Vol.2020-CSEC-91 No.20
Vol.2020-SPT-40 No.20
Vol.2020-EIP-90 No.20
2020/11/26

暗号化された悪性PDFファイルの検知手法

古川 菜摘[1,a)]　今泉 貴史[2,b)]

概要：標的型攻撃に用いられる悪性 PDF ファイルの静的解析には，JavaScript に着目したものや構造に着目したものなどがある．それらの手法の中には PDF ファイルが暗号化されていた場合，特徴を抽出できず検知できないものが存在する．本論文では，PDF ファイルが暗号化されていた場合に既存の手法が有効かを検討し，暗号化された悪性 PDF ファイルの検知に有用と考えられる新たな特徴を検討した．Stream から得られる情報を用いた検知実験の結果，99.125%の正解率を示した．

Encrypted malicious PDF detection method

1. はじめに

企業を狙った標的型攻撃では，閲覧ソフトの脆弱性を悪用するコードが埋め込まれていたり，別のマルウェアをダウンロードさせたりする PDF(以下，悪性 PDF ファイル) が使われる．PDF は静的な文書形式ではなく，スクリプトを実行したりできることが PDF ファイルを用いた攻撃を可能にしている．悪性 PDF ファイルは，メモリの破損や Use-After-Free，バッファオーバーフローといった PDF 閲覧ソフトの脆弱性を利用する．攻撃は JavaScript ベースのもの，ActionScript ベースのもの，別の悪性ファイルを埋め込んだものなどがある．悪性 PDF ファイルは拡張子などの見た目に違いがないため，正常な PDF ファイルと見分けることが難しい．よって，静的解析や動的解析による検知が必要である．

既存の手法の課題として，パスワードを用いて暗号化された PDF ファイルに有効でないものが多いことが挙げられる．パスワードで暗号化されたファイルはターゲットを安心させる狙いで使われることがある [1]．また，パスワード付きの悪性 PDF ファイルを用いた攻撃が観測されている [2]．

暗号化された PDF ファイルに有効でない例として，JavaScript の特徴を抽出する手法が挙げられる．暗号化されていた場合 PDF ファイルからコードを抽出できない．ほとんどの手法が PDF ファイルが暗号化されていないか，パスワードが判明している前提で検討されている．

暗号化されているファイルに対しての対処法として，Liu ら [3] がパスワード解析ツールの利用を挙げている．しかし，パスワード解析ツールは総当たりで解析し時間がかかるため，組織のゲートウェイでの検査には向かない．他にも，大坪ら [4] が暗号化された PDF ファイルの復号化を行っているが，空白のパスワードが使われる前提となっているため，任意のパスワードが指定されている場合には復号できない．

本論文では，PDF ファイルが暗号化されていた場合に既存の手法が有効かを検討し，有効と考えられる手法を実際に用いて検知実験を行った．次に，PDF の構成要素の一つである Stream のサイズや個数から得られる情報を元に機械学習のモデルを生成する手法を提案する．評価のために，10 分割交差検証法によるモデルの評価とテストデータの検知を行った．

2. PDF

2.1 PDF の基本構成要素

PDF は図 1 に示す 4 つの部分から構成されている．

ヘッダ

PDF のバージョンが記載される．

ボディ

インダイレクトオブジェクトと呼ばれる，名前付きオブ

1　千葉大学大学院融合理工学府
Graduate School of Science and Engineering, Chiba University
2　千葉大学統合情報センター
Institute of Management and Information Technologies, Chiba University
a)　Furukawa723@chiba-u.jp
b)　imaizumi_takashi@faculty.chiba-u.jp

Vol.2020-CSEC-91 No.20
Vol.2020-SPT-40 No.20
Vol.2020-EIP-90 No.20
2020/11/26

```
%PDF-1.2                          header
-----------------------------
obj 1 0
<<
/Type /Catalog
(中略)
>>
endobj
obj 5 0
<<                                body
/Length 1605
/Filter /FlateDecode
>>
stream
…
endstream
endobj
-----------------------------
xref
0 5
00000000000 00000 n               x-ref
00000000000 00000 n               table
00000000000 00000 n
-----------------------------
trailer
(中略)                             trailer
%EOF
```

図 1 PDF の構造の例

```
obj 4 0
 Type: /Page
 Referencing: 3 0 R, 5 0 R, 6 0 R

  <<
    /Type /Page
    /Parent 3 0 R
    /MediaBox [0 0 612 792]
    /Contents 5 0 R
    /Resources
      <<
        /ProcSet [/PDF /Text]
        /Font
          <<
            /F1 6 0 R
          >>
      >>
  >>
```

