情報処理学会論文誌 Vol.54 No.3 1211–1219 (Mar. 2013)

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

三村 守[1,a)] 田中 英彦[1]

受付日 2012年8月14日, 採録日 2012年12月7日

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．標的型攻撃に利用されるマルウェア本体が文書ファイルに埋め込まれた場合には，発見・解析はより困難となる．標的型メールに添付された不審な文書ファイルの動作を解析するためには，文書ファイルから，埋め込まれた実行ファイル（マルウェア本体）を抽出する必要がある．しかしながら，脆弱性を利用した悪意あるコードである exploit の動作条件が判明せず，動的解析によって速やかに実行ファイルを抽出することが困難なケースが増えている．そこで本稿では，悪性文書ファイルから埋め込まれた実行ファイルを自動的に抽出するツールを試作し，実験によりその有効性を示す．

キーワード：マルウェア，標的型攻撃，悪性文書ファイル，静的解析

Handy Scissors: An Automated Tool that Extracts an Executable File Embedded in a Malicious Document File

Mamoru Mimura[1,a)] Hidehiko Tanaka[1]

Received: August 14, 2012, Accepted: December 7, 2012

Abstract: Today, targeted attacks that exploit confidential information or personal information are serious threats for many organizations. If the malware is concealed in document files, it is more difficult to reveal and analyze it. Analyzing malicious document files attached to spear phishing e-mails requires extracting an embedded executable file. However, if we don't know the condition that the exploit code: a malicious code that takes advantage of vulnerability, runs normally, it is difficult to extract the executable file swiftly by dynamic analysis. In this paper, we develop an automated tool that extracts the embedded executable file from the document files, and the experimental results show the validity and effectiveness of this tool.

Keywords: malware, targeted attack, 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，大きな社会問題となったのは記憶に新しいところである．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）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，業務を装った件名やファイル名を

[1] 情報セキュリティ大学院大学
IISEC, Yokohama, Kanagawa 221–0835, Japan
[a)] dgs104101@iisec.ac.jp

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，ファイアウォールやプロキシを介してコマンド&コントロールサーバに送信される．最後に，コマンド&コントロールサーバに送信された情報は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，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，新たに業務を装った件名やファイル名を付与する悪質なケースも目立つようになってきている．

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 [2]．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．その実例として，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，マルウェア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，添付ファイルを検知できないケースがほとんどである．

2. 研究の目的と範囲

標的型攻撃に用いられる多くのマルウェア本体は実行ファイル（exe および dll）であり，他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 形式や pdf 形式の文書ファイルに埋め込まれて偽装される場合がある．実行ファイルが他のファイルに埋め込まれていない場合には，ただちにマルウェア本体の解析にとりかかることができる．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，ただちにマルウェア本体の解析にとりかかることはできない場合がある．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 [3]．このようなマルウェアの機能を分析するためには，悪性文書ファイルから実行ファイル（マルウェア本体）を抽出する必要がある．

マルウェア本体を抽出する最も簡単な方法は，隔離された安全な環境で，実際にマルウェア本体が埋め込まれた悪性文書ファイルを開き，作成されるファイル等の挙動を観測する動的解析と呼ばれる手法である．マルウェア本体が埋め込まれた悪性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，マルウェア本体である実行ファイルが作成される場合が多い．ゆえに，動的解析において exploit を動作させることができれば，作成された実行ファイルを回収するか，あるいは復元すれば，実行ファイルを抽出することが可能である．作成された実行ファイルの回収や復元は，Capture BAT [4] 等のツールを使用すれば容易に実施することができる．したがって動的解析では，exploit が正常に動作すれば，実行ファイルを取り出すことが可能となる場合がほとんどである．しかしながら，サンドボックス内で exploit が正常に動作せず，実行ファイルを取り出すことができない場合もある．また，ソフトウェアの脆弱性の数は年々増加しており，exploit の動作条件（特定のソフトウェアのバージョンやそれらの組合せ）を特定できず，やはり動的解析で実行ファイルを取り出せないケースが増加している．動的解析で実行ファイルが取り出せない場合には，バイナリエディタ等を利用して悪性文書ファイルから手動で実行ファイルを抽出する必要があり，労力を要する．

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，動的解析によらず短時間で自動的に抽出することができれば，exploit の動作条件が判明しない場合でも，マルウェア本体の解析を迅速に実施することが可能になるものと考えられる．また，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することができれば，早期にマルウェアか否かの見当をつけることができるという副次的効果も期待できる．そこで本研究の目的を，exploit を動作させずに悪性文書ファイルから実行ファイルを短時間で自動的に抽出することとする．

3. 関連研究

本研究では，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，本研究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と，それ以外のファイルも分析できる手法に分類される．

3.1 実行ファイルを分析する手法

文献 [5], [6] 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することによって，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 [7] 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 [8] 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．

本研究では，悪性文書ファイルに埋め込まれた実行ファイルを検出し，それを自動的に抽出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．

