

複数のエンコード方式に対応した 実行ファイル自動抽出ツール

三村 守^{1,a)} 田中 英彦^{1,b)}

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ドキュメント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ドキュメントファイルの動作を解析するためには，ドキュメントファイルから速やかにエンコードされた実行ファイル（マルウェア本体）を抽出する必要がある．しかしながら，exploit の動作条件が判明せず，動的解析によって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，ドキュメントファイルから実行ファイルを自動的に抽出するツールを試作し，実験によりその性能を評価する．

キーワード：マルウェア，標的型攻撃，悪性文書ファイル，シェルコード，動的解析

An automated tool that extracts an executable file encoded with multiple methods

MIMURA MAMORU^{1,a)} TANAKA HIDEHIKO^{1,b)}

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 encoded executable file swiftly. However, if we don't know the condition that an exploit code runs normally, it is difficult to extract the executable file by dynamic analysis. In this paper, we develop an automated tool that extracts an executable file from the document files, and the experimental results show the performance.

Keywords: Malware, Targeted Attack, Malicious Document File, Shellcode, Dynamic 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%に留まっていたが，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) と呼ばれており，大きな脅威となっている．

典型的な標的型攻撃の概要を図 1 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

¹ 情報セキュリティ大学院大学
IISec, Kanagawa, Yokohama 221-0835, Japan
^{a)} dgs104101@iisec.ac.jp
^{b)} tanaka@iisec.ac.jp

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させ、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、すべてのウイルス対策ソフトのベンダが定義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。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、最新の定義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。

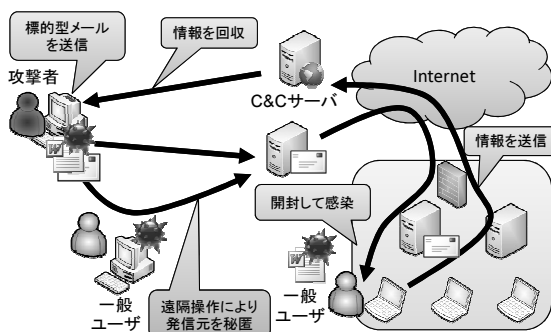


図 1 標的型攻撃の概要

2. 標的型攻撃に用いられるマルウェア分析の課題

標的型攻撃に用いられる多くのマルウェアの本体は実行ファイル (exe) であり、そのまま拡張子やアイコンが偽装される場合や、doc 形式や pdf 形式のドキュメントファイルに埋め込まれて偽装される場合がある。本稿では、マルウェアがドキュメントファイルに埋め込まれた場合を対象とする。実際に、実行ファイルと元となるダミーのドキュメント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、ドキュメントファイルからマルウェアの本体 (実行ファイル) 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、実際に隔離された安全な環境でマルウェアが埋め込まれたドキュメントファイルを実行し、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。動的解析の概要を図2に示す。動的解析では、exploit (脆弱性を利用した悪意あるコード) 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み合わせ) 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ドキュメントファイルから手動で実行ファイルを抽出する必要があるため、多大な労力となってしまう。

よって、ドキュメント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。

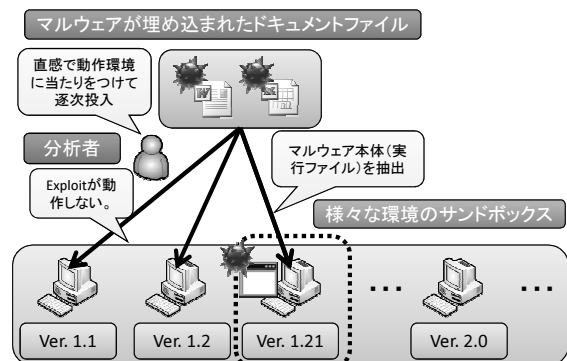


図 2 動的解析の概要

3. 実行ファイルのエンコード方式

実行ファイルは様々な形式でエンコードされ、ドキュメントファイルに埋め込まれる。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 XOR 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。

