Vol.2013-DPS-154 No.21
Vol.2013-CSEC-60 No.21
2013/3/14

悪性文書ファイルへのRATの埋め込み方式の調査

三村 守[1,2,a)]　大坪 雄平[2]　田中 英彦[1]

概要：今日，機密情報や個人情報の搾取を目的とする標的型攻撃は，多くの組織にとって脅威である．標的型攻撃の初期段階では，攻撃者はRAT（Remote Access Trojan）と呼ばれるマルウェアをメールで送付し，コンピュータの遠隔操作を試みることが多い．RATが文書ファイルに埋め込まれた場合には，検知はより困難となる．よって，標的型攻撃を防ぐためには、悪性文書ファイルに埋め込まれたRATを検知する必要がある．本稿では，RATがどのように悪性文書ファイルに埋め込まれているのかを調査し，その方式を体系化して説明する．さらに，悪性文書ファイルへのRATの埋め込み方式を解読し，RATを検知する手法について考察する．

キーワード：標的型攻撃，RAT，マルウェア，悪性文書ファイル，静的解析

Investigating how to embed RATs into a malicious document file

Mimura Mamoru[1,2,a)]　Otsubo Yuhei[2]　Tanaka Hidehiko[1]

Abstract: Today, targeted attacks that exploit confidential information or personal information are serious threats for many organizations. At an early phase of a targeted attack, most attackers send malware called RAT(Remote Access Trojan) by e-mails, and attempt to control the computer. If the RAT is concealed in a document file, it is more difficult to reveal it. Thus, to defeat targeted attacks, it is necessary to detect RATs in malicious document files. In this paper, we investigate how to embed RATs into a malicious document file, and explain the systematized methods. Moreover, we consider how to break the embedding methods and detect RATs in malicious document files.

Keywords: targeted attack, RAT, malware, malicious document file, static analysis

1. はじめに

近年，組織が保有する機密情報や個人情報の搾取を目的とするサイバー攻撃の脅威が顕在化している．2011年には国会，政府関係機関，民間企業等において大規模なサイバー攻撃が相次いで発生し，2012年には遠隔操作ウイルスに伴う誤認逮捕が発覚する等，大きな社会問題となっている．サイバー攻撃の中でも特に目立つのは，主に機密情報や個人情報の搾取を目的とし，ある組織や個人に標的を絞って実施される標的型攻撃である．経済産業省が実施した調査によると，2007年には標的型攻撃を受けた経験がある企業は5.4%にとどまっていたが，2011年には約6倍の33%に拡大している [1]．標的型攻撃の中でも，ある組織に特化した，時間および手法を問わずに継続的に行われる一連の攻撃はAPT （Advanced Persistent Threat）や新しいタイプの攻撃 [2] と呼ばれることもあり，大きな脅威となっている．

典型的な標的型攻撃の概要を以下に示す．まず攻撃者は，業務等を装った件名やファイル名をつけた標的型メールで不正プログラム（マルウェア）を送信する．標的型メールを受信したユーザが，業務等を装った件名やファイル名を不審に思わず，添付ファイルを開封した場合，その端末で不正な命令が実行され，マルウェアに感染する．マルウェアに感染した端末は踏み台とされ，攻撃者の遠隔操作によ

[1] 情報セキュリティ大学院大学
IISEC, Yokohama, Kanagawa 221–0835, Japan
[2] 内閣官房情報セキュリティセンター
NISC, Chiyoda, Tokyo 100–0014, Japan
[a)] dgs104101@iisec.ac.jp

り任意の命令が実行され，不正な情報の搾取に利用される．搾取された情報は，コマンド＆コントロールサーバ等に送信され，最終的に攻撃者によって回収される．攻撃者はまた，発信元を秘匿するために，Tor[5] 等の匿名性を向上させるソフトウェアを利用したり，すでにマルウェアに感染しており，遠隔操作が可能な一般ユーザの端末を利用して標的型メールを送信している可能性も考えられる．さらに，不正に搾取した情報を用い，新たに業務等を装った件名，ファイル名，あるいは搾取したファイルそのものを付与した巧妙な手口に発展してきている．