3.2 実行ファイル以外も分析できる手法

文献 [9] 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，悪性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 [10] ではバイナリデータの統計分析に加え，動的解析を組み合わせて悪性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．

本研究では，exploit を含む不正なコードではなく，エンコードされた実行ファイル（マルウェア本体）を直接抽出することを目的としている．マルウェア本体を抽出するためには，実行ファイルが埋め込まれている正確な位置やエンコード方式も明らかにする必要がある．

4. 実行ファイルのエンコード方式

4.1 主なエンコード方式

実行ファイルは様々な形式でエンコードされ，文書ファイルに埋め込まれる．2012 年に複数の組織で採取した検体（マルウェア）を分析した結果判明した，実行ファイルの主なエンコード方式を**表 1** に示す．表 1 の中でも最も頻繁に用いられるエンコード方式は，実行ファイルと任意の 1 byte の鍵を XOR 演算する方式である．XOR の場合にはエンコード演算とデコード演算は同一となるが，他の算術演算や論理演算を用いる形式ではエンコードとデコードは逆の演算となる．実際のマルウェアは，複数の方式を組み合わせてエンコードされる場合もある．エンコード方式には他にも，表示可能な ASCII 文字の範囲でのシフト（シーザ暗号），Base64，ストリーム暗号を用いる方式等がある．

4.2 NULL-Preserving XOR

実行ファイルと鍵の XOR 演算によるエンコード方式はさらに，通常の XOR による方式と NULL-Preserving XOR による方式に分類される．

表 1 実行ファイルの主なエンコード方式
Table 1 Main methods to encode executable files.

エンコード方式	説明
XOR	排他的論理和
ADD, SUB	加算または減算
ROL, ROR	左シフトまたは右シフト
Multi Byte	多バイト長の鍵による XOR

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，実行ファイルにおける NULL の部分は鍵と同一の値となる．よって，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．**図 1** に通常の XOR でエンコードされた実行ファイルのバイナリデータの例を示す．図中の文字列は，悪性文書ファイル（バイナリデータ）の一部を 16 進数で表示したものである．図中の破線で囲まれた部分は，悪性文書ファイルに埋め込まれた実行ファイルの先頭の部分に該当する．破線で囲まれた部分を見ると，AC という文字コードが多く分布していることが確認できる．ゆえに，実行ファイルの先頭部分には NULL が多く含まれるという特徴から，この実行ファイルは AC という鍵で，通常の XOR によりエンコードされたものと推定できる．

一方の NULL-Preserving XOR では，エンコードに用いる鍵の値と一致する部分，または NULL の部分には XOR 演算を実施しないため，一見して鍵が分かりにくいという特徴がある．**図 2** に NULL-Preserving XOR でエンコードでされた実行ファイルのバイナリデータの例を示す．図中の文字列は図 1 と同様に，悪性文書ファイル（バイナリデータ）の一部を 16 進数で表示したものである．図 1 と比較すると，図 2 からは NULL 以外の，特定の多く分布する文字コードを確認することができないため，一見してエンコードに用いた鍵を推定することが困難であることが確認できる．

4.3 Rolling XOR

これまでに示した方式では，エンコードに用いる鍵は同

```
AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
AC AC AC AC AC AC AC AC E1 F6 3C AC AF AC AC AC
A8 AC AC AC 53 53 AC AC 14 AC AC AC AC AC AC AC
EC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
AC AC AC AC 7C AC AC AC A2 B3 16 A2 AC 18 A5 61
```

図 1 通常の XOR でエンコードでされた実行ファイルの例
Fig. 1 An example of an executable file encoded with XOR.

```
37 34 35 30 66 66 37 35 37 30 35 36 66 66 37 35
36 34 66 66 35 35 32 30 38 31 65 62 30 30 30 34
30 30 30 30 38 33 66 62 30 30 37 66 61 38 66 66
37 35 36 34 66 66 35 35 32 38 38 31 63 34 31 30
C8 DF 15 00 86 00 00 00 81 00 00 00 7A 7A 00 00
3D 00 00 00 00 00 00 00 C5 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 4D 00 00 00
```

図 2 NULL-Preserving XOR でエンコードされた実行ファイルの例
Fig. 2 An example of an executable file encoded with NULL-Preserving XOR.

