Vol.2017-CSEC-79 No.8
2017/12/5

One-Class SVM を用いたマルウェア PDF 検出手法の改良

岩本 舞[1,a)]　小島 俊輔[1]　中嶋 卓雄[2]

概要：一般に PDF は静的な文書形式と認識されており，マルウェアとしての危険性は広く認知されていない．しかし実際には，PDF ファイルは JavaScript コードや外部プログラムの実行，URL アクセスなどを行うことができ，攻撃に利用されている．本研究では，One-Class SVM で正常な PDF から取り出した値を要素とするベクトルを学習し，外れ値に分類されるベクトルを持つファイルをマルウェアとして判別する．これまでの研究では，SVM のベクトル要素として，それぞれの特徴の有無を示す二値のデータを利用していた．本稿では，特徴に関係するデータ量や特徴の個数を利用し，また新たな特徴量を加えることで，検出精度の向上を図る．実験の結果，従来手法に比べ，提案手法ではより広い範囲のパラメタで Accuracy が向上し，評価用正常 PDF 7189 個中 7182 個，マルウェア PDF 221 個中 218 個を正しく検出することができた．

Mai Iwamoto[1,a)]　Shunsuke Oshima[1]　Takuo Nakashima[2]

1. はじめに

Portable Document Format [1]（以下 PDF）は異なる環境下においても高い再現性を持つために，全世界で広く利用されているファイル形式である．PDF ファイルは実行ファイル (.exe) やマクロファイル (MS Office) 等と異なり，一見すると静的文書に見えること，また普段からメール添付や Web などで頻繁にやりとりされることから，警戒せずに安易に開いてしまうユーザも多い．しかし実際には，PDF は JavaScript コードや外部プログラムの実行，URL アクセスなど動的に機能を実行できる形式であり，攻撃に利用される場合がある．ユーザを警戒させずに攻撃を行えるという意味では，攻撃者にとって有用性の高いファイル形式である．

従来研究において，マルウェアの検知には機械学習が用いられている．機械学習は教師あり学習と教師なし学習に分類される．教師あり学習では，分類したいクラスそれぞれについて教師となる学習用データを用意する必要があるが，次々に亜種や新型が誕生するマルウェアの世界では，常に最新の教師データを収集し続けることが難しい．そこで我々は，One-Class Support Vector Machine[2]（以下 One-Class SVM）を用いた，教師なし学習によるマルウェア検知手法を提案する．この手法では，インターネット上で容易に手に入る正常 PDF データを教師として分類器を作成し，外れ値となったファイルをマルウェアとして検出する．学習用に大量のマルウェアを用意する必要がないため，比較的分類器の作成が容易であるという特長がある．

2. 従来研究

文献 [3] では，マルウェアの脅威に関して広く説明が行われている．文献 [4] では，PDF をどのように悪用できるか，具体的な例を用いて説明している．また文献 [5] では，攻撃者に利用されうる危険な機能の紹介と分類を行い，PDF を利用したフィッシングや二段階攻撃等について，危険性を評価している．

マルウェア解析手法は，静的解析と動的解析に分類される．静的解析は，データの取り出しや解析にファイルの実行を伴わない方法である．動的解析には，コードを実行してデータを取り出す手法と，さらに取り出したデータを実行し挙動を解析する手法がある．

静的解析には，実行されるコードの解析を行うものや，メタデータの特徴から正常なファイルとマルウェアを判別す

[1] 熊本高等専門学校
Yatsushiro, Kumamoto 866-8501, Japan
[2] 東海大学
Toroku, Kumamoto 862-8652, Japan
[a)] m-iwamoto@kumamoto-nct.ac.jp

る手法がある．文献 [6] は，PDF 中の JavaScript を静的に取り出しトークン化したものをベクトルとして One-Class SVM で学習し，正常 PDF とマルウェア PDF の違いを判別するモデルを作成する．文献 [7] は PDFMS [8] というツールを提案している．PDFMS は機械学習に基づくマルウェア検出ツールで，PDF ファイルに出現するキーワードを正常セットとマルウェアセットに分類してクラスタリングし，機械学習により正常な PDF とマルウェア PDF を判別している．文献 [7] では，学習手法として Complement Naive Bayes, SVM Linear Kernel, J48 Decision Tree および Random Forest classifier を比較しており，PDFMS にはもっとも結果の良かった Random Forest classifier が用いられている．PDFMS は，本手法と同じ静的解析のツールであること，また筆者によりツールが提供されていることから，実験で本手法との比較対象とした．文献 [9] は PDF 文書の階層的な構造から正常 PDF とマルウェア PDF を判別する．階層構造の学習には SVM および Decision Tree を用いている．文献 [10] では，`/Font`，`/JavaScript`，`/JS` といった Name が出現する回数など，202 個の PDF の特徴を Random Forests で学習し，分類を行っている．

