Vol.2014-CSEC-66 No.6
Vol.2014-SPT-10 No.6
2014/7/3

ROP 攻撃コード検出による悪性文書ファイル特定手法の提案

田中恭之[†1,a)]　　後藤厚宏[†1]

概要：昨今の標的型攻撃では，攻撃コードを埋め込んだ悪性文書ファイルを送付し，被害者がファイルを開くことでマルウェア感染等が引き起こされ，情報の搾取等がなされるケースが多い．このとき悪用される脆弱性がゼロデイの場合，防御は非常に困難である．本稿では攻撃コードのうち，ホスト側の防御機構を突破するために付加されることの多い ROP(Return-Oriented Programming)コードを静的に検出することで，悪性文書ファイル判定を行う方式を提案し実験結果を示す．ゼロデイでは ROP は多用されるためゼロデイ検出に繋げられると考えた．

Proposal of a method to detect malicious document file based on ROP attack code detection

YASUYUKI TANAKA[†1,a)]　ATSUHIRO GOTO[†1]

Abstract: In targeted attacks of recent years, an attacker sends to victim malicious documents file with embedded attack code. As victim opens the file, malware infection occurs, Information is exploitation. If zero-day vulnerability is exploited, defense is very difficult. In this paper, we propose a method to detect statically malicious document file based on ROP (Return-Oriented Programming) attack code detection. Because ROP is often used in zero-day attack, we think to be effective in the detection of zero-day attacks.

1. はじめに

ゼロデイ脆弱性とはセキュリティ更新プログラム（パッチ）が未公開な状態の脆弱性である．シマンテック社によるとゼロデイ脆弱性の発生件数は，2011 年の 8 件，2012 年の 14 件に対して 2013 年は 22 件と近年増加傾向にある[1]．これらのゼロデイ脆弱性は標的型攻撃へ悪用されるため，社会的問題となっている[2]．標的型攻撃では，業務を装ったメールに不正プログラムを組み込んだ文書ファイルが添付されており，受信したユーザがその文書ファイルを閲覧することで，不正プログラムが実行され，最終的に企業が保有する機密情報の搾取等が行われるのが典型例である．

文書ファイルを閲覧しただけで不正プログラムを実行させるには，脆弱性を悪用する必要があるため，ユーザ環境のセキュリティアップデートを適切に行い，パッチレベルを最新化しておけばこのような攻撃の被害は受けない．しかしゼロデイ脆弱性の場合，パッチが提供されておらず無防備な状態であり，仮に脆弱性自体が公表されていない場合，危険があることすらわからない状態となってしまう．

本稿では，ゼロデイ脆弱性が標的型攻撃で用いられた場合でも検出ができることを目標として，不正プログラムを埋め込んだ悪性文書ファイルを静的解析により安全かつ効率的に検出する手法を提案し実験結果を示す．昨今のゼロデイ攻撃では後述する DEP や ASLR と呼ばれるホスト側の防御機構を突破する ROP 技法が多用されるため ROP 攻撃コードの検出によりゼロデイ検出に繋げられると考えた．攻撃コードを分析した結果，ROP 攻撃コードの目的が共通している点やシェルコード自体の暗号化対象とならない点を利用し検出を行う．提案方式を実装し他のツールと比較した実験結果と課題を示す．

†1 情報セキュリティ大学院大学
IISEC, Yokohama, Kanagawa 221–0835, Japan
a) mgs135505@iisec.ac.jp

2. 攻撃コードと防御メカニズム

2.1 悪性文書の基本的な攻撃の流れ

図 1 のように，ユーザが悪性文書ファイルを開くと，閲覧ソフトの脆弱性を攻撃するエクスプロイトコードが作動する．エクスプロイトコードは，閲覧ソフトの制御権をコントロールできるようにするまでの役割を担う．制御権が取られると，シェルコードが実行される．シェルコードは文書ファイル内に埋め込まれた実行ファイル(マルウェア)を取り出して実行する，ネットワーク上からマルウェアをダウンロードして実行する等を行う．