一の値であった．鍵が同一の値かつその長さが短い場合には，OfficeMalScanner [11] のようなあらゆる鍵を総当たりで試す機能を有するツールで，容易に鍵を特定することが可能である．これに対し，鍵の値が順次変化する Rolling XOR と呼ばれるエンコード方式で埋め込まれた実行ファイルも確認されている．たとえば，最初の 1 byte 目が 16 進数の AC という値の鍵でエンコードされ，2 byte 目が AD，3 byte 目が AE でエンコードされている場合，鍵の値は 1 ずつインクリメントしているものと推定できる．この例の場合には，256 byte ごとに鍵の値は初期の値に戻る．いい換えると，この場合の Rolling XOR の鍵の周期は 256 ということになる．これは，256 byte の長さの鍵を用いた XOR 演算と同義となる．したがって，Rolling XOR は多バイト長の鍵による XOR ともいえる．

4.4 2 byte の ASCII 文字

演算によるエンコード方式のほかにも，2 byte の ASCII 文字にエンコードする方式で埋め込まれた実行ファイルも確認されている．ASCII 文字でエンコードされた実行ファイルの例を**図 3** に示す．図中の左側（HEX）は図 1 および図 2 と同様に文書ファイル（バイナリデータ）の一部を 16 進数で表示したものであり，図中の右側（ASCII）は左側の 16 進数に対応する部分を表示可能な ASCII 文字で示したものである．右側（ASCII）の 2 行目の右端部分から，表示可能な ASCII 文字でエンコードされた文字列を確認することができる．この文字列は，0〜9 または A〜F の値から構成されており，2 文字で元のエンコードされた実行ファイルの 1 byte を示している．よってこの方式でエンコードした場合，元の実行ファイルの 2 倍の容量が必要になるため，文書ファイルの容量が大きくなるという特徴がある．

4.5 転置

演算によるエンコード方式は，元のデータを位置は変えずに別のデータに変換する換字方式である．これに対し，元のデータの順序を並べ替える転置方式によるエンコードもある．実際に，前後 1 byte のデータの順序を入れ替えた転置方式によってエンコードされた実行ファイルが確認されている．本稿では，この転置方式を SWAP と呼ぶ．

HEX

```
20 54 4F 43 20 48 65 61 64 6C 6F 63 6B 65 64 30
5C 2A 5C 64 61 74 61 73 74 6F 72 65 20 66 32 65
35 32 66 62 66 62 63 62 66 62 66 62 66 62 62 62
66 62 66 62 66 34 30 34 30 62 66 62 66 30 37 62
66 62 66 62 66 62 66 62 66 62 66 62 66 66 66 62
66 62 66 62 66 62 66 62 66 62 66 62 66 62 66 62
66 62 66 62 66 62 66 62 66 62 66 62 66 62 66 62
66 62 66 62 66 62 66 62 66 62 66 62 66 62 66 62
66 62 66 62 66 62 66 62 66 62 66 62 66 62 66 62
66 62 66 62 66 34 66 62 66 62 66 62 66 62 31 61
30 30 35 62 31 62 66 30 62 62 36 37 32 39 65 30
37 62 65 66 33 37 32 39 65 65 62 64 37 64 36 63
```

ASCII

```
TOC Heading;}}{
¥*¥datastore f2e
52fbfbcbfbfbfbbb
fbfbf4040bfbf07b
fbfbfbfbfbfbfbfb
fbfbfbfbfbfbfbfb
fbfbfbfbfbfbfbfb
fbfbfbfbfbfbfbfb
fbfbfbfbfbfbfbfb
fbfbf4fbfbfbfb1a
005b1bf0bb6729e0
7bef3729eebd7d6c
```

図 3 ASCII 文字でエンコードされた実行ファイルの例

Fig. 3 An example of an executable file encoded with ASCII character string.

5. 自動抽出ツールの試作

マルウェアの解析を迅速に実施するため，これまでに示した実際に確認されているエンコード方式に対応した，実行ファイルを自動的に抽出するツールを試作する．

5.1 動作の概要

試作する自動抽出ツールの動作の概要を**図 4** に示す．自動抽出ツールは doc 形式，xls 形式，pdf 形式等の文書ファイルを入力として受け付け，実行ファイル固有のパターンを検索し，発見した実行ファイルを出力する．STEP1 ではまず，2 byte の ASCII 文字にエンコードする方式で埋め込まれた実行ファイルを検索するため，一定の長さ以上の ASCII 文字列を抽出する．さらに，抽出した文字列をバイナリコードに変換し，そのバイナリコードから，指定した長さのあらゆる鍵を総当たりで試す方式で実行ファイルに含まれる固有のパターンを検索する．STEP2 では文書ファイル全体から，指定した長さのあらゆる鍵を総当たりで試す方式で実行ファイル固有のパターンを検索する．最後に，STEP3 では文書ファイル全体から繰返し文字列のパターンを検出し，多バイト長のエンコード鍵の推定を試みる．STEP1 から STEP3 のいずれの場合においても，実行ファイル固有のパターンを発見した場合には，実行ファイルをバイナリの形式で出力する．