筆者らの従来研究 [11] では，マルウェアの特徴を含むか否かのフラグを主とするベクトルを One-Class SVM で学習し，外れ値によりマルウェアを検出する手法を提案した．本手法は，解析にファイルの実行を行わないため，静的解析に分類される．また，分類器の作成に正常 PDF のみを必要とし，マルウェア PDF を必要としない，教師なし学習の手法である．本稿では，単にマルウェアの特徴の有無だけでなく，それぞれの特徴量をベクトルに反映するとともに，新たな特徴を追加し， Accuracy の向上を図る．

3. PDF ファイルフォーマット

PDF の仕様は ISO 32000-1:2008 として公開されている．ここでは，**図 1** を用いて，PDF の仕組みを簡単に説明する．

PDF では，すべての情報は Boolean（真偽値），Numeric（数値），String（文字列），Name（名前），Array（配列），Dictionary（辞書），Stream（ストリーム），Null の 8 つの基本的なオブジェクトの組み合わせで記述される．名前は，/から始まる任意の文字列であり，ファイル内で一意に定義される．配列は，[ ] 内に複数のオブジェクトを線形に並べたものである．辞書は，<< >>で囲まれた，名前オブジェクト（キー）と任意のオブジェクト（値）のペアの一覧である．ストリームは，辞書および `stream` と `endstream` で囲まれたバイト列からなる．バイト列は任意の形式で圧縮しておくことができ，読み込み時に辞書に記述された圧縮方式を参照してデコードする．

任意のオブジェクトに対し，他のオブジェクトから参照するために，ユニークなオブジェクト識別子を割り振ることができる．識別子を割り振られたオブジェクトを間接オブジェクトといい，`X Y obj` と `endobj` で囲むことで記述する．`X` はオブジェクト番号，`Y` は世代番号である．

```
%PDF-1.7
1 0 obj
<< /Type /Catalog /Pages 2 0 R /OpenAction 5 0 R >>
endobj
2 0 obj
<< /Type /Pages /Kids [3 0 R] /Count 1 >>
endobj
3 0 obj
<< /Type /Page /Parent 2 0 R /MediaBox [ 0 0 595 842 ]
   /Contents 4 0 R >>
endobj
4 0 obj
<< /Length 74 >>
stream
q
10 0 0 10 297 421 cm
0.8 0 0 rg
 :
Q
endstream
endobj
5 0 obj
<< /Type /Action /S /JavaScript /JS 6 0 R >>
endobj
6 0 obj
<< /Length 25 >>
stream
app.alert("Hello World!");
endstream
endobj
7 0 obj
<< /Title (TitleField) /Author (AuthorField)
   /Creator (CreatorField) /Producer (ProducerField) >>
endobj
xref
0 7
0000000000 65535 f
0000000009 00000 n
 :
0000000484 00000 n
trailer
<< /Size 7 /Root 1 0 R /Info 7 0 R >>
startxref
597
%%EOF
```

図 1 PDF ファイルの例

PDF は，`trailer` と呼ばれる辞書オブジェクトを起点に間接オブジェクトを参照していき，ページの構成要素を取得する．図 1 では，トレイラーは，オブジェクト 1 を参照している．オブジェクト 1 は，PDF ファイル全体の構成に必要な情報を格納しており，ページの一覧に関する情報としてオブジェクト 2 を参照している．オブジェクト 2 は，PDF ファイルのページ一覧情報を格納しており，ページとしてオブジェクト 3 を参照している．オブジェクト 3 は，ページの描画サイズを格納しており，ページの内容としてオブジェクト 4 を参照している．オブジェクト 4 には，実際にページに表示される内容が記述されている．オブジェクト 1 が`/OpenAction` の値としてオブジェクト 5 を参照すると，オブジェクト 6 をソースコードとして JavaScript が実行される．オブジェクト 7 には，PDF ファイルのプロパティ情報が格納されている．

PDF ファイルは，暗号化することで，閲覧・印刷・修正等に制限をかけることが可能である．PDF の暗号化は，

ファイル全体を暗号化するのではなく，実際のページの内容を示すストリームのバイト列（図 1 におけるオブジェクト 4 の `stream` から `endstream` で囲まれた部分）のみが暗号化の対象となる．このため，PDF ファイルが暗号化されている場合でも，ファイル構造は取り出すことができる．

4. One-class SVM

一般的に，Support Vector Machine [12]（以下 SVM）は，+1 と −1 の 2 クラスのベクトルを教師あり学習し，与えられたベクトルがどちらのクラスに属するか分類を行う．一方，One-class SVM では +1 のクラスに属するベクトルのみを学習し，与えられたベクトルが学習したクラスに属す (+1) か属さない (−1) かを判定する．ここでは正常な PDF ファイルの特徴量ベクトルをあらかじめ +1 のクラスとして学習しておき，−1 のクラスに分類された PDF ファイルをマルウェアであると判定する．

一般に SVM では 2 つのクラスを超平面で線形分離するが，単純に線形分離可能であるケースはまれである．そこで，ある学習用データ $x_1, \ldots, x_l \in X$ が与えられたとき，X を非線形の写像 $\Phi(x)$ によって高次元の特徴空間 F に写像し，写像先の空間 F で線形分離を行う．

