書トピックベクトルの次元数が大きくなるにつれてＦ値が増加していることが確認できる．また，ＬＯＦでは，次元数に関係なく高いＦ値を得ていることが確認できる．よって，この実験結果から，提案手法はデータセットに関わりなく，極めて正確にテストサンプルを分類できることが示された．

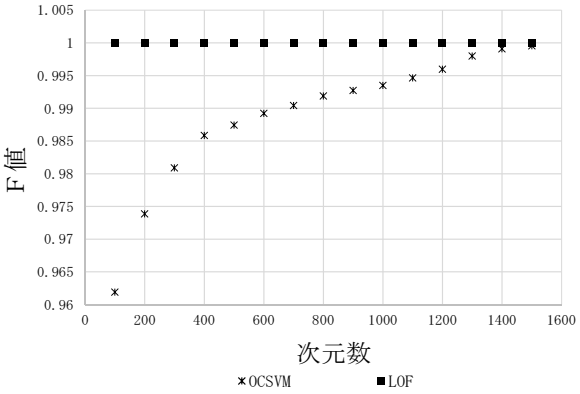
図 2 および図 3 の結果から，共通してＬＯＦはＯＣＳＶＭよりも高い検知率で未知の悪性マクロを検知できることが確認された．このことから，ＬＯＦの方がＯＣＳＶＭよりも未知の悪性マクロの検知に効果的であることが確認された．

6.　考察

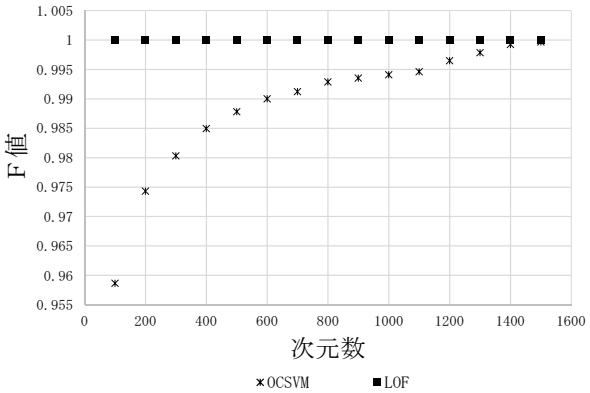
6.1　正確性

一般的に，メールで受信される悪性マクロの割合は良性マクロに比べて少ないものと考えられる．そのため，現実的には，テストサンプルは不均衡データになるものと考えられる．検証実験では不均衡データを極めて正確に分類した．したがって，提案手法の正確性は高いと考えられる．

実験では，SO のデータセットを訓練サンプルとし，VT のデータセットをテストサンプルとして分類を行った．訓

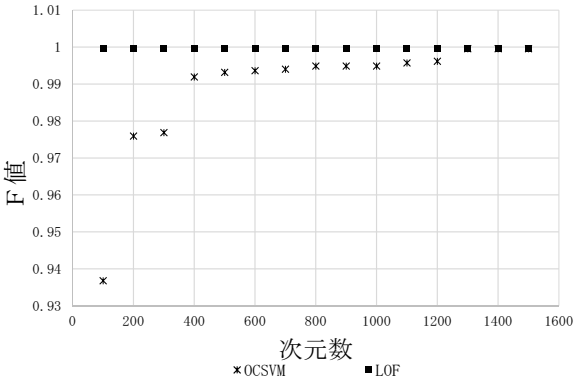


(a) 訓練：2015年度（ＳＯ）　テスト：2016年度（ＳＯ+ＶＴ）

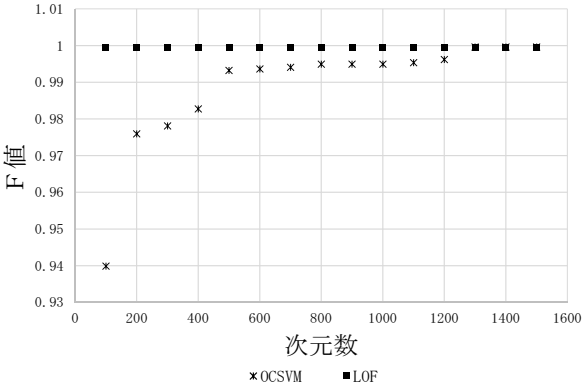


(b) 訓練：2015年度（ＳＯ）　テスト：2017年度（ＳＯ+ＶＴ）

図 2　実験手法 1 における実験結果



(a) 訓練：2015年度（ＳＯ）　テスト：2016年度（ＶＴ）



(b) 訓練：2015年度（ＳＯ）　テスト：2017年度（ＶＴ）

図 3　実験手法 2 における実験結果

練サンプルとテストサンプルは収集元が異なるため，出現する単語の傾向も異なるものであると考えられるが，高い検知率で悪性マクロを検知することができた．したがって，提案手法は汎化性が高いと考えられる．