図 2 オブジェクトの例

ジェクトから構成されている．オブジェクトとは，ファイルを構成する様々な要素である．obj の数字は他のオブジェクトからの参照に使われている．オブジェクトは，boolean, number, string, name, array, dictionary, Stream, null の 8 種類がある．dictionary は<<>>で囲まれたキーと任意の値のペアであり，Stream を付加できる．また，オブジェクトには文書内容の情報を保持する Catalog，PDF ファイルのページの内容を格納する Page，PDF ファイルを開くときに指定した動作を実行する OpenAction などの種類がある．

クロスリファレンス・テーブル

ボディにあるインダイレクトオブジェクトにランダムにアクセスするために，どの位置にあるかを示したものである．

トレイラー

クロスリファレンステーブルと特殊なオブジェクトを読み取るためのもので，PDF の最後に置かれている．PDF ファイルの情報を格納した Info オブジェクトが存在する．

2.2 キー

dictionary に含まれる，/から始まる文字列のことを指す．図 2 に示す通り，キーは様々な型の値をとる．

2.3 Stream

Stream には，画像やスクリプトなどのデータが格納されている．stream と endstream で囲まれた部分がデータになる．多くの場合，FlateDecode や Ascii85Decode などのフィルタで圧縮されている．

2.4 暗号化

PDF はファイルが暗号化されていることもある．暗号化処理を行うセキュリティハンドラには，共通鍵方式の標準セキュリティハンドラと公開鍵方式の公開鍵セキュリティハンドラがある．本稿では広く使われている標準セキュリティハンドラ (パスワード方式) を前提とし，暗号化はパスワードによって行う．暗号化されている場合，暗号化されるのは PDF 中の string，stream，EmbeddedFiles のみである．よって構造やメタデータといった情報は暗号化しても消えない．暗号化方式は，初期は RC4 でバージョン 1.6 からは AES が使用可能になった．

3. 既存の PDF マルウェア検知手法

暗号化された PDF ファイルでは，既存の研究で使われた特徴が使えない場合がある．静的解析の手法を 3 種類に分け，暗号化された PDF ファイルに有効かどうかを検討した．表中の○は暗号化されていない場合と精度が変わらないもの，△は暗号化されていても特徴を利用可能だが制約があるもの，×は暗号化されていると利用できない特徴である．

なお，既存の PDF マルウェア検知手法の中には複数のカテゴリにまたがるものが存在する．検知手法が複数の手法から成り立っている場合には該当するカテゴリの手法のみを考慮した．

3.1 JavaScript，文字列を用いるもの

Pareek ら [5] は，難読化された JavaScript の存在などを用いて検知している．Slayer Neo[6] は，埋め込まれたキーワードやその出現頻度を用いている．Vatamanu ら [7]，PJScan[8] は JavaScript の字句解析を行っている．MPScan[9] は JavaScript から抽出した文字列のエントロピーを計算している．PDF Scrutinizer[10] は使用された文字列の長さを用いている．

JavaScript を解析する手法では，暗号化されている場

合に JavaScript のコードを抽出できず，検知できない．Stream 以外の文字列も暗号化されるため，JavaScript 以外の文字列を利用する手法でも検知精度が下がることが考えられる．関連研究の手法のまとめを表 1 に示す．

表 1 暗号化への対応：JavaScript，文字列を用いるもの

関連研究	暗号化への対応	備考
Pareek ら [5]	△	JavaScript に関する特徴は抽出不可能．EOF の後の文字列は暗号化されないと考えられるため，利用可能
Slayer Neo[6]	×	JavaScript の情報は抽出不可能
Vatamanu ら [7]	×	
PJScan[8]	×	
MPScan[9]	×	
PDF Scrutinizer[10]	×	

3.2 内部構造に着目したもの

分類できないセクションの存在や参照されない Stream の存在，リンクされていないオブジェクト [4] や構造の仕様からの逸脱 [11] に着目したものがある．また，Hidost[12] は，PDF のキーワードの木構造と値を利用し，検知している．暗号化しても PDF ファイルの構造はほぼそのまま残るため，内部構造に着目したものは精度が落ちないと考える．ただし，オブジェクトストリームでオブジェクトが圧縮されていたり，良性 PDF ファイルと同じ構造をもつ悪性 PDF ファイルを検知できないという問題点がある．関連研究の手法のまとめを表 2 に示す．