ここで，あるベクトルが超平面のどちら側にあるか識別する関数 $f(x) = \mathrm{sgn}(w \cdot \Phi(x) - \rho)$ を考える．$\mathrm{sgn}(x)$ は x が正の値なら +1，負の値なら −1 を返す関数である．One-Class SVM は，原点と学習データを分離する超平面の，原点からの距離を最大化することを目的とする．そこで，$||w||$ が小さくなるように，また同時に多くの学習用データで $f(x) = +1$ になり，外れ値では $f(x) = -1$ になるように，最適な w および ρ を設定する．

上記の 2 つの問題を解決するために，(1) の最小化問題を解く．

$$\min_{w \in F, \xi \in \mathbb{R}^l, \rho \in \mathbb{R}} \frac{1}{2}||w||^2 + \frac{1}{\nu l}\sum_i \xi_i - \rho \qquad (1)$$

subject to

$$(w \cdot \Phi(x_i)) \geq \rho - \xi_i,\ \xi_i \geq 0\ (i = 1 \ldots l) \qquad (2)$$

ξ_i は超平面の反対側に入り込んでしまった 学習用データ x_i のエラー距離を示すパラメタであり，$1/(\nu l)$ で重みづけされる．パラメタ $\nu \in (0, 1)$ はトレードオフの関係にある $||w||$ の最小化および $\sum \xi_i$ の最小化の 2 つの問題をコントロールするために設定する値である．

(1)(2) から導いた ラグランジュ関数を解くと，以下の双対問題に帰結する．

$$\min_{\alpha} \frac{1}{2}\sum_{ij} \alpha_i \alpha_j \Phi(x_i) \cdot \Phi(x_j) \qquad (3)$$

subject to

$$0 \leq \alpha_i \leq \frac{1}{\nu l},\ \sum_i \alpha_i = 1 \qquad (4)$$

しかし (3) を解くにあたって，すべての学習用データを $\Phi(x)$ で写像すると計算量が膨大となってしまう．実際に必要となるのは $\Phi(x)$ ではなく $\Phi(x_i) \cdot \Phi(x_j)$ であるため，SVM では写像関数 $\Phi(x)$ を計算する代わりに $k(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$ となるようなカーネル関数 $k(x_i, x_j)$ を利用する．カーネル関数には一般に線形関数，多項式関数，RBF (Radial Basis Function) 関数，シグモイド関数などが用いられるが，今回の実験ではもっとも一般的に用いられる RBF 関数 $k(x_i, x_j) = e^{-||x_i - x_j||^2/c}$ を使用した．

5. 提案手法

筆者らの従来研究 [11] では，マルウェアの特徴を含むか否かのフラグを主とする 8 次元のベクトルを One-Class SVM で学習し，外れ値によりマルウェアを検出する手法を提案した．本稿では，単にマルウェアの特徴の有無だけでなく，それぞれの特徴に関係するデータ量や特徴の個数をベクトルに反映するとともに，新たな特徴を追加し，Accuracy の向上を図る．

ベクトルの作成にあたり，SVM では，各ベクトルの値を 0〜1 に正規化する必要があるため，今回は，特徴量に応じ**表 1** に示す正規化を行った．提案手法でベクトルに使用した 12 の特徴量を**表 2** に示す．

実験にあたっては，自作の PDF 構文解析プログラムを使用し，xref テーブルは使わずにファイルの先頭から解析を行った．圧縮されたストリームの復号化には `/FlateDecode`, `/ASCIIHexDecode`, `/ASCII85Decode`, `/LZWDecode`, `/RunLengthDecode` 用のデコーダーを用意した．その他の主に画像に対して利用される圧縮形式（JPEG 圧縮に利用される `/DCTDecode` など）については，画像用の圧縮形式であるためコードを埋め込むことは困難であると判断し，解析しない．また，ファイルに文法的なエラーが存在する場合には，その部分を無視して可能な限り解析するようにした．

なお今回の実験では，JavaScript の構文解析はせず，表 2 の (2)(3)(4) の特徴量である `this.info`, `this.getAnnot`, `eval` に対しては，前後にアルファベットを含まないという条件のみで単純な文字列マッチングを行った．

6. 実験

6.1 データセット

今回の実験では，正常 PDF として，熊本高等専門学校八代キャンパスに設置されたプロキシサーバで収集した PDF ファイルを使用した．2015 年 6 月から 2016 年 12 月

*1 実際のファイルではストリームは圧縮された状態だが，図では内容が分かるようデコード後のストリームを示している

表 1 正規化の種類

正規化の種類	説明
Flag	特徴の有無を表す二値の値 (0, 1)
nLength (normalized length)	文字列長 N byte を $\min(1, (\log_{10} \mathrm{N})/8)$ により 0〜1 に正規化した値．PDF のファイルサイズは 100MB 程度までのものが多いため，今回は 100MB を上限として正規化した．
nNum (normalized number)	数値 N を $\min(1, N/10)$ により 0〜1 に正規化した値