5.2 総当たり方式による鍵の検索

実行ファイルを検索する手法として，指定した長さのあ

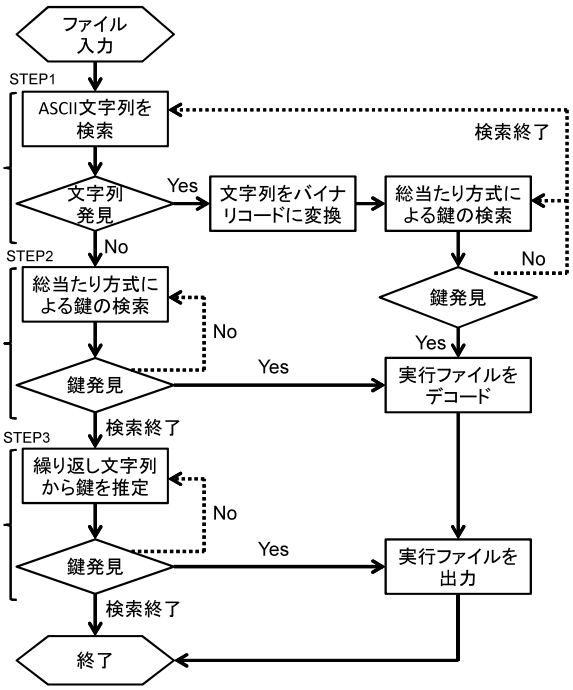

図 4 自動抽出ツールの動作

Fig. 4 The algorithm of the automated tool.

らゆる鍵を総当たりで試す方式を用いる．Windows の実行ファイルに含まれる “This program cannot be run in DOS mode” または “This program must be run under Win32” といった文字列を生成した鍵でエンコードし，エンコードした文字列を対象とする文書ファイルから検索する．総当たり方式による鍵の検索のプロセスでボトルネックとなるのは，すべての考えられるエンコード鍵を生成するループ内における処理である．よってまず “This” という短い文字列を検索し，ヒットした場合にのみ全文字列を検索する 2 段階検索を実施することで処理の高速化を図る．実行ファイルに含まれる文字列を発見した場合には，そのときの生成した鍵で文書ファイル全体をデコードする．次に，デコードした文書ファイルから実行ファイルの先頭部分を示す “MZ” という文字列を検索し，実行ファイルの位置を特定する．なお，“MZ” という文字列を発見した時点で，エンコード方式およびエンコード鍵の値を特定し，実行ファイルを発見したと判定する．最後に，発見した実行ファイルをバイナリの形式で出力する．

5.3 繰返し文字列の検出による鍵の推定

総当たり方式による鍵の検索では，多バイト長の鍵の検索に時間を要してしまい，実行ファイルを短時間で抽出することができない．したがってそれを緩和するために，文書ファイルから繰返し文字列を検出し，多バイト長の鍵の推定を試みる．以下，具体的な手順を示す．

まず，文書ファイルを 256 byte の長さで分割し，256 byte の値を順番に並べたリストを作成する．次に，連続するリストの値を比較し，一致するリストの値を 256 byte 長の鍵の候補として蓄積する．以後，分割する文書ファイルの長さ（鍵候補の長さ）を半分とし，分割する文書ファイルの長さが 2 byte になるまで同手順を繰り返す．最後に，蓄積された鍵の候補を用いて文書ファイルをデコードし，実行ファイルを検索する．以後の手順は，総当たり方式による鍵の検索において，実行ファイルに含まれる文字列を発見した場合の処理と同様である．

5.4 NULL-Preserving XOR の自動判定

図 4 において実行ファイルをデコードする際に，通常の XOR によるエンコードか NULL-Preserving XOR によるエンコードかの自動判定を試みる．まず，実行ファイルの最初の 64 byte 内における最頻出の文字コードの数を数える．このとき，NULL-Preserving XOR では NULL の部分には XOR 演算を実施しないため，図 2 の破線で囲まれた部分のように，最頻出文字コードの値は NULL となっているものと考えられる．一方の通常の XOR 演算では，図 1 のように最頻出文字コードの値はエンコードに用いる鍵自身の値となっているものと考えられる．よって実行ファイルの最初の 64 byte 内の最頻出文字コードの数が 32 byte 以上であり，かつその文字コードが NULL の場合には，NULL-Preserving XOR と判定することとする．

表 2 自動抽出ツールの仕様
Table 2 The specification of the automated tool.

OS	Windows XP
実装言語	Python 2.7
対応ファイル形式	doc, xls, pdf 等
対応エンコード方式	XOR, NULL-Preserving XOR Rolling XOR, 2 byte ASCII Multi Byte
検索可能な組み合わせ	ADD, SUB, ROR, ROL SWAP
検索可能鍵長	1 byte〜指定可能

5.5 試作したツールの仕様

これまでに示した手法を採用し，最終的に試作した Handy Scissors の仕様を**表 2** に示す．Handy Scissors の動作確認は Windows XP の Python 2.7 で実施したが，Python が動作する環境であれば他の OS でも動作可能である．対応ファイル形式については，doc 形式，xls 形式，pdf 形式の 3 種類のファイル形式で動作を確認しているが，ファイル形式固有の処理は実施していないため，他のファイル形式も検査することは可能である．エンコード方式については，通常の XOR 演算，NULL-Preserving XOR，鍵の値が変化する Rolling XOR，2 byte ASCII に加え，多バイト長の鍵によるエンコードにも短時間で対応可能である．各対応エンコード方式においては，加算および減算の算術演算，左シフトおよび右シフトの論理演算の組合せも同時に検索する．これらの算術演算および論理演算の組合せの検索は，Rolling XOR の鍵の値の変化にも適用される．さらに，転置の一種である SWAP も同時に検索する．