標的型攻撃に用いられるマルウェアは，かつての無作為に拡散するウイルスとは異なる特徴が指摘されている [3]．たとえば，特定の組織にのみ送信されるため，マルウェアを入手できないウイルス対策ソフトのベンダはパターンファイルを作成することができず，結果としてウイルス対策ソフトで検出されにくいという特徴がある．このような特徴の実例としては，Microsoft Word，Adobe Reader 等のアプリケーションに存在する脆弱性を悪用する添付ファイルを開いた際に感染する事象が報告されている．これらの事象では，最新のパターンファイルを適用したウイルス対策ソフトでも，添付ファイルを検出できないケースがほとんどである．

標的型攻撃に用いられるほとんどのマルウェアの本体は実行ファイル（EXE および DLL）であり，メールで直接送付される場合と，外部からダウンロードされる場合がある．本稿では，マルウェアの本体である実行ファイルが，メールで直接送付される場合を想定している．この場合さらに，ほかのファイルには埋め込まれずに拡張子やアイコンが偽装される場合と，DOC 形式，PDF 形式等の文書ファイルに埋め込まれて偽装される場合が考えられる．実行ファイルがほかのファイルに埋め込まれていない場合には，利用者は拡張子やアイコンの偽装に注意すれば比較的容易に不審なメールを見抜くことができる．しかしながら，実行ファイルが文書ファイルに埋め込まれて偽装された場合には，利用者には一般に不審なメールを見抜く手段がない．実際に，実行ファイルと元となるダミーの文書ファイルを指定するだけで，容易にマルウェアを作成するツールの存在が確認されている [4]．マルウェアが埋め込まれた文書ファイルを開くと，exploit（脆弱性を利用した悪意あるコード）が実行され，マルウェアの本体である実行ファイルが作成される場合が多い．よって，悪性文書ファイルに埋め込まれた実行ファイルを，短時間で自動的に検知することができれば，文書ファイルに実行ファイルが埋め込まれているか否かを判定し，マルウェアを検知することができれるものと考えられる．そこで本稿の目的を，文書ファイルに実行ファイルが埋め込まれているか否かを短時間で自動的に判定することとする．

2. 関連研究

本稿では，文書ファイルに実行ファイルが埋め込まれているか否かを，exploit を動作させずに検査する．この検査では実際にマルウェアは動作しないため，本稿は静的解析の一種といえる．ファイルの特徴をもとに静的解析によってマルウェアを検出する手法としては，実行ファイルを分析する手法，それ以外のファイルも分析できる手法および文書ファイルを分類する手法に分類される．

2.1 実行ファイルを分析する手法

文献 [6], [7] では，マルウェアに含まれる可読な文字列を抽出し，教師あり学習の 1 つであるサポートベクタマシンを適用ことによって，通常の実行ファイルとマルウェアを識別する手法が提案されている．文献 [8] では，実行ファイルをいくつかの区間に分割し，分割した区間ごとの情報エントロピーの統計を算出することで，通常の実行ファイルとパッキングされたマルウェアを識別する手法が提案されている．また，文献 [9] では，実行ファイルにパターン認識の手法を適用することで，パッキングされているか否かを判定する手法が提案されている．これらの手法は，実行ファイルがマルウェアか否かを判定する手法である．

本稿では，文書ファイルに埋め込まれた実行ファイルを検知することを目的としており，これらの手法のように実行ファイル自体の分析は実施しない．

2.2 実行ファイル以外も分析できる手法

文献 [10] では，バイナリデータの値を統計的に分析することによって，文書ファイルに隠された不正なコードを検出する手法が提案されている．文献 [11] ではバイナリデータの統計分析に加え，動的解析を組合せて文書ファイルに隠された不正なコードを検出する手法が提案されている．これらの手法では，実行ファイル以外も分析の対象としており，ファイルに隠された exploit を含む不正なコードを検出することに主眼を置いている．また，これらはファイルに不正なコードが含まれているか否かを判定する手法であり，不正なコードの位置やどのような方式で埋め込まれているかを分析する手法ではない．