表 2 暗号化への対応：内部構造に着目したもの

関連研究	暗号化への対応	備考
大坪ら [4]	△	Stream を復号できない
Hidost[12]	△	一部の文字列が暗号化されている
ドゥアンパチャンら [11]	△	オブジェクトストリームに非対応

3.3 メタデータなどに着目したもの

Pareek ら [5]，PDF Malware Slayer[13]，Slayer Neo[6]，PDFrate[14] は，PDF ファイルに埋め込まれたキーやその出現頻度を用いている．Slayer Neo[6] は Stream などの構成要素の出現数を考慮している．オブジェクトの数やサイズ，キーは暗号化によって変化しにくい．よってメタデータを用いる手法で精度が得られる場合には，暗号化されたファイルの検知に一番有用であると考える．関連研究の手法のまとめを表 3 に示す．

表 3 暗号化への対応：メタデータに着目したもの

関連研究	暗号化への対応	備考
Pareek ら [5]	○	使用しているフィルタの種類，キーの出現頻度など
Slayer Neo[6]	○	ファイルサイズ，Stream の数，キーなど
PDF Malware Slayer[13]	○	
PDFrate[14]	○	
Pareek ら [15]	△	エントロピーは暗号化前と後で異なるため，精度が落ちると考えられる．

4. Stream の情報を利用した悪性 PDF ファイルの検知手法

暗号化されている場合，ファイルのメタデータや構造を用いて検知する手法が有効であると考えられる．本論文では，Stream のサイズや個数などの特徴の差を用いて，機械学習により検知する．事前に学習用データを用いて構築した学習モデルにテスト用データを入力し，良性か悪性かを判定する．機械学習のアルゴリズムはランダムフォレストを用いる．

以下に各特徴の概要を示す．特徴の抽出には，Python で作成したツールとフリーの PDF 解析ツールである peepdf[16] を用いた．

Stream のサイズの最大値・最小値・平均値

最大値・最小値・平均値を特徴として利用する．良性 PDF ファイルと悪性 PDF ファイルで，Stream のサイズの傾向が異なると考えられる．これは，良性 PDF ファイルは画像などサイズの大きい構成要素が含まれていたり，通常発生しうる小さなサイズの Stream があるのに対し，悪性 PDF ファイルは最低限の構成要素で十分であるため特定のサイズの Stream に偏っていると考えられるためである．

全体に対する Stream のサイズ・個数の割合

$$\frac{\sum \text{Stream のサイズ}}{\text{ファイルサイズ}} \tag{1}$$

$$\frac{\text{Stream の個数}}{\text{オブジェクトの個数}} \tag{2}$$

バイナリ全体における Stream のサイズの割合 (式 1) と，

オブジェクト全体における個数の割合 (式 2) を用いる．前述したとおり悪性 PDF ファイルは最低限の構成要素で成り立っているケースが多いため，Stream のサイズやオブジェクト全体における個数の割合が良性より大きくなる．

Stream の含まれているパスの深さの最大値・最小値

特徴として，Stream を含むパスの長さの最大値と最小値を用いる．PDF ファイルのオブジェクトは親子関係を持つ木構造となっている．図 3 に PDF のオブジェクトの木構造の例を示す．括弧の中の数字はオブジェクト番号である．この場合，stream(5)，stream(7) のパスの長さは 5，stream(9) のパスの長さは 3 となる．

```
/Catalog (1)
    /Pages (2)
        /Page (3)
            /Pages (2)
            /Annot (4)
                stream (5)
            /Annot (6)
                stream (7)
    /Action /JavaScript (8)
        stream (9)
```

図 3　PDF の木構造

通常生成される PDF ファイルは，オブジェクトの数が多くなるため Stream を含むパスの長さの最大値が大きくなる傾向にある．一方，悪性 PDF ファイルは必要最低限の構成要素しか持たない傾向にあるため，Stream を含むパスの長さの最大値が良性 PDF ファイルと比べて小さくなる．また，必要最低限の構成要素には Catalog，Pages など外せないオブジェクトが存在するため，Stream を含むパスの長さの最小値は悪性 PDF ファイルの方が大きくなる．