XOR 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。

通常の XOR による方式では、実行ファイルにおける NULL の部分は XOR 演算によってエンコードキーと同一

さらに、XORsearch[4]等のバイナリファイルにエンコードされた文字列を検索するツールを回避するため、2byte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5に示す。図中の左側(HEX)は図3および図4と同様にドキュメントファイル(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ASCII文字で表示したものである。右側(ASCII)の2行目の右端部分から、ASCII文字でエンコードされた16進数の部分を確認することができる。ASCII文字でエンコードされた16進数は、2文字で元の実行ファイルの1byte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、ドキュメントファイルの容量が大きくなる特徴がある。

表 1 実行ファイルの主なエンコード方式

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
ROT	表示可能文字でのシフト
Multi Byte	複数 byte の鍵による XOR

4. 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、先に示した様々な形式でエンコードされた実行ファイルを自動的に抽

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
A8	AC	AC	AC	53	53	AC	AC	14	AC	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	AC	AC	7C	AC	AC	AC	A2	B3	16	A2	AC	18	A5	61	

図 3 通常の XOR でエンコードされた実行ファイルの例

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	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 4 NULL-Preserving XOR でエンコードされた実行ファイルの例

[illegible]

図 5 ASCII 文字でエンコードされた実行ファイルの例

出するツールを試作する。

4.1 動作の概要

試作する自動抽出ツールの動作の概要を図 6 に示す。自動抽出ツールはドキュメント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、一定の長さ以上の ASCII 文字列を抽出し、抽出した文字列をバイナリに変換してパターンを検索を実施する。STEP2 ではドキュメントファイル全体のバイナリファイルに対してパターンを検索を実施する。STEP1 および STEP2 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。

4.2 実行ファイルの検索

実行ファイルを検索する手法として、指定した長さのすべての考えられる値をエンコードキーとして試すブルート・フォース攻撃を用いる。Windows の実行ファイル固有のパターンである”This program cannot be run in DOS mode”や”This program must be run under Win32”といった文字列を、生成したキーでエンコードし、エンコードした文字列を対象とするドキュメントファイルから検索す

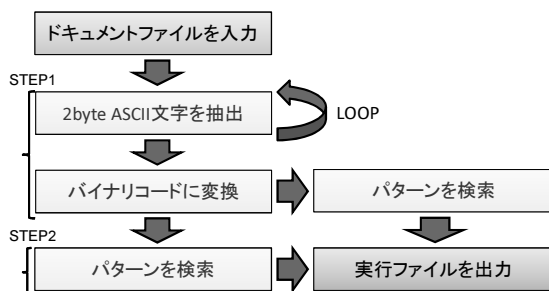


図 6 自動抽出ツールの動作

る．ブルート・フォース攻撃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キー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．

4.3 NULL-Preserving XOR の自動判定

実行ファイルの固有のパターンを発見した後，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 64byte 内における最頻出の文字コードの数を数える．この時，NULL-Preserving XOR では NULL の部分には XOR 演算を実施しないため，図 4 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 3 のように最頻出文字コードの値はエンコードキー自身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 64byte 内の最頻出文字コードの数が 32byte 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．

4.4 試作したツールの仕様

これまでに説明した手法を用いて最終的に試作したツールの仕様を表 2 に示す．試作したツールの動作確認は 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．また，対応ファイルについても同様に 3 種類のファイル形式で動作を確認したが，ファイル形式固有の処理を実施していないため，他のファイル形式にも対応は可能である．エンコード方式については，XOR 演算，ADD および SUB の算術演算，ROL および ROR の論理演算等を併用して検索することが可能である．

5. 実験

5.1 実験内容

試作した自動抽出ツール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ドキュメントファイルを入力

表 2 自動抽出ツールの仕様

OS	Windows XP
実装言語	Python 2.7
対応ファイル形式	doc,rtf,xls 等
対応エンコード方式	XOR,NULL-Preserving XOR ADD,SUB,ROL,ROR,ROT 2byte ASCII
検索可能鍵長	1byte ~ 指定可能 ADD,SUB は 4byte まで