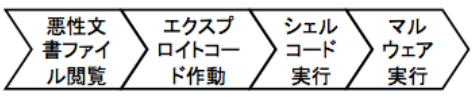

図 1 基本的な攻撃の流れ

2.2 防御メカニズム：DEP

DEP(Data Execution Prevention)とは，データ領域内でのコード実行を阻止するもので，Microsoft Windows では WindowsXP SP2 で導入された．エクスプロイトコードにより制御権を取られ配置されるシェルコードは，メモリ上のプログラム領域ではなく，スタックやヒープと呼ばれるデ

ⓒ 2014 Information Processing Society of Japan

ータ領域内に配置される．DEP 有効下では，データ領域での実行権が無くシェルコードは実行できず攻撃者は攻撃に失敗する．

2.3 DEP を回避する ROP 等 Code-Reuse 攻撃

DEP を回避するため Return-to-libc という技法が用いられるようになった．これはメモリ中に存在する API 関数を直接コールする手法で，API 関数への引数を適切にスタックに積んでコールすることで API 関数を動作させ目的を達成することができる．さらにこの技法を進めた ROP(Return Oriented Programming)[3]という技法が最近主流となっている．ROP 技法を用いたコードの一例を図 2 に示す．ROPgadget とは，実行可能領域にある ret 命令で終了する数バイトのコード断片である．この例では，最終的にアドレス 0x50505050 に値 0xDEADBEEF を格納することを実現している．まず攻撃者は目的が達成できるように適切にスタックを積んでおく．スタックの最上位の値にリターンするように調整し，最初に gadget No1 が実行される．レジスタ EAX には意図した値(0x50505050)が格納され，ret 命令でスタックを参照し，次の gadget No2 にリターンする形で gadget No2 が実行される．このようにして，最終的に gadget No3 が実行されることで目的が達成できる．ROP を進めた技法として JOP(Jump Oriented Programming)[4]が提案されている．ret の代わりに jmp を用いる物だが，gadget をコントロールする Dispatcher を用意し，jmp 先をこの Dispatcher にするように工夫している．また SOP(String Oriented Programming)[5]はフォーマットストリング脆弱性を利用する技法である．ただ本稿執筆時では JOP や SOP については具体的な攻撃コードを目にすることは無かった．尚 ROP,JOP,SOP は Code-Reuse 攻撃とも呼ばれる．

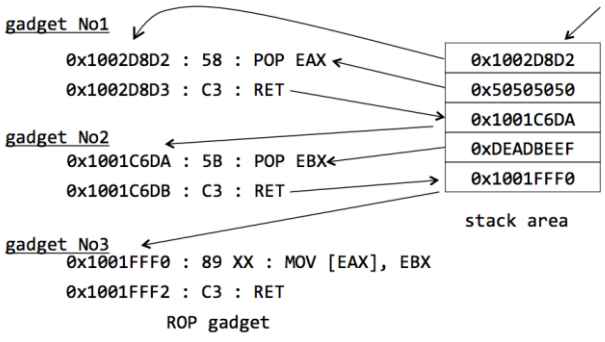

図 2 ROP コードの動作原理

2.4 防御メカニズム：ASLR

ASLR(Address Space Layout Randomization)はアドレス空間を OS 起動時にランダム化するもので，WindowsVista から導入されている．Return-to-libc や ROP で見たように，攻撃者が API 関数やスタック・ヒープの固定な既知アドレスの利用することに対抗した方式である．

2.5 ASLR の回避

攻撃者は当該 OS が 32bitOS である場合は，ランダム化されたアドレス空間をスキャンして必要なアドレスを見つけ出すことによって，現実時間・現実試行回数内で ASLR を回避可能である．また dll によってはランダム化が行われない物が存在し，これを悪用した手法が最近の主流となっている．さらに，非 ASLR な dll しか得られない場合でも，脆弱性を利用して特定の dll のベースアドレスを得ることで動的に ROP 攻撃コードを組み立てる手法も登場している．

2.6 防御メカニズムを回避する悪性コード

防御・検出メカニズムを回避するために進化した悪性文書ファイルは図 3 のように構成・実行される．ROP コードを用いて復号コード部に実行権を付与し DEP を回避する．DEP の回避が必要ない場合でも安定して攻撃を成功させる為に SEH コードが実行される場合が多い．SEH コードは Windows の例外処理機構である SEH(Structured Exception Handling)を悪用するコードで古くから存在するが未だ悪用が見られるものである．