表 2 One-Class SVM のベクトルに使用した特徴量

特徴量	正規化の種類	説明
(1) JavaScript コード	nLength	PDF 中の JavaScript コード長の合計．PDF では，JavaScirpt のソースコードは`/JS` で指定される．そこで，`/JS` で指定された文字列・ストリーム（デコード後）の長さの合計値を特徴量とする．
(2) `this.info`（JavaScript）	Flag	JavaScript コード中の `this.info` の有無．PDF JavaScript では，`this.info` で PDF のプロパティ情報 (trailer の`/Info` で指定された辞書に格納されている情報) を取得できる．マルウェアの中には，JavaScript コード長さを減少させ検出者の目を欺くために，**図 2** のようにプロパティ情報にソースコード本体や難読化したコードを仕込み，`this.info` を使って読み込む手口を利用したものがある．
(3) `this.getAnnot`（JavaScript）	Flag	JavaScript コード中の `this.getAnnot` の有無．PDF JavaScript では，`this.getAnnot` で注釈情報を取得できる．この機能は，`this.info` と同様に，JavaScript コード長を減少させ検出者の目を欺く手口に利用されている．
(4) `eval`（JavaScript）	Flag	JavaScript コード中の `eval` の有無．`eval(`*str*`)` は *str* を JavaScript コードとみなして実行する関数であり，図 2 のように，コードの難読化に利用される．
(5) 不正な `/Length 0`	Flag	実際にはデータがあるにもかかわらず，辞書に`/Length 0` と記述されているストリームの有無．不正な `/Length 0` は，攻撃者によるコードの隠ぺい工作であると考えられる．
(6) 非暗号化ファイルのストリームデコードエラー	nLength	指定された形式でのデコードに失敗したストリーム長の合計．ストリームは圧縮されている場合があり，圧縮形式は`/Filter` で指定される．ファイルが暗号化されていないにもかかわらず，指定された圧縮形式でデコードできない場合，攻撃者によって何らかの細工が施されている可能性がある．
(7) application/x-javascript	Flag	ストリーム中の application/x-javascript の記述の有無．PDF1.5 以降では，AdobeXFA（XML Forms Architecture）ベースの対話型フォームがサポートされている．XFA ベースの場合，JavaScript を動作させるためにリソース中で application/x-javascript が指定される．
(8) `%%EOF` なし	Flag	End-Of-File コメント `%%EOF` の有無．仕様により PDF ファイルは`%%EOF` で終わることになっている．一般に PDF ファイルはソフトウェアによって作成されるものであり，ユーザが直接コードを変更することはないため，正常な PDF ファイルのほとんどはこのような仕様に則っているが，マルウェア PDF ファイルには基本的な仕様に従っていないものが見られた．
(9) `%%EOF` 後のデータ長	nLength	最後の `%%EOF` の後にあるデータの長さ．マルウェアの中には，`%%EOF` の後に JavaScript コードの読み込みなどのソースを埋め込んだものが見られた．
(10) シンタックスエラー数	nNum	pdfinfo[13] で PDF を解析した際のシンタックスエラーのうち，正常 PDF で見られた “Missing orinvalid default viewing OCCD”, “Unknown filter ’Crypt’ ” 以外のエラーの数．
(11) `/Info` のデータ長	nLength	`/Info` オブジェクトに直接記述または `/Info` オブジェクトから参照された文字列やストリームの長さの合計．通常 `/Info` に含まれるのは筆者名や作成日時などの情報であるが，`this.info` で難読化した JavaScript コードを読み込む場合，`/Info` に含まれる文字列が通常に比べ長くなる傾向にある．
(12) ファイルサイズ	nLength	PDF ファイルのサイズ．

の間にプロキシサーバを通じてアクセスがあったファイルで，現在も取得可能かつすでに VirusTotal[14] に登録されており，すべてのアンチウイルスソフトでマルウェアとして検出されなかった 7189 個を用いた．収集の時期を 1 年前に設定し，すでに VirusTotal に登録されていたファイルのみを正常 PDF として使用した理由は，新しいファイルや，VirusTotal に登録のないファイルを使用した場合，解析が終了しておらず，マルウェアであってもアンチウイルスソフトで検出できない可能性があることを考慮したためである．

マルウェア PDF としては，2010 年から 2015 年の間に収集された D3M データセット [15] に含まれる PDF ファ

```
5 0 obj
<< /Filter /FlateDecode /Length 3171 >>
stream
z117f188p716v57666z117f188p716v57675z117f188p716v5766ez
117f188p716v57663z117f188p716v57674z117f188p716v57669z1
 :
endstream
endobj
6 0 obj
<< /Filter /FlateDecode /Length 151 >>
stream
function d19s47h56h03(){z46h28l81 = this.info.title;var
 h9t7u9j2 = "";var j3u6c7 = "%";h9t7u9j2 = z46h28l81.re
place(/z117f188p716v576/g,j3u6c7);var n9s1f9r4 = unesca
pe(h9t7u9j2);eval(n9s1f9r4);} d19s47h56h03();
endstream
endobj
7 0 obj
<< /JS 6 0 R /S /JavaScript >>
endobj
9 0 obj
<< /Author (HeirsUvulaHints) /CreationDate (D:200902132
23739) /Creator (Adobe) /Producer (EmEditor) /Title 5 0
 R >>
endobj
```