本稿では，exploit を含む不正なコードではなく，エンコードされたマルウェアの本体（実行ファイル）を検知することを目的としている．マルウェアの本体を検知するためには，実行ファイルが埋め込まれている位置やエンコード方式も明らかにする必要がある．

2.3 文書ファイルを分析する手法

文献 [12] では，潜在的に危険なアクションを伴うフィルタを対象に機械学習を適用することで，不審な PDF 形式の

ファイルを高速に検知する手法が提案されている．この手法では，教師あり学習モデルを用いているため，学習するためのサンプルを集める必要があるだけでなく，その精度は学習のサンプルに依存する．また，PDF 形式のファイル以外の文書ファイルを分析することができない．文献 [13] では，様々な形式の悪性文書ファイルに埋め込まれた実行ファイルを自動的に抽出するツールが提案されている．この手法では，PDF 形式のファイル以外の文書ファイルも分析することが可能であり，学習するためのサンプルを集める必要もない．しかしながら，PDF 形式のファイルの検知率が低いという課題がある．

本稿では，悪性文書ファイルに埋め込まれた実行ファイルの検知率を向上させるため，2009 年から 2012 年の間に複数の組織において発生した標的型攻撃に用いられた検体を分析し，検知に失敗した実行ファイルのエンコード方式を明らかにし，その解読手法を検討する．

3. RAT の埋め込み方式

実行ファイルである RAT は，そのまま文書ファイルに埋め込まれる場合もあるが，多くの場合には何らかの方式でエンコードされて文書ファイルに埋め込まれる．2009 年から 2012 年の間に複数の組織において採取した検体のシェルコードを分析し，判明したエンコード方式に用いられていた主な演算を表 1 に示す．エンコード方式は，主としてこれらの基本的な演算を組み合せて構成されており，換字による方式，転置による方式および固有の方式に分類することができる．換字による方式は，実行ファイルをエンコードする場合に必ず使用される基本的なエンコード方式である．転置による方式および固有の方式は，換字による方式に加えて使用される付加的なエンコード方式であり，複数の方式を組合せることも可能である．

表 1 エンコードに用いられる主な演算
Table 1 Main instructions to encode executable files.

演算	説明
XOR	排他的論理和
ADD, SUB	算術加算または算術減算
ROL, ROR	左論理シフトまたは右論理シフト

3.1 換字による方式

換字による方式は，何らかの方式で作成した鍵との排他的論理和（XOR）を計算することで，実行ファイルを別の文字列に変換する方式である．換字による方式では，元の実行ファイルのオフセット位置はエンコード後にも変化しない．

3.1.1 XOR

実行ファイルと任意の 1byte の鍵を XOR 演算するエンコード方式であり，最も基本的かつ頻繁に用いられている．通常の XOR による方式では，実行ファイルと鍵の XOR 演算を実施すると，実行ファイルの NULL の部分は鍵と同一の値となる．そのため，実行ファイルの NULL の部分に着目することで，容易にエンコードに用いる鍵を特定することが可能である．

3.1.2 Multibyte XOR

実行ファイルと任意の 2byte 以上の鍵を XOR 演算するエンコード方式である．この方式もやはり，実行ファイルの NULL の領域に 2byte 以上の鍵の値がそのまま表れる傾向がある．したがって，実行ファイルの NULL の部分に着目し，2byte 以上の繰返し文字列を検出することで，エンコードに用いる鍵を推定することが可能である．ただし，繰返し文字列を検出することで鍵を特定するためには，実行ファイルに鍵の長さの 2 倍以上の NULL の領域が存在する必要がある．