5. 実験

5.1 データセット

本研究では，データセットとして contagio[17] で配布されている良性 PDF ファイルと悪性 PDF ファイルのデータセットを用いた．そしてそれらを，pdftk を用いて暗号化した．悪性 PDF ファイルの中には暗号化に失敗した検体があり，それらはデータセットから除いている．contagio のデータセットの中で暗号化できたファイルの中から，暗号化良性ファイル，暗号化悪性ファイルをそれぞれ 1900 個ずつ抽出した．良性悪性各 1900 個のうち 1500 個は本論文の提案手法のモデル作成に用い，残り 400 個は作成したモデルや先行研究に対するテストデータとして利用した．

5.2 実験内容

まず教師データ 1500 個ずつを用いてランダムフォレストによるモデルを作成し，10 分割交差検証によりモデルの性能を評価した．次に，テストデータをそれぞれ 400 個ずつ用意し，検知実験を行った．機械学習のツールとして WEKA[18] を利用した．

5.3 結果

表 4　モデルの交差検証

TPR	FPR	TNR	FNR
0.989	0.011	0.989	0.011

表 5　モデルの分類結果

	良性	悪性
暗号化良性 PDF ファイル	1484	16
暗号化悪性 PDF ファイル	16	1484

モデルの評価結果を表 4-表 5 に示す．TPR，TNR の 2 つとも 0.989 であった．TPR は悪性検体を正しく悪性と検知した割合で，TNR は良性検体を正しく良性と検知した割合である．

次に，テストデータの検知結果を表 6-表 7 に示す．正解率は全体で 99.125%であった．暗号化良性 PDF ファイル 400 個のうち，悪性と誤って判断されたのは 3 個で，暗号化悪性 PDF ファイル 400 個のうち良性と誤って判断されたものは 4 個だった．

表 6　テストデータの TPR/FPR/TNR/FNR

TPR	FPR	TNR	FNR
0.990	0.008	0.993	0.010

表 7　テストデータの分類結果

	良性	悪性
暗号化良性 PDF ファイル	397	3
暗号化悪性 PDF ファイル	4	396

6. 考察

6.1 既存手法の評価

既存の悪性 PDF ファイル検知手法で，構造の特徴を用いているものやメタデータを用いているものは精度が下がりにくいと考える．ドゥアンパチャン [11] の手法を用い，暗号化悪性 PDF ファイルと暗号化良性 PDF ファイルそれぞれ 400 個ずつのテストデータに対し検知実験を行った．

また，比較として，暗号化前の元の PDF ファイルに対しても同様に検知実験を行った．文書ファイルの構造の矛盾に着目した研究の分類結果を表 9 に示す．なお，良性の検体に試験プログラムでエラーが発生したものが存在した．

正解率は 91.5%となり，約 8.5%の誤検知が発生している．良性の PDF ファイルにおいては，暗号化前は大幅に誤検知が起きている．暗号化後に誤検知が減った理由としては，実験でファイルの暗号化に用いた pdftk が構造の欠落や矛盾を補っていたり，元の良性ファイルに存在したオ

表 8 暗号化された PDF ファイルの分類結果

	良性	悪性	エラー
暗号化良性ファイル	376	23	1
暗号化悪性ファイル	45	355	0
元の良性ファイル	301	93	6
元の悪性ファイル	44	356	0

ブジェクトストリームが展開される仕様であったからと考えられる．

以上より，構造に着目した手法を利用することで，暗号化されていても悪性 PDF ファイルを検知できる．しかし，現状では誤検知が多く発生している．

最後に，検知できなかった検体のみをテストデータに用いて提案手法で検知できるかどうかを調べた．

表 9 先行研究で正しく分類できなかったものの分類結果

	良性	悪性
暗号化良性 PDF ファイル	23	0
暗号化悪性 PDF ファイル	1	44

結果から，構造に矛盾のない悪性 PDF ファイルや構造に矛盾のある良性 PDF ファイルを正しく分類できていることがわかる．本研究の手法を併用することで，より高精度な検知が行えると考える．

6.2 良性 PDF ファイルを元にした暗号化悪性 PDF ファイルの検知

悪性 PDF ファイルは exploit を実行できれば十分なため，表示用のコンテンツが少ないという特徴がある．このことからくる特徴を検知に用いた場合，表示用のコンテンツを多く含む PDF ファイルを悪性 PDF ファイル作成に用いることで，簡単に検知を回避できてしまう．本研究では kali linux にインストールされている metasploit を用いて，良性 PDF ファイル 200 個をベースとした悪性 PDF ファイルを作成し，それらを暗号化して検知した．