図 2 this.info と eval を利用した攻撃例 *1

イル 349 個のうち，VirusTotal にマルウェアとして登録されていた 221 個を用いた．実験に用いたファイルはすべてユニークであり，重複するものはない．同じマルウェア PDF が複数年にわたって収集されている場合は，一番古い年に繰り入れている．

6.2 実験手法

本手法は，学習用データに存在する正常 PDF ファイルの特徴ベクトルを +1 クラスとする One-Class SVM の分類器を作成し，評価用データのベクトルを分類し，+1 クラスに分類されれば正常 PDF，−1 クラスに分類されればマルウェア PDF と判定されたとみなす．

評価には，一般に SVM の評価に用いられる K-fold Cross Validation を用い，$K = 4$ で実験を行った．4 つのサブセットには，それぞれ学習用正常 PDF 5392 個，評価用正常 PDF 1797 個，評価用マルウェア PDF 55 個が含まれており，サブセット内でのファイルの重複はない．また，どのファイルも全サブセットを通して一度ずつ評価が行われる．

なお，実験には LIBSVM[16] に実装された One-Class SVM を使用し，カーネル関数は一般的に利用される RBF カーネル $k(x_i, x_j) = e^{-||x_i - x_j||^2/c}$ を用いた．

6.3 パラメタの選定および従来手法との比較

SVM の性能はパラメタに大きく左右される．そこでまず，One-Class SVM に必要なパラメタ c および ν について，それぞれ $c = 2^n$，$\nu = 2^m$ とし，n を −10 から 10 まで 2 刻み，m を −13 から −3 まで 1 刻み（$2^{-13} \simeq 0.0001$，$2^{-3} \simeq 0.1$）で変化させ，正常 PDF，マルウェア PDF および双方の Accuracy について，4 回の実験の平均を求め，最適なパラメタを決定する．

さらに，本稿では，筆者らによる従来手法 [11] と提案手法を同じデータセットを用いて比較することにより，提案手法によって Accuracy が改善していることを示す．

表 3 に従来手法での結果，**表 4** に提案手法の結果を示す．表の背景が薄いグレーの部分は Accuracy が正規分布における 2σ，濃いグレーの部分は 3σ 以上であることを意味する．なお，正常 PDF ファイル 7189 個のうち，プログラムで解析ができなかった 2 個については， Accuracy の計算には含めていない．

表 3(a) と**表 4(a)** を比較すると，従来手法では Benign Accuracy が 3σ 以上となったパラメタは $2^2 \leq c \leq 2^8$，$2^{-13} \leq \nu \leq 2^{-10}$ であったが，提案手法では $2^{-2} \leq c \leq 2^6$，$2^{-13} \leq \nu \leq 2^{-9}$ と，より広い範囲のパラメタで高くなっていることわかる．

一方，**表 3(b)** と**表 4(b)** を比較すると，従来手法の Malicious Accuracy は，$c \leq 2^{-4}$，$\nu \geq 2^{-8}$ の範囲では 1.0 であるが，この範囲では Benign Accuracy が 3σ 未満であり，多くの正常 PDF をマルウェアとして検出してしまう．逆に同手法で Benign Accuracy が 3σ 以上となる $2^2 \leq c \leq 2^8$ かつ $\nu \leq 2^{-10}$ の範囲では Malicious Accuracy が 2σ 未満になり，十分に検出できない．一方，提案手法では Malicious Accuracy はすべてのパラメタで 2σ 以上であり，正常 PDF とマルウェア PDF の双方を正しく検出できる．なお，従来手法で Malicious Accuracy が 1.0 であるにもかかわらず，提案手法で 1.0 未満になっているパラメタについて調べたところ，2 つのマルウェア PDF において，JavaScript が使用されていたが，ソースコードとして指定されているストリームオブジェクトがデコードエラーになり，ソースコード長が 0 とみなされていた．そのため，単純に`/JavaScript` の有無を特徴とした従来手法では検出されたが，ソースコード長を特徴とした提案手法では検出されなかった．デコードできなかったストリームには，何らかの細工がされている可能性があるが，解析については今後の課題とする．

また，評価用データ全体の Accuracy を示す**表 3(c)** と**表 4(c)** を比較すると，従来手法では，3σ となるパラメタは $2^4 \leq c \leq 2^8$ かつ $\nu = 2^{-9}$ の 3 か所しかないが，提案手法では，$2^{-2} \leq c \leq 2^6$，$2^{-13} \leq \nu \leq 2^{-9}$ と，3σ 以上の高い Accuracy となる範囲が広くなった．より広い範囲のパラメタで Accuracy が高くなったことで，多少データセットが変化して最適なパラメタが変わっても高い Accuracy を見込むことができる．