3.1.3 NULL-Preserving XOR

NULL-Preserving XOR は，エンコードに用いる任意の 1byte の鍵の値と一致する場合，または NULL の場合は演算を実施せず，それ以外の場合のみ XOR 演算を実施する方式である．NULL-Preserving XOR では実行ファイルの NULL の部分は鍵の値とならず，NULL のままとなっているため，一見しただけでは鍵を特定することができない．

3.1.4 Multibyte NULL-Preserving XOR

Multibyte NULL-Preserving XOR は，鍵と同じ長さの領域がすべて NULL の場合には演算を実施せず，それ以外の場合には任意の 2byte 以上の鍵を XOR 演算するエンコード方式である．Multibyte NULL-Preserving XOR では，Multibyte XOR のように実行ファイルの NULL の領域に 2byte 以上の鍵の値がそのまま表れることがない．したがって，2byte 以上の繰返し文字列を検出することで，エンコードに用いる鍵を推定することができない．ただし，Multibyte NULL-Preserving XOR では鍵と同じ長さの領域に 1byte でも NULL 以外の値が含まれている場合には XOR 演算を実施するため，文字コードの頻度分析によって鍵を推定できる可能性がある．

3.1.5 Rolling XOR

Rolling XOR は 1byte 単位の鍵による XOR 演算と考えた場合，鍵の値がある周期で 1byte ごとに変化するエンコード方式である．最も基本的な鍵の値の変化の方式は，1 ずつ加算するインクリメント方式および 1 ずつ減算するデクリメント方式である．これらの方式の場合，256byte ごとに鍵の値が元の値となるため，周期は 256 となる．Rolling XOR を 2byte 以上の鍵による XOR 演算と考えた場合には，Multibyte XOR と実質的に同義となる．この場合，Rolling XOR の鍵の周期が Multibyte XOR の鍵の

長さに相当する．したがって，Multibyte XOR と同様に実行ファイルの NULL の部分に着目し，2byte 以上の繰返し文字列を検出することで，エンコードに用いる鍵を推定することが可能である．Rolling XOR や Multibyte XOR によるエンコード結果は，Rolling XOR と XOR による多重エンコード結果とも同義となる．なぜならば，Rolling XOR と XOR による多重エンコード演算には結合法則が成り立つため，結果として生成される鍵の長さは元の Rolling XOR の周期となるからである．よって多重にエンコード方式が用いられている場合にも，2byte 以上の繰返し文字列を検出することで，エンコードに用いる鍵を推定できる可能性がある．

3.1.6 そのほかの方式

そのほかの換字によるエンコード方式としては，特定の周期のオフセットの場合にのみ演算を実施する方式や，独自のストリーム暗号を用いる方式等があげられる。また，少し変わった方式としては，次の 1byte の値を鍵として XOR 演算を実施する方式も確認されている．

3.2 転置による方式

転置による方式は，何らかの方式で実行ファイルあるいは鍵の順序を並び替える方式である．並び替えの単位は，方式によって byte 単位の場合と bit 単位の場合がある．byte 単位の転置による方式では，元の実行ファイルのオフセット位置はエンコード後に変化する．

3.2.1 論理シフト

対象とする 1byte の値を，1〜4bit 単位で左論理シフトまたは右論理シフトする方式である．4bit 左論理シフトした値と 4bit 右論理シフトした値は同一となる．論理シフトは，実行ファイルに対して実施される場合と，Rolling XOR の鍵に対して実施される場合がある．周期が 256 の Rolling XOR の鍵に対して論理シフトが実施された場合にも，結果として生成される鍵の周期は変わらない．なぜならば，論理シフトの周期である 8 は，256 の約数となるからである．よって，Rolling XOR に論旨シフトが併用された場合にも，2byte 以上の繰返し文字列を検出することで，エンコードに用いる鍵を推定できる可能性がある．