して結果を分析する．実験の対象となるドキュメントファイルの概要を表 3 に示す．これらは 2012 年に採取した固有のハッシュ値を持つ検体（マルウェア）であり，不審な動作を実行することがあらかじめ確認されている．これらの検体を自動抽出ツールに入力し，自動抽出の成功率，XOR と NULL-Preserving XOR の自動判定の成功率および平均実行時間を求める．指定する鍵長については 1byte とする．また，同期間に採取した不審な動作を実行しない検体 18 体を入力し，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．

表 3 検体の概要

拡張子	数量	平均容量 (KB)
doc	49	280.0
xls	8	171.4
pdf	14	209.7
すべて	71	253.9

表 4 実験環境

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

5.2 実験結果

実験結果を表 5 に示す．自動抽出の成功率は全体で 71.8%であり，特に doc 形式において 87.5%の高い成功率が得られた．しかしながら，xls 形式および pdf 形式では 3 ~ 5 割の低い成功率であった．

自動抽出に成功した 51 体の検体は，すべて 1byte の算術加減算，論理シフト等を組み合わせた XOR 演算でエンコードされていた．このうち通常の XOR でエンコードされていた検体は 20 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。

1byte の鍵を検索した場合の平均実行時間は約 2.4(s) であり、実行ファイルの抽出処理が発生する場合には数秒を要した。

不審な動作を実行しない検体 18 体 (doc 9 体, xls 1 体, pdf 8 体) については、誤抽出は発生しなかった。

表 5 実験結果

拡張子	成功数	成功率 (%)	平均実行時間 (s)
doc	42	87.5	2.62
xls	4	50.0	1.89
pdf	5	35.7	2.06
すべて	51	71.8	2.43

6. 考察

6.1 自動抽出に失敗した原因

自動抽出に失敗した原因として、以下の 5 つが考えられる。

- ・ 実行ファイルが 2byte 以上の鍵でエンコードされている。
- ・ 試作したツールで対応していないエンコード方式が利用されている。
- ・ 難読化した JavaScript に埋め込まれている。
- ・ exe および dll 以外の実行形式に相当するファイルが埋め込まれている。
- ・ 実行形式に相当するファイルが埋め込まれていない。

今回の実験では、鍵の検索は 1byte しか実施していない。よって実行ファイルが 2byte 以上の長さの鍵でエンコードされていた場合には、実行ファイルを発見することができない。また、試作したツールが対応していないエンコード方式で埋め込まれた場合も同様に、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、自動的に抽出することはできない。自動抽出に失敗した検体の中には、4byte の長さの鍵で実行ファイルがエンコードされた検体や、動的解析で実行ファイルが埋め込まれていることを確認することができた検体も含まれていた。

別の原因としては、実行ファイルが難読化した JavaScript に埋め込まれている可能性も考えられる。pdf 形式において成功率が低かったのは、このケースに該当するものと考えられる。このような難読化された JavaScript に対しては、jsunpack[5] 等の別のツールを使用して難読化を解除する必要がある。

試作したツールでは、exe および dll 形式の実行ファイルに固有に含まれているパターンを検索し、実行ファイルの抽出を実施している。ゆえに、pif, scr 等の、exe およ

び dll 形式以外の実行形式に相当するファイルが埋め込まれていた場合には、試作したツールでは発見することができない。また、そもそも実行形式に相当するファイルが埋め込まれておらず、直接インターネットからマルウェア本体をダウンロードするタイプの検体も確認されている。したがって、これらの場合にも自動的に抽出することはできない。

6.2 自動抽出ツールの効果

試作した自動抽出ツールは、exploit の動作条件が不明な場合でも、速やかにドキュメント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、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の多大な労力が必要となる。したがって試作した自動抽出ツールにより、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。