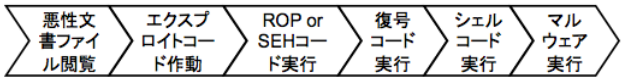

図 3 防御メカニズムを回避する悪性コード

3. 関連研究

悪性文書ファイルの検出に関する先行研究を示す．いずれもファイルを実行せずに検出する静的解析手法を用いたものである．三村らは悪性文書ファイルに埋め込まれた実行ファイル（マルウェア本体）を自動抽出する Handy Scissors を提案している[6]．複数のエンコード方式への対応や総当たり方式による鍵の探索により実行ファイルの抽出を行う．大坪らの Ochecker は Microsoft 文書ファイルのサイズや構造に関する情報を検査することで悪性文書ファイルを検知する手法を提案している[7]．いずれの手法も大手ベンダのアンチウイルスソフトの検知率と比較して高い検知率で検出できており検査処理に要する時間も短い．鍵の探索機能を持つツールとして OfficeMalScanner がある[8]．鍵の探索やエンコード方式のバリエーションは HandyScissors に劣るが OfficeMalScanner の優位点はシェルコード検査機能を持っている点である．大坪らの評価でも，大坪らの方式には劣るもののアンチウイルスソフトと比較して高い検知性能を出している．

次に ROP の検出分野での先行研究を示す．ROP を防ぐには関数が呼ばれてリターンした場合に適切に呼び元のアドレスの次のアドレスに戻っているかをチェックすれば良い．これは従来から提案されている方式で，L. Davi らの ROPdefender では，shadow stack という領域を本来の stack 領域とは別にもうける．この領域を用いリターンアドレスを記録し適切にリターンしているかをチェックする．この方式で ROP の検出は可能となるが，shadow stack の管理の

ⓒ 2014 Information Processing Society of Japan

オーバヘッドが大きくなり性能に影響が出る[9].

そこで V. Pappas らの kBouncer では，最近の Intel プロセッサで提供される機能である LBR（Last Branch Recording）を用い ROP 検出を実現している[10]．LBR には過去 16 回分と限られはするが直近の分岐情報が記録されている．これの利点は LBR は CPU が提供するレジスタであるため，この記録にかかる性能劣化はゼロと見なすことができる点である．この LBR の履歴を活用し ROP を検出する．具体的には, 攻撃者は最終的に API コールやシステムコールを目的とするのでその時点で過去の分岐履歴をみて正常なコードか ROP かを判定する．LBR の制約から履歴は 16 個に限られるが，V. Pappas らの調査結果では，API コール，システムコールを目的とした ROP gadget は通常 10 個程度であるため問題ないと結論づけている．

4. 提案方式

本提案では悪性文書か否かの判定に，ホスト側の防御機構を突破する ROP 攻撃コードを静的に特定することで判断する手法を提案する．図 3 に示すように ROP 攻撃コード部はシェルコートの暗号化ルーチンの外部に置かれることに着目している．これは復号コード自体に実行権限の付与が必要な為である. このため ROP コードはシェルコードの暗号化ロジックの内部に入らない．2.6 節で示した悪性コードが組み込まれた悪性文書ファイルの一例を図 4 に示す．いずれかのコードを検出できれば悪性文書ファイルと特定できるためコード部分毎に考察を行う．

- エクスプロイトコード部

脆弱性を発動させるために特徴が現れやすく意図的な暗号化は難しいため特定がしやすいが，ゼロデイのように未知の脆弱性の場合は特定ができない.

- ROP もしくは SEH コード部

DEP 回避や安定動作を目的とする．後続する復号コード部の外に配置され，復号コードの実行権を付与する．復号コード部の外に配置されることからシェルコードの暗号化対象とはならない．ROP コードは 100Byte 程度，SEH コードは 10Byte 程度であり，SEH コードについては短いため特徴を捕らえるのは困難と考え, 本研究では ROP コードの特徴を捕らえることとした.