3.2.2 SWAP

対象とする 2byte の値を，1byte 単位で順序を入れ換える方式である．SWAP 演算を実施した文字列に対し，オフセットを 1byte ずらして再度 SWAP 演算を実施する Double SWAP 方式も確認されている．また，1byte 単位ではなく，対象とする 1byte を half byte 単位で順序を入れ換える Nibble SWAP 方式もある．なお，Nibble SWAP 方式は，4bit 左論理シフトおよび 4bit 右論理シフトと同義である．

3.3 固有の方式

固有の方式は，対象とする文書ファイルのファイル形式に依存したエンコード方式である．対象とする文書ファイルが，ほかの異なる形式を解釈することが可能であるために，このような固有の方式によるエンコードが可能となる．

3.3.1 RTF ファイル

ほとんどの RTF（Rich Text Format）形式のファイルにおいては，実行ファイルの 1byte を 2byte の 0〜f の 16 進数にエンコードする方式が用いられている．この方式は，他の方式と組合せて使用され，単独で用いられることはほとんどない．この方式でエンコードされた実行ファイルは，元の実行ファイルの長さの 2 倍の長さの 0〜f の 16 進数にエンコードされているため，容易に発見することが可能である．

3.3.2 PDF ファイル

PDF（Portable Document Format）形式のファイルにおいては，換字による方式および転置による方式で直接実行ファイルを埋め込む以外にも，Java Script を用いたエンコード方式が確認されている．Java Script には，最終的に実行ファイルの 1byte を 2byte の 0〜f の 16 進数にエンコードして埋め込まれていた．ただし，PDF 形式の場合には，実行ファイルが埋め込まれた Java Script がさらに別の方式でエンコードされるため，容易に発見することはできない．

3.3.3 その他の方式

これまでに示した方式以外にも，VBScript や Flash に使用されるプログラム言語である Action Script を用いて実行ファイルをエンコードする方式が確認されている．これらの方式は，そのプログラム言語をサポートしている文書ファイルであれば用いることが可能であり，Microsoft Word や Excell に組み込まれていることが多い．

4. 埋め込み方式の解読手法

悪性文書ファイルに埋め込まれた RAT を検知するためには，その埋め込み方式を解読する必要がある．RAT の埋め込み方式を解読する手法としては，方式 1：総当たり方式，方式 2：繰返し文字列を検出する方式および方式 3：文字コードの頻度分析が考えられる．各エンコード方式と解読手法の対応を表 2 に示す．

4.1 方式 1：総当たり方式

最も単純な埋め込み方式の解読手法は，すべての考えられ得る鍵のパターンを総当たりで試す方式である．ファイル形式固有の方式は，そのファイル形式が対象の場合のみ検索する．総当たり方式は，XOR や NULL-Preserving XOR 等の鍵長が短い場合に有効である．特に，NULL-Preserving XOR は他の方式では解読できないため，唯一の解読手法であると考えられる．しかしながら，鍵長が長い Multibyte

表 2　各エンコード方式に対応する解読手法
Table 2　Breaking methods that correspond to each encoding method.

エンコード方式	方式 1	方式 2	方式 3
XOR	○	○	△
Multibyte XOR	×	○	△
NULL-Preserving XOR	○	×	×
Multibyte NULL-Preserving XOR	×	×	○
Rolling XOR	△	○	△

XOR や Multibyte NULL-Preserving XOR に対しては，現在のコンピュータでは実用的な時間内での解読は不可能である．Rolling XOR については，単純なインクリメント方式やデクリメント方式であれば，実用的な時間内での解読は可能である．同様に，転置による方式や固有の方式についても，併用して検索することで実用的な時間内での解読は可能であると考えられる．