表 10 良性 PDF ファイルを元にした暗号化悪性 PDF ファイルの分類結果

	良性	悪性
暗号化悪性 PDF ファイル	200	0

表 10 に示すとおり，全ての検体において誤検知が発生した．理由として，検知に用いている暗号化された悪性 PDF ファイルの特徴がファイルの構成要素の少なさからきているため，良性 PDF ファイルをベースにした検体では特徴量が良性 PDF ファイルのものと近くなってしまったことが挙げられる．良性 PDF ファイルと同等の構成要素を持つ検体を検知するには，悪性 PDF ファイルにのみみられる構造の特徴を用いることが必要である．

6.3 悪性 PDF ファイルを模した暗号化良性 PDF ファイルの検知

悪性 PDF ファイルには，
- JavaScript
- 実行可能ファイル
- 別の PDF

が埋め込まれていることが多い．そのうち JavaScript と PDF の埋め込みは良性 PDF ファイルでも存在する．よってそれらの特徴があっても誤検知しないことが望ましい．

悪性 PDF ファイルを模してそれらの要素を埋め込んだ良性 PDF ファイルのデータセット [19] を用いて，検知実験を行った．データセットの内訳は，PDF ファイルを埋め込んだ良性 PDF ファイルが 500 個，JavaScript を埋め込んだ良性 PDF ファイルが 500 個である．それらを pdftk で暗号化して用いた．表 11 に結果を示す．

表 11 PDF/JS が埋め込まれた暗号化良性 PDF ファイルの分類結果

	良性	悪性
JS を含む良性 PDF ファイル	477	23
PDF を含む良性 PDF ファイル	444	56

JavaScript が埋め込まれた良性 PDF ファイルは，正解率が 95.4%であった．PDF ファイルが埋め込まれた良性 PDF ファイルは，正解率が 88.8%だった．Stream から得られる特徴量が，PDF や実行可能ファイルの埋め込みには影響を受けないことが考えられる．

6.4 有効な特徴の選択

提案した 7 つの特徴の中から，有効であったと考えられる特徴を抽出した．抽出には，WEKA に搭載されている Attribute Evaluator を用いた．抽出の結果，以下の 3 つの特徴が有効であると示された．
- Stream の含まれているパスの深さの最大値
- Stream の含まれているパスの深さの最小値
- Stream の最大値

特徴量を削除して，再度モデルの評価とテストデータでの検知実験を行った．モデルの 10 分割交差検証の結果の比較と，テストデータの分類結果を表 12-表 13 に示す．

表 12 モデルの性能の比較

	TPR	FPR	TNR	FNR
特徴 (3 個)	0.981	0.016	0.984	0.019
特徴 (7 個)	0.989	0.011	0.989	0.011

表 13 分類結果

	良性	悪性
暗号化良性 PDF ファイル	395	5
暗号化悪性 PDF ファイル	10	390

特徴を削除しても，大きく検知率は変わらなかった．特徴をこれら 3 つに減らすことで，精度を保ちながら前処理の時間を削減できると考える．実際にテストデータ 400 個の総処理時間を測定したところ，良性は元の 70%，悪性は元の 76%の処理時間となった．結果を表 14 に示す．