6. 実験

6.1 実験内容

Handy Scissors の性能を評価するため，実際に実行ファイルが埋め込まれた悪性文書ファイルを入力して結果を分析する．実験の対象となる文書ファイルの概要を**表 3** に示す．これらの文書ファイルは，2012 年に複数の組織において受信した不審なメール（送信者や内容に心当たりがない）に添付された固有のハッシュ値を持つ検体である．検体に偏りが生じるのを防ぐため，検体の採取期間は 2012 年 1 月から 9 月とし，拡張子で機械的に選定した．表 3 の左側のファイルは，あらかじめ不審な挙動を示す実行ファイルが埋め込まれていることが確認されているマルウェアである．これらのマルウェアを Handy Scissors に入力し，自動抽出の成功率，XOR と NULL-Preserving XOR の自

表 3 検体の概要
Table 3 A summary of the specimens.

	マルウェア		マルウェアではない	
拡張子	検体数	平均容量（KB）	検体数	平均容量（KB）
doc	71	257.7	14	243.2
xls	16	198.5	2	44.0
pdf	16	221.8	17	265.7
合計	103	243.0	33	242.7

表 4 実験環境
Table 4 An experimental environment.

CPU	Core2 Duo 2.4 GHz
Memory	DDR2 SDRAM 3.0 GB
OS	Windows XP SP3
Interpreter	Python 2.7.3

表 5 自動抽出の成功率
Table 5 Successful rates of automated extracting.

拡張子	成功数	成功率（%）	平均実行時間（s）
doc	68	95.8	4.59
xls	14	87.5	3.84
pdf	6	37.5	2.62
合計	88	85.4	4.17

動判定の成功率および平均実行時間を求める．あらかじめ指定する検索鍵長については 1 byte とする．また，Handy Scissors の自動抽出の成功率と，採取した当時の最新のパターンファイルを適用した大手ベンダのウイルス対策ソフトの検知率および OfficeMalScanner [11] の検知率を比較する．

表 3 の右側のファイルは，あらかじめ不審な挙動をしない（マルウェアではない）ことが確認されている文書ファイルである．これらのマルウェアではない文書ファイルを Handy Scissors に入力し，誤抽出がないことを確認する．実験を実施する環境は**表 4** に示すとおりである．

6.2 実験結果

検体の拡張子ごとの自動抽出の成功率を**表 5** に示す．自動抽出の成功率は全体で 85.4%であり，特に doc 拡張子において 95.8%の高い成功率が得られた．しかしながら，pdf 拡張子では 37.5%の低い成功率であった．平均実行時間は約 4.2 s であり，大容量のファイルにおいて実行ファイルの抽出処理が発生する場合には 10 秒以上かかる場合もあった．なお，実行ファイルの抽出処理が発生しない場合の平均実行時間，つまり，検索処理のみに要する時間の平均値は 2.0 s であった．

自動抽出に成功した 88 体の検体のエンコード方式の概要は**表 6** に示すとおりである．自動抽出に成功した検体のうち，通常の XOR でエンコードされていた検体は 24 体，NULL-Preserving XOR でエンコードされていた検体は 9 体であった．XOR と NULL-Preserving XOR の自動判定の成功率は 100%であり，誤判定は発生しなかった．よって，XOR と NULL-Preserving XOR の自動判定の精度は良好であるといえる．その他のエンコード方式としては，論理シフトを組み合わせた XOR 演算によってエンコードされた検体が 25 体確認された．Rolling XOR でエンコードされていた検体は 24 体であり，このうちの 15 体は SWAP も併用されていた．多バイト長の鍵でエンコードされた検体に関しては 6 体確認され，その鍵の長さは 4 体が 256 byte，もう 2 体が 8 byte であった．また，88 体の検体から自動抽出した実行ファイルのうち，実行ファイルとして正常に動作したものは 86 体であった．

表 6 検体のエンコード方式
Table 6 Encoding methods of the specimens.

エンコード方式	検体数
XOR	24
NULL-Preserving XOR	9
XOR + ROR or ROL	25
Rolling XOR	24
Multi Byte	6
成功数合計	88

表 7 ウイルス対策ソフトとの検知率の比較
Table 7 Comparing detection rates with antivirus softwares.