- 復号コード部

暗号化シェルコードを復号する目的で数 10Byte 程度からなる．XOR 等の論理演算を用いる単純な物が多く，正常なコードと区別が困難である．ポリモーフィックコードを生成するエンコーダの場合，サイズが大きくなり特徴を捕らえられる可能性があるが本研究では対象としなかった．

- シェルコード部

シェルコードはいくつかの共通した特徴を持つ[11]．その中でも多くのコードは API 関数アドレスの自己解決を行う為に PEB(Process Environment Block)を参照するため判定が可能である．しかし容易に暗号化や難読化が行われ，特に複数回暗号化を行う手法であるマルチエンコーディングが行われると検出が困難となる．

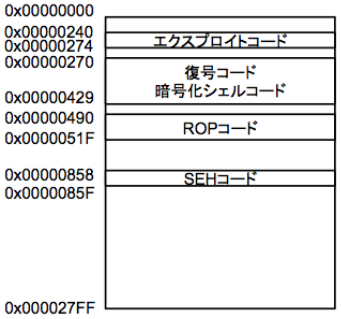

図 4 悪性文書ファイルの例

4.1 ROP コードの内部構成

ROP コードは 2.3 節で示したように実行権限のあるメモリ領域のコード部分を繋いで利用することで DEP を回避して任意のコードを実行可能とする技法である．一般に流通する攻撃コード*を調査した結果, 攻撃コードとして ROP 手法が用いられる場合，共通した目的があることが判明した．ROP を利用して自由度の高い任意のシェルコードを書くのではなく後続するコード領域に実行権限付与するのみであった．これは 2.1 節で言及したシェルコードのサイズ要求が厳しいことに起因する．Windows では関数をコールし実行権限を付与する．実行権限を付与可能な関数例を表 1 に列挙する. また VirtualProtect 関数について表 2 に示す. 次に VirtualProtect 関数を用いた ROP コードがスタックに積まれた状態例を図 5 に示す.

表 1 実行権限付与関数例

関数名	
VirtualProtect	SetProcessDEPPolicy
VirtualAlloc	WriteProcessMemory
HeapCreate	NtSetInformationProcess

表 2 VirtualProtect 関数

名前	説明
lpAddress	アクセス権限を変えたいページ領域のアドレス
dwSize	領域のサイズ
flNewProtect	アクセス権限．実行権を付与するには 0x40(PAGE_EXECUTE_READWRITE)

図 5 で DEP を制御する表 1 表 2 に示す関数の物理アドレス（図では VirtualProtect 関数）に関する ROPgadget コードを濃いグレー，これらの関数に適切な引数等を準備するのに用いられる ROPgadget コードを通常の ROPgadget コードとして薄いグレー，関数等への引数を白で示す．攻撃者はスタックにこの 2 種類の ROPgadget コード及び引数等を適切に積んでおき，ROP コードを実行し，シェルコード配置エリアのメモリ領域に実行権限を付与し DEP を回避する．DEP が回避されると複合コードの実行等次の処理が開

* The Exploit Database. http://www.exploit-db.com/

ⓒ 2014 Information Processing Society of Japan

始する．

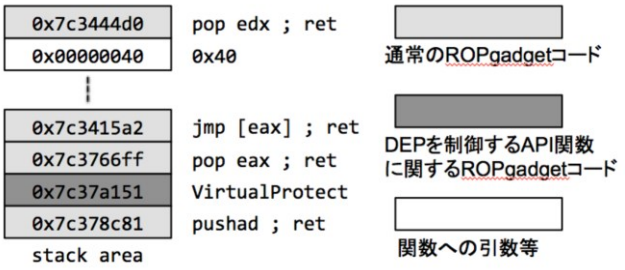

図 5 ROP コードが積まれたスタック

攻撃者は安定して攻撃を成功させるためにメモリ空間に固定値として存在する ROPgadget コードを用いようとする．2.4 節で示した ASLR が機能している場合これが困難となるが 2.5 節で示した回避方法がある．攻撃コードを調査したところ，ASLR 非対応モジュールの物理アドレスを用いる攻撃方法が多数存在し，また逆に ASLR 非対応モジュールの数は限られることから用いられる ROPgadget コードの数は限定的で，この物理アドレスを特徴文字列として検出手法が有効であると考えられる．