表 14 検体 400 個の処理時間

	特徴 (7 個)	特徴 (3 個)
良性の処理時間 (s)	322	226
悪性の処理時間 (s)	198	150

また，特徴量がどのような分布となっているか，教師データを元に調べた．ここで，各グラフの縦軸は累積検体数である．

Stream のサイズの最大値・最小値・平均値

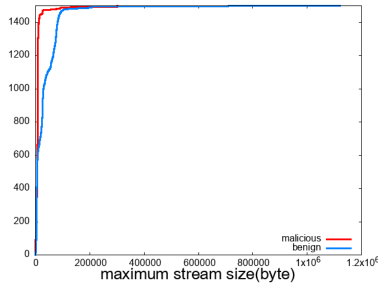

図 4 Stream のサイズの最大値

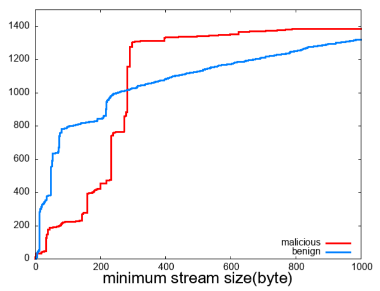

図 5 Stream のサイズの最小値

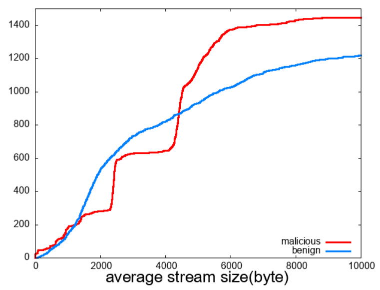

図 6 Stream のサイズの平均値

図 4 は Stream のサイズの最大値の累積検体数を示したものである．悪性 PDF ファイルの Stream のサイズの最大値の分布の方が良性 PDF ファイルのものより小さくなっている．良性 PDF ファイルの方が最大値が大きい傾向にあることから，検知に有用な特徴であったと考える．

一方，Stream のサイズの最小値 (図 5) と Stream のサイズの平均値 (図 6) は値が同じような分布になっており，うまく良性と悪性を分離できなかったと考える．

全体に対する Stream のサイズ・個数の割合

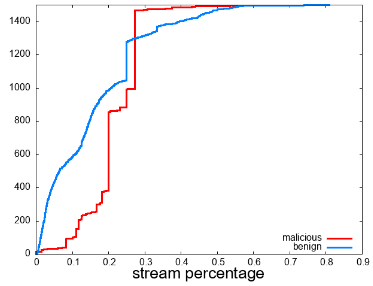

図 7 Stream の個数の割合

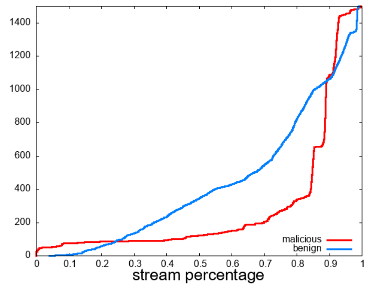

図 8 Stream のサイズの割合

Stream の個数の割合 (図 7) は，特定の値周辺に悪性 PDF ファイルの Stream の個数の割合が集中しているものの，0.25 付近に良性 PDF ファイルの山があることから，良性と悪性がうまく分類できなかったと考える．Stream のサイズの割合 (図 8) は悪性 PDF ファイルが 1 周辺に固まっている．Attribute Evaluator では有効でないとされたものの，ある程度検知に利用できる特徴であると考える．

Stream の含まれているパスの深さの最大値・最小値

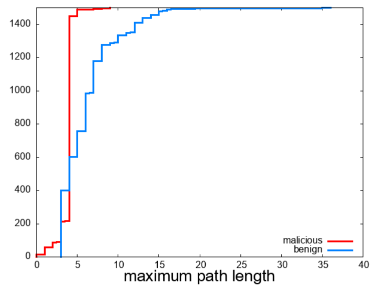

図 9 Stream の含まれているパスの深さの最大値

Stream を含むパスの深さの最大値 (図 9) は悪性 PDF ファイルが 4 付近に集中している．一方良性 PDF ファイルは分散が大きいため，検知にうまく利用できたと考える．

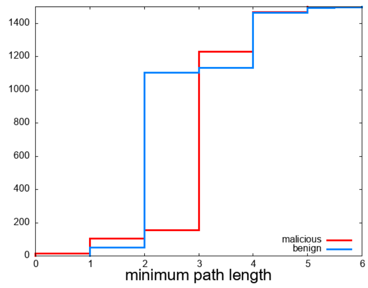

図 10 Stream の含まれているパスの深さの最小値

最小値 (図 10) も同様に，良性が 2，悪性が 3 に集中しているため，うまく分類できたと考える．

7. おわりに

標的型攻撃に用いられる悪性 PDF ファイルの静的解析には，JavaScript に着目したものや構造に着目したものなどがある．それらの手法の中には PDF ファイルが暗号化されていた場合，特徴を抽出できず検知できないものが存在する．本論文では，PDF ファイルが暗号化されていた場合に既存の手法が有効かを検討した．また，暗号化された悪性 PDF ファイルの検知に Stream から得られる情報を用いた．機械学習により暗号化良性/暗号化悪性テストデータ 400 個ずつで実験し，99.125%の正解率で検知に成功した．

今後の課題としては，模倣攻撃への耐性のある特徴の検討が挙げられる．Stream のサイズやパスの深さなどは，構成要素の多い良性 PDF ファイルを元に悪性 PDF ファイルを作成した場合，良性 PDF ファイルの特徴に寄ることが考えられる．実際に良性 PDF ファイルを元にした暗号化悪性 PDF ファイルを作成しての検知実験を行ったところ，1 つも検知できなかった．