6.4 他手法との比較

比較実験には，既存手法として PDFMS[7][8] を用いた．PDFMS を比較に用いた理由は，提案手法と同じ静的解析手法であり，また筆者によりツールが提供されているためである．PDFMS は Random Forest Classifier を用いた教

表 3 従来手法におけるパラメタによる Accuracy の変化
(a) Benign Accuracy

$n \backslash m$	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3
-10	0.5526	0.6401	0.7076	0.6909	0.7013	0.6901	0.7502	0.7306	0.7189	0.6826	0.6755
-8	0.6227	0.6671	0.6865	0.6634	0.7387	0.7374	0.7435	0.7387	0.8436	0.8709	0.8506
-6	0.5987	0.7224	0.7011	0.7786	0.8613	0.8735	0.9115	0.9308	0.9498	0.9318	0.8799
-4	0.7015	0.8093	0.9057	0.9439	0.9528	0.9676	0.9773	0.9677	0.9645	0.9361	0.8745
-2	0.9969	0.9968	0.9965	0.9965	0.9965	0.9951	0.9912	0.9846	0.9677	0.9366	0.8748
0	0.9974	0.9972	0.9976	0.9976	0.9967	0.9957	0.9914	0.9846	0.9684	0.9368	0.8748
2	0.9987	0.9986	0.9987	0.9985	0.9969	0.9960	0.9922	0.9844	0.9688	0.9386	0.8753
4	0.9992	0.9990	0.9992	0.9986	0.9972	0.9960	0.9925	0.9839	0.9684	0.9378	0.8737
6	0.9992	0.9992	0.9990	0.9990	0.9974	0.9961	0.9925	0.9840	0.9681	0.9367	0.8751
8	0.9982	0.9982	0.9987	0.9990	0.9979	0.9954	0.9925	0.9837	0.9681	0.9370	0.8746
10	0.1228	0.9498	0.9901	0.9993	0.9969	0.9964	0.9939	0.9782	0.9634	0.9322	0.8759

(b) Malicious Accuracy

$n \backslash m$	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3
-10	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
-8	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
-6	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
-4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
-2	0.9909	0.9909	0.9909	0.9909	0.9909	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0	0.9235	0.9235	0.9235	0.9235	0.9909	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
2	0.9144	0.9099	0.9144	0.9099	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
4	0.9054	0.9099	0.9054	0.9009	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
6	0.9054	0.9054	0.9054	0.9009	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
8	0.9411	0.8418	0.8416	0.9054	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
10	1.0000	0.9327	0.8691	0.8381	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

(c) Total Accuracy

$n \backslash m$	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3
-10	0.5660	0.6508	0.7163	0.7001	0.7102	0.6994	0.7577	0.7387	0.7273	0.6921	0.6852
-8	0.6339	0.6770	0.6959	0.6735	0.7465	0.7453	0.7512	0.7465	0.8483	0.8747	0.8550
-6	0.6107	0.7307	0.7100	0.7852	0.8654	0.8773	0.9141	0.9329	0.9513	0.9339	0.8835
-4	0.7104	0.8149	0.9085	0.9456	0.9542	0.9685	0.9780	0.9687	0.9656	0.9380	0.8782
-2	0.9968	0.9966	0.9964	0.9964	0.9964	0.9953	0.9915	0.9850	0.9687	0.9384	0.8785
0	0.9951	0.9950	0.9954	0.9954	0.9965	0.9958	0.9916	0.9850	0.9694	0.9387	0.8785
2	0.9962	0.9960	0.9962	0.9958	0.9970	0.9961	0.9924	0.9849	0.9698	0.9405	0.8791
4	0.9964	0.9964	0.9964	0.9957	0.9973	0.9961	0.9927	0.9843	0.9694	0.9397	0.8774
6	0.9964	0.9964	0.9962	0.9961	0.9974	0.9962	0.9927	0.9845	0.9691	0.9386	0.8788
8	0.9965	0.9935	0.9941	0.9962	0.9980	0.9955	0.9927	0.9842	0.9691	0.9389	0.8784
10	0.1490	0.9493	0.9865	0.9945	0.9970	0.9965	0.9941	0.9788	0.9645	0.9343	0.8796

師あり学習のツールであるため，本手法と異なり学習用のマルウェアが必要である．そこで 4 つの各サブセットに，本手法で用いた学習用正常 PDF 5392 個のほかに，学習用マルウェア PDF 166 個を追加する．評価については，提案手法と同様に正常 PDF 1797 個，マルウェア PDF 55 個を用いて実験を行った．従来手法，提案手法については，もっとも Accuracy が高いパラメタを用いた．

既存手法と筆者らによる従来手法 [11]，提案手法を比較した結果を**表 5** に示す．Benign Error，Malicious Error は，正常 PDF およびマルウェア PDF のそれぞれで，解析エラーとなり分類ができなかったファイル数である．PDFMS でのエラーの内容を調査すると， PDFMS 内で PDF の解析に使用している pdfminer での PDF 解析エラーが多く，全体で正常 PDF 368 個，マルウェア PDF 12 個がエラーになった．一方，本手法で解析ができなかったファイルは 2 個であった．