	検知数	検知率（%）
Handy Scissors	88	85.4
T 社 AV	29	28.2
S 社 AV	21	20.4
M 社 AV	25	24.3
T, S, M 社 AV	55	52.9

次に，Handy Scissors の自動抽出の成功率と，大手ベンダのウイルス対策ソフトの検知率の比較結果を**表 7** に示す．標的型攻撃に対しては，最新のパターンファイルを適用した大手ベンダのウイルス対策ソフトでも，20.4%から 28.2%の低い確率でしかマルウェアを検出することができなかった．しかも，ウイルス対策ソフトで検知できるマルウェアの種類には重複があったため，3 種類のウイルス対策ソフトを組み合わせた場合でも，検知率は 5 割程度であった．これに対し，Handy Scissors はパターンファイルを必要としないにもかかわらず，ウイルス対策ソフトの約 3 倍から 4 倍程度の高い確率でマルウェアを検出することができた．

Handy Scissors の自動抽出の成功率と，OfficeMalScanner の検知率の比較結果を**表 8** に示す．OfficeMalScanner は，一般的なシェルコードのパターンを検索する SCAN

表 8 OfficeMalScanner との検知率の比較
Table 8 Comparing detection rates with OfficeMalScanner.

	検知数	入力数	検知率（%）
Handy Scissors	23	25	92.0
OfficeMalScanner-0.56	20	25	80.0
Handy Scissors	59	62	95.2
RTFScan-0.2	44	62	71.0

オプションおよび，総当たりで実行ファイル等を検索する BRUTE オプションを使用して実行した．また，RTF のシグネチャが検知された場合には，OfficeMalScanner に同封されている RTFScan を，SCAN オプションを使用して実行した．表中の Handy Scissors の検知数は，対応する OfficeMalScanner または RTFScan と同じ検体を対象とした場合の自動抽出の成功数を示す．OfficeMalScanner または RTFScan の検知数は，検体を入力した場合に Malcious と判定された数を示す．Handy Scissors では，いずれの場合でも 9 割以上の高い確率で実行ファイルを抽出することができたのに対し，OfficeMalScanner の不審な挙動の検知率は 7 割〜8 割であった．このことから，単純に検索処理のみを比較した場合でも，OfficeMalScanner よりも高い確率で悪性文書ファイルを検知できることが確認できた．

マルウェアではない文書ファイル 33 体については，誤抽出は発生しなかった．ゆえに，Handy Scissors の誤抽出の可能性はきわめて低いものと考えられる．

7. 考察

7.1 自動抽出に失敗した原因

Handy Scissors が自動抽出に失敗した検体を分析した結果，失敗の原因は以下の 3 点に集約された．

- 対応していないエンコード方式が利用されている．
- 実行ファイルのヘッダが加工されている．
- 圧縮または暗号化されている．

まず最初に失敗の原因としてあげられるのは，対応していないエンコード方式が利用されている場合である．Handy Scissors が対応していないエンコード方式で埋め込まれていた場合には，実行ファイルを発見することができない．また，実行ファイルが多重にエンコードされて埋め込まれている場合にも，実行ファイルを発見することができない．よってこれらの場合には，実行ファイルを発見することができないため，自動的に抽出することはできない．

次の原因としては，実行ファイルのヘッダが加工されている場合があげられる．自動抽出に失敗した検体の中には，実行ファイルのヘッダの先頭部分が加工されているものが確認された．また，自動抽出には成功したものの，実行ファイルとして正常に動作しなかった検体もヘッダ部分が加工されていた．したがって，実行ファイルの検索に利用するヘッダ部分が加工されている場合には，実行ファイルを発見することができないため，自動的に抽出することはできない．これらの検体では，シェルコードに加工したヘッダを復元する処理が記述されていた．

他の原因としては，圧縮または暗号化されている場合が考えられる．自動抽出に失敗したほとんどの pdf 形式の検体には，Flate [12] 方式で圧縮されていたり，暗号化されていたりしたものが含まれていた．このように圧縮または暗号化されている場合にも，Handy Scissors では実行ファイルを発見することができないため，自動的に抽出することはできない．

7.2 Handy Scissors の効果

Handy Scissors は，exploit の動作条件が不明な場合でも，速やかに悪性文書ファイルから埋め込まれた実行ファイルを自動的に抽出することを可能とする．exploit を動作させることができない場合，分析者は通常，バイナリエディタ等を用いて手動で実行ファイルを抽出するため，労力が必要となる．したがって Handy Scissors により，マルウェア解析に要する時間を短縮することが可能となる．

Handy Scissors の副次的効果として，未知のマルウェアの有無を簡易にチェックする用途にも応用が可能であると考えられる．実験に用いた検体は，少なくとも採取した時点では大手ベンダの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，ほとんど検知することができない未知のマルウェアであった．それにもかかわらず，Handy Scissors はパターンファイルを用いず，大手ベンダのウイルス対策ソフトの 3 倍から 4 倍程度の高い確率で悪性文書ファイルを検出し，埋め込まれた実行ファイルを抽出することに成功した．実行ファイルが埋め込まれた文書ファイルは，必ずしも有害であるわけではないが，ほぼ間違いなく不審と考えて問題ないであろう．Handy Scissors の検索処理に要する時間の平均値はわずか 2.0 s であった．したがって，Handy Scissors をウイルス対策ソフトと組み合わせ，組織内のメールサーバで自動実行すれば，組織内に到達するメールの簡易チェックを実施することが可能である．