6.2　誤検知の原因

検証実験において誤検知したマクロは，すべて良性マクロであった．誤検知した良性マクロの単語は，16 進数を演算する処理が多く含まれていた．悪性マクロの単語には，ＡＳＣＩＩコード等の 16 進数が多く含まれているため，誤検知した良性マクロの単語と類似する．よって，悪性マクロに類似する文書トピックベクトルがこの良性マクロから抽出されたため，この良性マクロは誤検知されと考えられる．しかしながらこれは，悪性マクロの見逃しではないため，深刻なインシデントにつながる可能性は低いものと考えられる．

6.3　従来手法との比較

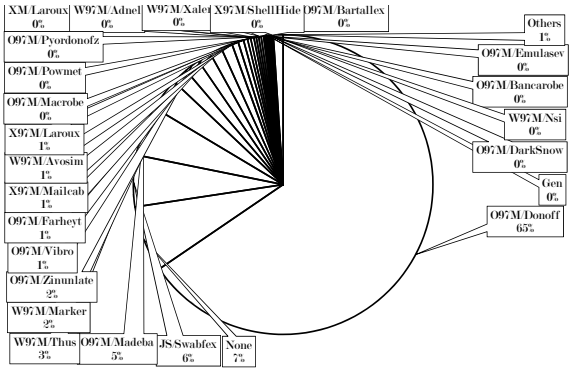
先行研究 [3] では，悪性マクロと良性マクロの両方を分類器の訓練に必要とした．一般的に悪性マクロの収集は難しいため，先行研究には運用上の課題がある．提案手法は，収集が容易な良性マクロしか必要としない．よって，提案手法は既存の手法よりも運用性が高いと考えられる．

先行研究で行った検証実験では，悪性マクロと良性マクロの比が概ね均衡のテストサンプルを分類した．その結果，Ｆ値は 0.89 であった．提案手法では，不均衡データであるテストサンプルであっても，安定的に 0.99 のＦ値を得ることができる．そのため，既存の手法よりも，提案手法は未知の悪性マクロの検知に効果的である．

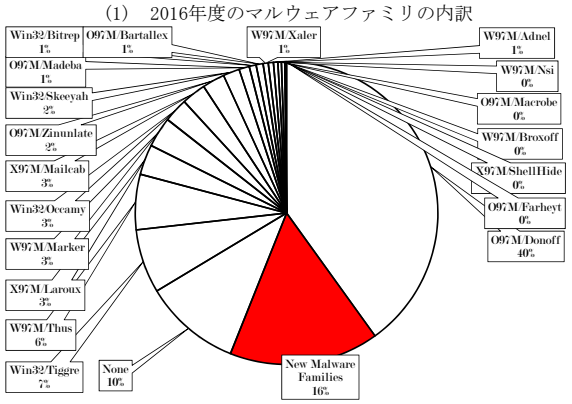
パターンマッチング方式のウイルス対策ソフトは，パターンファイルの定期的な更新を必要とする．検証実験では，2015 年度のサンプルを LOF に学習させることにより，2016 年度または 2017 年度のテストサンプルを極めて正確に分類することができた．検証実験の結果から，提案手法は少なくとも 2 年間の訓練サンプルの更新を必要としないことが確認できた．さらに，パターンマッチング方式のウイルス対策ソフトは，未知の悪性マクロを検知すること困難であるが，提案手法では可能と考えられる．

6.4　新種のマルウェアファミリの検知

図 4 に，表 4 に示す悪性マクロのマルウェアファミリの内訳を示す．ダウンローダ型悪性マクロの代表的なマルウェアファミリは `097M/Donoff` と `X97/Donoff` であり，ドロッパ型悪性マクロの代表的なマルウェアファミリは