試作した自動抽出ツール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、少なくとも解析時点ではほとんどのウイルス対策ソフトで検知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、自動抽出ツールはパターンファイルを用いずにある程度の確率で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ドキュメントファイルは、ほぼ間違いなくマルウェアと考えて問題ないであろう。このツールを他のツールと組み合わせ、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。

6.3 既存のツールとの比較

試作した自動抽出ツールと、既存の類似の機能を有するツールである OfficeMalScanner[6] および Cryptam Multi tool[7] との違いについて考察する。

各ツールの違いを表 6 に示す。対応するエンコード方式については、OfficeMalScanner では XOR のみ対応しているのに対し、試作したツールおよび Cryptam Multi tool は他の複数のエンコード方式にも対応している。特に、2byte の ASCII 文字の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試作したツール独自の機能であり、有益であると考えられる。また、ブルート・フォース攻撃の機能に関しても、OfficeMalScanner では 1byte の XOR のみ可能であるのに対し、試作したツールでは複数 byte の多様なエンコード方式への攻撃に対応している。Cryptam Multi tool については、単体ではブルート・フォース攻撃の機能は有していない。実行ファイルの抽出については、OfficeMalScanner では手動で実施する必要があるのに対し、試作したツールおよび Cryptam Multi

tool は自動で実施することが可能である．試作したツールおよび 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．試作したツールを用いれば，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．

表 6 既存のツールとの違い

	試作した ツール	Office Mal Scanner	Cryptam Multi tool
対応する エンコード方式	XOR,ADD SUB,ROL ROR,ROT	XOR のみ	XOR,ROL ROR,NOT
2byte ASCII	自動検索	×	×
NULL-Preserving XOR	自動判定	×	×
Brute Force	Multi Byte	1byte のみ	単体では 不可
exe の抽出	自動	×	自動
検体または ハッシュ提供	不要	不要	必要

7. おわりに

本稿では，標的型メールに添付されたドキュメント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を試作した．さらに，実験によってその性能を評価し，自動抽出ツールの効果について考察した．

今後の課題としては，自動抽出の成功率の向上が挙げられる．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．

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については，エンコード方式の分析が必要である．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，対応できるようにツールを改良することで成功率の向上が期待できる．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，常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える．

また，exe および dll 形式以外の実行形式に相当するファイルが埋め込まれている可能性についても検証が必要である．そのためには，自動抽出に失敗した検体をさらに詳しく分析する必要がある．

今回検証した doc，xls および pdf 形式のファイルは，2011 年に発生した標的型攻撃のメールに添付されたドキュメントファイルのほとんどを占めている [8]．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），入手先〈<http://www.meti.go.jp/press/2011/05/20110527004/20110527004.html>〉（参照 2012-05-31）（2011）.
- [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），入手先〈<http://www.ipa.go.jp/security/txt/2012/01outline.html>〉（参照 2012-05-31）（2009）.
- [3] 情報処理推進機構：脆弱性を利用した新たな脅威の監視・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 2012-05-31）（2009）.
- [4] Didier Stevens：XORSearch，入手先〈<http://blog.didierstevens.com/programs/xorsearch/>〉（参照 2012-05-31）.
- [5] jsunpack - a generic JavaScript unpacker，入手先〈<http://jsunpack.jeeek.org/>〉（参照 2012-05-31）.
- [6] OfficeMalScanner，Aldeid，入手先〈<http://www.aldeid.com/wiki/OfficeMalScanner>〉（参照 2012-05-31）.
- [7] Cryptam Multi tool，malware tracker blog，入手先〈<http://blog.malwaretracker.com/2012/03/cryptam-multi-tool-automatic-extraction.html>〉（参照 2012-05-31）.
- [8] 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），入手先〈http://www-935.ibm.com/services/jp/its/pdf/tokyo_soc_report2011_h2.pdf〉（参照 2012-05-31）（2012）.