結局のところ，Handy Scissors が対応するエンコード方式は，ウイルス対策ソフトのパターンファイルに相当するものと考えることができる．ウイルス対策ソフトをマルウェアに直接対応するものと考えると，Handy Scissors はマルウェア本体（実行ファイル）を文書ファイルに埋め込むツール（マルウェア作成ツール）に対応するものであるといえよう．ゆえに，Handy Scissors も自動抽出の成功率を維持するためには，マルウェア作成ツールの更新に合わせ，対応するエンコード方式の更新が必要となる．しかしながら，更新はウイルス対策ソフトのパターンファイルほど頻繁である必要はない．なぜならば，マルウェア作成ツールの更新よりも，そのツールによって作成されるマルウェア

表 9 Handy Scissors と既存のツールとの違い
Table 9 Differences between Handy Scissors and other tools.

	Handy Scissors	Office Mal Scanner	Cryptam Multi tool
対応するエンコード方式	XOR, ADD SUB, ROL ROR	XOR, ADD	XOR, ROL ROR, NOT
2 byte ASCII	自動検索	RTFScan	×
Rolling XOR	自動検索	×	×
NULL-Preserving XOR	自動判定	×	×
総当たり方式	Multi Byte	1 byte のみ	単体では不可
繰返し文字列	自動検索	×	×
exe の抽出	自動	手動	自動
検体またはハッシュ提供	不要	不要	必要
シェルコード検査	×	○	×

の種類の方が，圧倒的に多いと考えられるからである．対応するエンコード方式の更新は，実質的にはマルウェア作成ツールの無力化を意味するものといえる．

7.3 既存のツールとの比較

Handy Scissors と，既存の類似の機能を有するツールである OfficeMalScanner [11] および Cryptam Multi tool [13] との違いを，実行ファイルの抽出という観点から考察する．各ツールの違いを**表 9** に示す．

対応するエンコード方式については，Handy Scissors および Cryptam Multi tool は多くのエンコード方式に対応している．OfficeMalScanner は，BRUTE オプションにより XOR および ADD の組合せを検索することができる．2 byte の ASCII 文字にエンコードする方式については，Handy Scissors および OfficeMalScanner に同封されている RTFScan が対応している．特に，Rolling XOR によるエンコード方式への対応や，NULL-Preserving XOR 方式の自動判定機能は Handy Scissors 独自の機能であり，有益であると考えられる．総当たり方式による鍵の検索機能に関しては，OfficeMalScanner では 1 byte のみであるのに対し，Handy Scissors では他バイト長の多様なエンコード方式に対応している．Cryptam Multi tool については，単体では総当たり方式による鍵の検索機能は有していない．Handy Scissors は総当たり方式による鍵の検索機能に加え，繰返し文字列の検出による鍵の推定機能も有している．実行ファイルの抽出については，OfficeMalScanner では多くの場合に手動で実施する必要があるのに対し，Handy Scissors および Cryptam Multi tool は自動で実施することが可能である．Handy Scissors および OfficeMalScanner は，スタンドアロン環境で動作することが可能であるが，Cryptam Multi tool はハッシュ値か検体をアップロードする必要があるため，スタンドアロン環境で完全な機能を利用することができない．Handy Scissors を用いれば，スタンドアロン環境で安全にマルウェアの解析を実施することが可能である．

このように，Handy Scissors は，実行ファイルの抽出という観点では，既存の類似のツールである OfficeMalScanner や Cryptam Multi tool と比較して，多くの長所を持っていることが確認できる．ただし，Handy Scissors および Cryptam Multi tool は，OfficeMalScanner のような一般的なシェルコードのパターンの検索機能は有していない．OfficeMalScanner は実行ファイルの抽出以外にも複数の検索機能を有しているのに対し，Handy Scissors および Cryptam Multi tool は実行ファイルの抽出に特化している点に特徴がある．したがって，各ツールのこれらの特徴に留意しつつ，分析を実施する必要がある．

8. おわりに

本稿では，標的型メールに添付された悪性文書ファイルから実行ファイルを検索し，自動的に抽出するツールである Handy Scissors を試作した．さらに，実験によってその性能を測定し，その結果および効果について考察した．また，Handy Scissors とウイルス対策ソフトおよび既存の類似の機能を有するツールを比較し，その優位性を示した．