(1)　2016年度のマルウェアファミリの内訳



(2)　2017年度のマルウェアファミリの内訳

図 4　各年度の悪性ＭＳ文書ファイルにおけるマルウェアファミリの内訳

XM/Laroux と 097M/Agent であった．2017 年度のテストサンプルにおいて，2016 年度の悪性マクロには存在しない新種のマルウェアファミリを，円グラフ (2) では色付きで示した．2017 年度の悪性サンプルでは，新種のマルウェアファミリが全体の 16%を占めた．新種のマルウェアファミリは 37 種類存在した．ＬＯＦを用いた場合，新種のマルウェアファミリを全て検知できた．この結果から，提案手法は新種のマルウェアファミリを高い精度で検知できることが確認できた．

6.5　倫理性

本稿の検証実験で用いた訓練サンプルは，ＳＯから，クローラを用いて収集した．訓練サンプルの収集では，ＳＯのウェブサーバにクローリングによる過度なアクセス負荷をかけないよう，適切な待機時間を設けて行った．そのため，本研究は公共性を重んじて行われたと考えられる．

提案手法は，特別に演算性能の高い端末を用いる必要はなく，一般的な演算性能の端末で動作することができる．そのため，提案手法は導入が容易であるため，公益性が高いと考えられる．

本稿で用いた訓練サンプルは，クローラを用いてＳＯのマクロを収集したものである．そのため，訓練サンプルの収集は，特別なコストを必要としないため，容易である．また，提案手法は，gensim や scikit learn が公開するモジュールのみを用いるため，実装が容易である．したがって，提案手法の再現性は高いと考えられる．

7.　おわりに

本稿では，良性マクロのみを分類器に学習させることによって，悪性マクロを異常検知する手法を提案した．検証実験では，提案手法を用いて 2016 年度および 2017 年度にアップロードされたテストサンプルを分類し，検知率を評価した．その結果，それぞれのテストサンプルで 0.99 のＦ値を得ることができただけでなく，新種のマルウェアファミリを全て検知できた．

今後の課題は，悪性な Javascript 等の他のプログラミング言語で記述されたマルウェアに対して提案手法を適用し，その検知率を検証することである．

参考文献

[1] 羊の皮をかぶったオオカミ:ＶＢＡを使用したマルウェアに関する調査報告, https://nakedsecurity.sophos.com/ja/2017/05/31/wolf-in-sheeps-clothing-a-sophoslabs-investigation-into-delivering-malware-via-vba/
[2] Y.Otsubo, M.Mimura, H.Tanaka :O-checker: Detection of Malicious Documents through Deviation from File Format Specifcation, Black Hat USA 2016 (2016)
[3] H.Miura, M.Mamoru, T.Hidema : Macros Finder: Do You Rememmber LOVELETTER?, 14th International Conference on Security Practice and Eexperience, 印刷中
[4] Nir, N., Aviad, C., Yuval, E. :ALDOCX: Detection of Unknown Malicious Microsoft Office Documents Using Designated Active Learning Methods Based on New Structural Feature Extraction Methodology,IEEE Transactions on Information Forensics and Security volume 12,Issue:3,pp.631-646,(2017)
[5] A,Naser, A,Hadi :Analyzing and Detecting Malicious Content: DOCX Files, International Journal of Computer Science and Information Security (IJCSIS), volume 14, Issue 8, pp.404-412, (2016)
[6] Boldewin, F. :Analyzing MSOffice malware with OfficeMalScanner, https://ja.scribd.com/document/21143233/Analyzing-MSOffice-Malware-WithOfficeMalScanner
[7] M.Mimura, Y.Otsubo, H.Tanaka :Evaluation of a Brute Forcing Tool that Extracts the RAT from a Malicious Document File, 2016 11th Asia Joint Conference on Information Security (Asia JCIS), DOI 10.1109
[8] teratail, https://teratail.com/
[9] olevba (https://github.com/decalage2/oletools/wiki/olevba)
[10] scikit-learn, http://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html
[11] Anomaly detection with Local Outlier Factor (LOF), http://scikit-learn.org/stable/auto_examples/neighbors/plot_lof.html
[12] python package index scikit learn 0.19.0,https://pypi.python.org/pypi/scikitlearn/0.19.0
[13] python package index gensim 0.10.1,https://pypi.python.org/pypi/gensim/0.10.1
[14] Virus Total (https://stackoverflow.com/)
[15] stackoverflow (https://stackoverflow.com/)