解析可能であったファイルについて，正常 PDF をマルウェア PDF と判定した数を見ると，PDFMS 4 個に対し本手法は 5 個である．また，マルウェア PDF を正常 PDF と判定した数では PDFMS 2 個に対し本手法では 3 個であ

表 4 提案手法におけるパラメタによる Accuracy の変化
(a) Benign Accuracy

$n \backslash m$	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3
-10	0.2602	0.3872	0.5182	0.5617	0.5880	0.6190	0.6731	0.6656	0.6812	0.6957	0.7171
-8	0.5531	0.6880	0.7781	0.7491	0.8105	0.8542	0.9009	0.9135	0.9168	0.9096	0.8604
-6	0.9500	0.9624	0.9729	0.9733	0.9779	0.9812	0.9805	0.9765	0.9626	0.9345	0.8705
-4	0.9922	0.9936	0.9921	0.9936	0.9935	0.9922	0.9901	0.9830	0.9679	0.9361	0.8744
-2	0.9982	0.9982	0.9986	0.9983	0.9975	0.9955	0.9915	0.9841	0.9684	0.9366	0.8752
0	0.9989	0.9989	0.9992	0.9986	0.9979	0.9955	0.9922	0.9843	0.9684	0.9371	0.8745
2	0.9992	0.9992	0.9992	0.9987	0.9982	0.9955	0.9918	0.9839	0.9687	0.9379	0.8738
4	0.9992	0.9992	0.9993	0.9989	0.9981	0.9960	0.9921	0.9841	0.9683	0.9382	0.8737
6	0.9993	0.9993	0.9993	0.9989	0.9981	0.9960	0.9922	0.9839	0.9681	0.9374	0.8744
8	0.9592	0.9932	0.9939	0.9987	0.9978	0.9958	0.9921	0.9840	0.9690	0.9375	0.8742
10	0.0008	0.7553	0.8147	0.9739	0.9407	0.9962	0.9853	0.9790	0.9669	0.9340	0.8724

(b) Malicious Accuracy

$n \backslash m$	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3
-10	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
-8	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
-6	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
-4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
-2	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0	0.9910	0.9910	0.9910	0.9910	0.9910	0.9910	0.9910	0.9910	0.9910	0.9955	1.0000
2	0.9864	0.9864	0.9864	0.9910	0.9910	0.9910	0.9910	0.9910	0.9910	0.9910	1.0000
4	0.9864	0.9864	0.9864	0.9910	0.9910	0.9910	0.9910	0.9910	0.9910	0.9910	1.0000
6	0.9864	0.9864	0.9864	0.9910	0.9910	0.9910	0.9910	0.9910	0.9910	0.9910	0.9955
8	0.9910	0.9637	0.9864	0.9910	0.9910	0.9910	0.9910	0.9910	0.9910	0.9910	0.9955
10	1.0000	0.9598	1.0000	0.9864	0.9910	0.9910	0.9910	0.9910	0.9910	0.9910	0.9955

(c) Total Accuracy

$n \backslash m$	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3
-10	0.2822	0.4055	0.5325	0.5748	0.6003	0.6304	0.6829	0.6756	0.6907	0.7048	0.7256
-8	0.5664	0.6974	0.7847	0.7566	0.8161	0.8585	0.9039	0.9160	0.9193	0.9123	0.8646
-6	0.9515	0.9636	0.9737	0.9741	0.9785	0.9818	0.9811	0.9772	0.9637	0.9364	0.8743
-4	0.9924	0.9938	0.9923	0.9938	0.9937	0.9924	0.9904	0.9835	0.9688	0.9380	0.8781
-2	0.9982	0.9982	0.9987	0.9984	0.9976	0.9957	0.9918	0.9846	0.9694	0.9384	0.8789
0	0.9987	0.9987	0.9989	0.9984	0.9977	0.9954	0.9922	0.9845	0.9691	0.9389	0.8782
2	0.9988	0.9988	0.9988	0.9985	0.9980	0.9954	0.9918	0.9841	0.9694	0.9395	0.8776
4	0.9988	0.9988	0.9989	0.9987	0.9978	0.9958	0.9920	0.9843	0.9690	0.9398	0.8774
6	0.9989	0.9989	0.9989	0.9987	0.9978	0.9958	0.9922	0.9841	0.9688	0.9390	0.8780
8	0.9602	0.9923	0.9937	0.9985	0.9976	0.9957	0.9920	0.9842	0.9696	0.9391	0.8778
10	0.0306	0.7614	0.8202	0.9742	0.9422	0.9961	0.9854	0.9794	0.9676	0.9357	0.8761

表 5 既存手法と提案手法の正答数および Accuracy の比較

	Benign (7189)			Malicious (221)			Total Accuracy	
Method	True	False	Error	True	False	Error	exclude error	all
PDFMS	6817	4	368	207	2	12	0.9991	0.9479
Previous ($c = 2^8$, $\nu = 2^{-9}$)	7172	15	2	221	0	0	0.9980	0.9977
Propsed ($c = 2^4$, $\nu = 2^{-11}$)	7182	5	2	218	3	0	0.9989	0.9987