今後の課題としては，自動抽出の成功率の向上があげられる．自動抽出の成功率を向上させるためには，自動抽出に失敗した原因をよく分析し，検証する必要がある．Handy Scissors は，本稿で示したすでに判明しているエンコード方式にはすべて対応している．よって実行ファイルが埋め込まれているにもかかわらず，自動抽出に失敗した検体や，実行ファイルのヘッダが加工されていた検体については，エンコード方式の解明が必要である．検体に含まれたシェルコードを解読し，エンコード方式を特定することができれば，それを用いて Handy Scissors を改良することで，自動抽出の成功率を向上させることができる．エンコード方式を解明するためには，検体のリバース・エンジニアリングだけでなく，マルウェア作成ツールがどのようにエンコードするのかを分析することも有効である．複数のエンコード方式を組み合わせる方式についても考慮が必要であり，つねに実行ファイルのエンコード方式の傾向には留意する必要があるものと考える．

また，近年増加している Microsoft Office Open XML Formats [14] 形式のファイル等の圧縮されたファイルや，暗号化されたファイルへの対応も今後の課題である．

今回検証した doc，xls および pdf 拡張子のファイルは，2011 年に発生した標的型攻撃のメールに添付された悪性文書ファイルのほとんどを占めている [15]．ゆえに自動抽出の成功率を向上させることができれば，大半の標的型攻撃

のメールに対応することが可能になるものと考えられる．

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）(2011)，入手先 ⟨http://www.meti.go.jp/press/2011/05/20110527004/20110527004.html⟩（参照 2012-10-29）．

[2] 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）(2009)，入手先 ⟨http://www.ipa.go.jp/security/txt/2012/01outline.html⟩（参照 2012-10-29）．

[3] 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）(2009)，入手先 ⟨http://www.ipa.go.jp/security/vuln/report/newthreat200907.html⟩（参照 2012-10-29）．

[4] Capture BAT, available from ⟨http://www.honeynet.org/project/CaptureBAT/⟩ (accessed 2012-10-29).

[5] 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).

[6] 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).

[7] Lyda, R. and Hamrock, J.: Using Entropy Analysis to Find Encrypted and Packed Malware, *Security and Privacy*, Vol.5, No.2, pp.40–45, IEEE (2007).

[8] Perdisci, R., Lanzi, A. and Lee, W.: Classification of packed executables for accurate computer virus detection, *Pattern Recognition Letters*, Vol.29, No.14, pp.1941–1946 (2008).

[9] Stolfo, S.J., Wang, K. and Li, W.J.: Towards Stealthy Malware Detection, *Advances in Information Security*, Vol.27, pp.231–249 (2007).

[10] 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).

[11] Boldewin, F.: Analyzing MSOffice malware with OfficeMalScanner (2009), available from ⟨http://www.reconstructer.org/papers/Analyzing%20MSOffice%20malware%20with%20OfficeMalScanner.zip⟩ (accessed 2012-10-29).

[12] Deutsch, P.: ZLIB Compressed Data Format Specification version 3.3, Request for Comments: 1950 (1996).

[13] Cryptam Multi tool, malware tracker blog, available from ⟨http://blog.malwaretracker.com/2012/03/cryptam-multi-tool-automatic-extraction.html⟩ (accessed 2012-10-29).

[14] Microsoft Office Open XML Formats, available from ⟨http://msdn.microsoft.com/en-us/library/office/aa338205%28v=office.12%29.aspx⟩ (accessed 2012-10-29).

[15] 日本アイ・ビー・エム株式会社：2011 年下半期 Tokyo SOC 情報分析レポート，IBM - Japan（オンライン）(2012)，入手先 ⟨http://www-935.ibm.com/services/jp/its/pdf/tokyo_soc_report2011_h2.pdf⟩（参照 2012-10-29）．

三村 守　（正会員）

2001 年防衛大学校情報工学科卒業．同年海上自衛隊入隊．2008 年防衛大学校理工学研究科前期課程修了．同年海上自衛隊保全監査隊勤務．2011 年情報セキュリティ大学院大学博士後期課程修了．博士（情報学）．同年内閣官房情報セキュリティセンター出向．情報セキュリティ大学院大学客員研究員．マルウェア解析，標的型攻撃の相関分析に関する研究に従事．

田中 英彦　（名誉会員，フェロー）

1970 年東京大学大学院工学系研究科電気工学専門課程修了．工学博士．東京大学にて計算機アーキテクチャ，並列処理，人工知能，自然言語処理，分散処理，メディア処理等の教育・研究に従事．東京大学大学院情報理工学系研究科長を経て，2004 年情報セキュリティ大学院大学情報セキュリティ研究科長・教授に就任．2012 年より同学長・研究科長・教授．情報処理学会名誉員，人工知能学会論文賞，ACM SIGGRAPH'99 Impact Paper Award，人工知能学会功績賞，東京都科学技術功労者表彰，経済産業大臣表彰等受賞．情報・システム研究機構教育研究評議会評議員，日本学術会議会員，日本ネットワークセキュリティ協会（JNSA）会長，IEEE Life Fellow，東京大学名誉教授．