4.2　方式 2：繰返し文字列を検出する方式

鍵長が長いエンコード方式に対しては，総当たり方式では実用的な時間内で解読することはできない．そこで，実行ファイルの NULL の部分に着目し，2byte 以上の繰返し文字列を検出することで，エンコードに用いる鍵を推定する方式が考えられる．この方式が機能するためには，実行ファイルに鍵の長さの 2 倍以上の NULL の領域が存在し，鍵がそのままの状態で悪性文書ファイルに含まれている必要がある．この方式は，XOR，Multibyte XOR および Rolling XOR に有効であると考えられる．同様に，転置による方式や固有の方式についても，併用して検索することで実用的な時間内での解読は可能であると考えられる．しかしながら，鍵がそのままの状態で悪性文書ファイルに含まれない NULL-Preserving XOR や Multibyte NULL-Preserving XOR に対しては，原理的に解読は不可能である．

4.3　方式 3：文字コードの頻度分析

鍵長が長い Multibyte NULL-Preserving XOR に対しては，総当たり方式でも繰返し文字列を検出する方式でも実用的な時間内で解読することができない．Multibyte NULL-Preserving XOR では，鍵と同じ長さの領域に 1byte でも NULL 以外の値が含まれている場合には XOR 演算を実施する．ゆえに，実行ファイルのある程度以上の NULL の領域が含まれていれば，鍵はそのままの状態では含まれていないが，文字コードの頻度分析によって鍵を推定できる可能性が考えられる．この方式は，XOR，Multibyte XOR および Rolling XOR に対してもある程度有効であると考えられる．同様に，転置による方式や固有の方式についても，併用して検索することで実用的な時間内での解読は可能であると考えられる．しかしながら，ファイル形式固有の特徴として多く含まれる文字列によって，文字コードの頻度に誤差が発生する可能性が考えられる．

5.　RAT の検知方式

悪性文書ファイルに埋め込まれた RAT を検知するためには，解読した埋め込み方式を用いてデコードしたデータから，実行ファイルが含まれていることを確認する必要がある．実行ファイルが含まれていることを確認する手法は，MS-DOS 用スタブプログラムを検索する方式と，実行ファイルのヘッダを検索する方式が考えられる．

5.1　MS-DOS 用スタブプログラムを検索する方式

RAT を検知するためには，実行ファイルの MS-DOS 用スタブプログラムに含まれる文字列を検索するのが基本である．MS-DOS スタブプログラムは，実行ファイルが MS-DOS 上で実行された場合に実行されるプログラムであり，一般的には“This program cannot be run in DOS mode”や“This program must be run under Win32”という文字列を表示して終了するプログラムである．しかしながら，MS-DOS 用スタブプログラムに含まれる文字列は任意に設定することが可能であるため，この文字列のみで RAT を検知することはできない．この方式による検知を回避するため，これらの文字列を 1byte のみ変更した実行ファイルも確認されている．また，“PE for Win32”や“Win32 only!”のようなまったく異なる文字列が含まれている場合や，あるいは空白となっている場合もある．

5.2　ヘッダを検索する方式

MS-DOS 用スタブプログラムを検索するよりも確実なのは，実行ファイルのヘッダを検索する方式である．実行ファイルの先頭には，MZ ヘッダおよび PE ヘッダがついているため，“MZ”や“PE”といったこれらのヘッダに含まれる固有のパターンを検索することで，実行ファイルである RAT を検知することができる．しかしながら，この方式による検知を回避するため，”MZ”や”PE”の文字列を削除した実行ファイルを埋め込んでいる場合も確認されている．“MZ”や“PE”の文字列は，シェルコードが実行された場合には復元されるため，実行ファイルは正常に動作する．このような場合には，MS-DOS 用スタブプログラムを検索する方式と合わせて総合的に検索する方式が有効である．実行ファイルの NULL が含まれるオフセットの位置等も，RAT を検知するための有効な手がかりとなるものと考えられる．