本論文の提案手法には上記のような条件でうまく検知できないといった制約があるものの，他の手法と併用して検知することで，より高精度に暗号化悪性 PDF ファイルを検知できると考える．

参考文献

[1] EMSISOFT: 5 ways to protect yourself against encrypted email attachment malware, , available from ⟨https://blog.emsisoft.com/en/31624/5-ways-to-protect-yourself-against-encrypted-email-attachment-malware/⟩ (accessed 2020-09-28).

[2] Abrams, L.: CIA Porn Extortion Scams Now Use Password Protected PDFs, , available from ⟨https://www.bleepingcomputer.com/news/security/cia-porn-extortion-scams-now-use-password-protected-pdfs/⟩ (accessed 2020-09-28).

[3] Liu, D., Wang, H. and Stavrou, A.: Detecting Malicious Javascript in PDF through Document Instrumentation, *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 100–111 (2014).

[4] 大坪雄平，三村守，田中英彦：ファイル構造検査の悪性 PDF ファイル検知への応用, 情報処理学会論文誌, Vol. 55, No. 10, pp. 2281–2289 (2014).

[5] Pareek, H., Eswari, P. and Babu, N. S. C.: Malicious PDF document detection based on feature extraction and entropy, *International Journal Journal of Security, Privacy and Trust Management*, Vol. 2, No. 5 (2013).

[6] Maiorca, D., Ariu, D., Corona, I. and Giacinto, G.: A structural and content-based approach for a precise and robust detection of malicious PDF files, *2015 International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 27–36 (2015).

[7] Vatamanu, C., Gavriluţ, D. and Benchea, R.: A practical approach on clustering malicious PDF documents, *Journal in Computer Virology*, Vol. 8, No. 4, pp. 151–163 (2012).

[8] Laskov, P. and Šrndić, N.: Static detection of malicious JavaScript-bearing PDF documents, *Proceedings of the 27th annual computer security applications conference*, pp. 373–382 (2011).

[9] Lu, X., Zhuge, J., Wang, R., Cao, Y. and Chen, Y.: De-obfuscation and detection of malicious PDF files with high accuracy, *2013 46th Hawaii International Conference on System Sciences*, IEEE, pp. 4890–4899 (2013).

[10] Schmitt, F., Gassen, J. and Gerhards-Padilla, E.: PDF SCRUTINIZER: Detecting JavaScript-based attacks in PDF documents, *2012 tenth annual international conference on privacy, security and trust*, IEEE, pp. 104–111 (2012).

[11] ドゥアンパチャンカンボリスット，今泉貴史：構成要素の関係性を利用した悪性 PDF ファイルの検知，情報科学技術フォーラム講演論文集，pp. 171–175 (2015).

[12] Šrndić, N. and Laskov, P.: Hidost: a static machine-learning-based detector of malicious files, *EURASIP Journal on Information Security*, Vol. 2016, No. 1, p. 22 (2016).

[13] Maiorca, D., Giacinto, G. and Corona, I.: A pattern recognition system for malicious pdf files detection, *International workshop on machine learning and data mining in pattern recognition*, Springer, pp. 510–524 (2012).

[14] Smutz, C. and Stavrou, A.: Malicious PDF detection using metadata and structural features, *Proceedings of the 28th annual computer security applications conference*, pp. 239–248 (2012).

[15] Pareek, H., Eswari, P., Babu, N. S. C. and Bangalore, C.: Entropy and n-gram analysis of malicious PDF documents, *International Journal of Engineering*, Vol. 2, No. 2 (2013).

[16] : peepdf - PDF Analysis Tool, , available from ⟨https://eternal-todo.com/tools/peepdf-pdf-analysis-tool⟩ (accessed 2020-10-22).

[17] Mila: contagio, , available from ⟨http://contagiodump.blogspot.com/⟩ (accessed 2020-09-28).

[18] The University of Waikato: WEKA, , available from ⟨https://www.cs.waikato.ac.nz/ml/weka/⟩ (accessed 2020-10-15).

[19] PRA Lab, University of Cagliari: PDF Reverse Mimicry Dataset, , available from ⟨https://pralab.diee.unica.it/en/pdf-reverse-mimicry/⟩ (accessed 2020-09-28).