る．先述のとおり，PDFMS は教師あり学習を用いたツールのため，学習に正常 PDF とマルウェア PDF の双方を必要とする．一方，本手法は教師なし学習のため，学習にマルウェアが必要なく，分類器を作成するためには正常 PDF のみを収集すればよい．

結論として，本手法は教師なし学習でありながら，教師あり学習手法とほぼ同等の性能を有する手法である．

7. まとめ

本稿では，正常 PDF の特徴ベクトルを +1 クラスとす

る One-Class SVM の分類器を作成し，−1 クラスに分類されたベクトルをマルウェアと判定する手法を改良した．学習には，筆者らの研究 [11] で提案した 8 次元のベクトルの特徴量に改良を加え，新たな特徴量を追加した 12 次元のベクトルを用いた．実験の結果，提案手法は，従来手法より広い範囲のパラメタで，より高い Accuracy を示した．Total Accuracy が最大となった $c = 2^4$，$\nu = 2^{-11}$ では，評価用正常 PDF 7189 個中 7182 個，マルウェア PDF 221 個中 218 個を正しく検出することができた．また，本手法は教師なし学習手法でありながら，教師あり学習を用いた既存手法と比べてもほぼ同等の性能を有している．教師なし学習手法は，教師あり学習手法に比べ分類器の作成が容易であるため，本手法は有用であるといえる．

参考文献

[1] Adobe Systems Incorporated: Document management - Portable document format - Part 1:PDF 1.7, Adobe System Incorporated (online), available from ⟨http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/PDF32000_2008.pdf⟩ (accessed 2017-11-05).

[2] Schölkopf, B., Williamson, R. C., Smola, A. J., Shawe-Taylor, J. and Platt, J. C.: Support Vector Method for Novelty Detection, *Advances in Neural Information Processing Systems 12* (Solla, S. A., Leen, T. K. and Müller, K., eds.), MIT Press, pp. 582–588 (2000).

[3] Selvaraj, K. and Gutierrez, N. F.: The Rise of PDF Malware, Symantec Corporation (online), available from ⟨https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the_rise_of_pdf_malware.pdf⟩ (accessed 2017-11-05).

[4] Raynal, F., Delugr, G. and Aumaitre, D.: Malicious origami in PDF., *Journal in Computer Virology*, Vol. 6, No. 4, pp. 289–315 (2010).

[5] Blonce, A., Filiol, E. and Frayssignes, L.: Portable document format (pdf) security analysis and malware threats, *Presentations of Europe BlackHat 2008 Conference* (2008).

[6] Laskov, P. and Šrndić, N.: Static Detection of Malicious JavaScript-bearing PDF Documents, *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, New York, NY, USA, ACM, pp. 373–382 (2011).

[7] Maiorca, D., Giacinto, G. and Corona, I.: *A Pattern Recognition System for Malicious PDF Files Detection*, pp. 510–524, Springer Berlin Heidelberg (2012).

[8] Pattern Recognition and Applications Lab: Slayer, https://pralab.diee.unica.it/en/Slayer.

[9] Srndic, N. and Laskov, P.: Detection of Malicious PDF Files Based on Hierarchical Document Structure., *NDSS*, The Internet Society (2013).

[10] Smutz, C. and Stavrou, A.: Malicious PDF Detection Using Metadata and Structural Features, *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, New York, NY, USA, ACM, pp. 239–248 (2012).

[11] Iwamoto, M., Oshima, S. and Nakashima, T.: A Malware Detechtion Method Based on One-Class SVM Focusing on Features of PDF Files, *ICIC EXPRESS LETTERS*, Vol. 11, pp. 1611–1618 (2017).

[12] Cristianini, N. and Shawe-Taylor, J.: *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*, Cambridge University Press, New York, NY, USA (2000).

[13] Clyph & Cog, LLC.: Xpdf, Clyph & Cog, LLC. (online), available from ⟨http://www.foolabs.com/xpdf/home.html⟩ (accessed 2017-11-05).

[14] VirusTotal: VirusTotal, VirusTotal (online), available from ⟨https://www.virustotal.com/ja/⟩ (accessed 2017-11-05).

[15] 神薗雅紀，秋山満昭，笠間貴弘，村上純一，畑田充弘，寺田真敏：マルウェア対策のための研究用データセット〜MWS Datasets 2015〜，技術報告 6，プライスウォーターハウスクーパース株式会社, 日本電信電話株式会社，NTT セキュアプラットフォーム研究所, 国立研究開発法人情報通信研究機構, 株式会社 FFRI, エヌ・ティ・ティ・コミュニケーションズ株式会社, 株式会社日立製作所 (2015).

[16] Chang, C.-C. and Lin, C.-J.: LIBSVM – A Library for Support Vector Machines, National Taiwan University (online), available from ⟨https://www.csie.ntu.edu.tw/~cjlin/libsvm/⟩ (accessed 2017-11-05).