6.　おわりに

本稿では，実行ファイルである RAT がどのように悪性文書ファイルに埋め込まれているのかを調査し，その方式

を体系化して整理するとともに，各々の方式の特徴について考察した．さらに，その特徴を考慮して悪性文書ファイルからの RAT の埋め込み方式を解読し，RAT を検知する手法について考察した．

今後の課題としては，本稿で提案した３つの埋め込み方式の解読手法および２つの RAT の検知方式の具体的なアルゴリズムの考案，実装，検知率の評価等があげられる．提案方式の埋め込み方式ごとの検知率を比較すれば，ファイル形式等に応じた最適な解読手法を選択することが可能となる．最適な解読手法を選択することで，RAT の検知に要する時間を短縮することができる．メールサーバやネットワークセンサでリアルタイム処理ができる程度に実用的な速度が得られれば，より強固な標的型攻撃に対する防御システムの構築に寄与するものと考える．

参考文献

[1] 経済産業省：最近の動向を踏まえた情報セキュリティ対策の提示と徹底，経済産業省（オンライン），入手先 ⟨http://www.meti.go.jp/press/2011/05/20110527004/20110527004.html⟩ （参照 2012-10-29）(2011).
[2] 情報処理推進機構：『新しいタイプの攻撃』に関するレポート ～Stuxnet（スタックスネット）等の新しいサイバー攻撃手法の出現～，情報処理推進機構（オンライン），入手先 ⟨http://www.ipa.go.jp/about/technicalwatch/20101217.html⟩（参照 2012-10-29）(2010).
[3] 情報処理推進機構：コンピュータウイルス・不正アクセスの届出状況 [12 月分および 2011 年年間] について，情報処理推進機構（オンライン），入手先 ⟨http://www.ipa.go.jp/security/txt/2012/01outline.html⟩（参照 2012-10-29）(2009).
[4] 情報処理推進機構：脆弱性を利用した新たなる脅威の監視・分析による調査，情報処理推進機構（オンライン），入手先 ⟨http://www.ipa.go.jp/security/vuln/report/newthreat200907.html⟩（参照 2012-10-29）(2009).
[5] Tor Project: Anonymity Online，入手先 ⟨https://www.torproject.org/⟩ (accessed 2012-10-29).
[6] Ye, Y. Chen, L. Wang, D. Li, T. Jiang, Q. and Zhao, M.: SBMDS: an interpretable string based malware detection system, Journal in Computer Virology, Vol.5, No.4, pp.283–293 (2009).
[7] 戸部和洋，森達哉，千葉大紀，下田晃弘，後藤滋樹：実行ファイルに含まれる文字列の学習に基づくマルウェア検出方法，コンピュータセキュリティシンポジウム 2010 論文集, 第二分冊, pp.777–782 (2010).
[8] Lyda, L. and Hamrock, J.: Using Entropy Analysis to Find Encrypted and Packed Malware, Security and Privacy, IEEE, Vol.5, Issue 2, pp.40–45 (2007).
[9] Perdisci, R. Lanzi, A. Lee, W.: Classification of packed executables for accurate computer virus detection, Pattern Recognition Letters, Vol.29, Issue 14, pp.1941–1946 (2008).
[10] Stolfo, S.J. Wang, K. and Li, W.J.: Towards Stealthy Malware Detection, Advances in Information Security, Vol.27, pp.231–249 (2007).
[11] Li, W.J. Stolfo, S.J. Stavrou, A. Androulaki, E. and Keromytis, A.D.: A Study of Malcode-Bearing Documents, Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA), Vol.4 of Lecture Notes in Computer Science, pp.231–250 (2007).
[12] Xu, W. Wang, X. Zhang, Y. and Xie, H.: A Fast and Precise Malicious PDF Filter, Proceedings of the 22nd Virus Bulletin International Conference, pp.14–19 (2012).
[13] 三村守，田中英彦：Handy Scissors: 悪性文書ファイルに埋め込まれた実行ファイルの自動抽出ツール，情報処理学会論文誌, Vol.54, No.3 (2013).