4.2 提案方式 1 の概要

悪性文書ファイル内で ROP 攻撃コードは，図 5 で示すスタックの状態を作る為，図 6 のように構成される．DEP を制御する関数の物理アドレスを指す ROPgadget コードを特徴文字列①と定義する．またこの DEP 制御関数への適切な引数等を準備する ROPgadget コードを特徴文字列②と定義する．ここで図 6 の□は 1byte を示し，特徴文字列①及び②は 32bit 環境の場合 4byte の物理メモリアドレスである．このとき特徴文字列①は特徴文字列②及び関数への引数等の値に挟まれるような形で悪性文書内に出現する．4 章で示したとおり，この ROP コードはシェルコードを対象とした暗号化・難読化の影響を受けないことから文書ファイル内の文字列から検出が可能である．

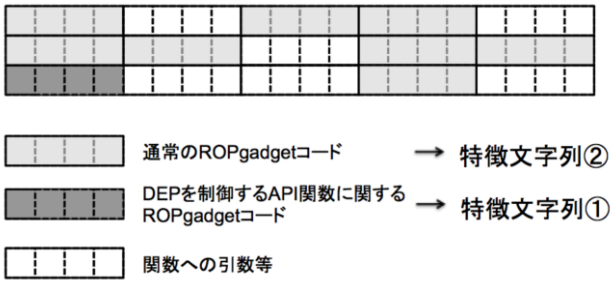

図 6 悪性文書ファイル内の ROP 攻撃コード

(1) 特徴文字列①の抽出

ゼロデイ脆弱性が多く発生するブラウザへの攻撃に ROP 技法が多用されることから，今回はプラットフォーム windows のカテゴリ browser における攻撃コード総計 221 個から ROP 技法が使われる 22 個の攻撃コードから表 1 で示した 6 つの関数を呼ぶと判断できる物理アドレスを取り出しそれらを特徴文字列①とした．結果を表 3 に示す．物理アドレスは 16 アドレス抽出された．また，対象関数は VirtualProtect と VirtualAlloc の 2 つのみであった．該当する dll 名とアプリケーション名も併記した．

(2) 特徴文字列②の抽出

特徴文字列②は特徴文字列①と同じく Metasploit Framework のプラットフォーム windows のカテゴリ browser における攻撃コード総計 221 個から ROP 技法が用いられる 22 個のコードを対象に ROPgadget となるすべてのアドレスから特徴文字列①を除外したものを取り出した．合計 500 個程度の物理アドレスであった．

4.3 提案方式 1 の具体例

これらの特徴文字列を利用して ROP 攻撃コードを検出する方式例を図 7 に示す．特徴文字列②が特徴文字列①の前後に連続して出現することでスコア値を上げていき，特徴文字列①が出現しかつスコア値が一定の閾値以上の際に検出と判断する．特徴文字列①のみの場合 False positives がおこる可能性があるので，ROP コードを組み立てる上で必要になる特徴文字列②を併用し精度を向上させる．

表 3 特徴文字列①

アドレス	関数名	DLL名	アプリケーション
0x7c37a151	VirtualProtect	msvcr71.dll	JRE1.6
0x7c37a140	VirtualProtect	msvcr71.dll	JRE1.6
0x77c11120	VirtualProtect	msvcrt.dll	XP sp3
0x77ba1114	VirtualProtect	msvcrt.dll	XP sp3
0x7c801ad4	VirtualProtect	xul.dll	FF3.6
0x6d9fc094	VirtualProtect	jvm.dll	JVM
0x6d6c227c	VirtualProtect	regutil.dll	Java Regutils
0x63f010f4	VirtualProtect	mscorie.dll	.NET2.0
0x77c1110c	VirtualAlloc	msvcrt.dll	XP sp3
0x781a909c	VirtualAlloc	mozcrt19.dll	FF7,8,9
0x51bd115c	VirtualAlloc	hxds.dll	office2007
0x51bd10bc	VirtualAlloc	hxds.dll	office2010
0x1083828c	VirtualAlloc	xul.dll	FF3.6
0x100361a8	VirtualAlloc	FoxitReaderPlugin.dll	FoxitReaderPlugin
0x1001a22c	VirtualAlloc	dwa85.dll	Lotus iNotes ActiveX
0x4401a130	VirtualAlloc	AnnotateX.dll	Quest InTrust ActiveX

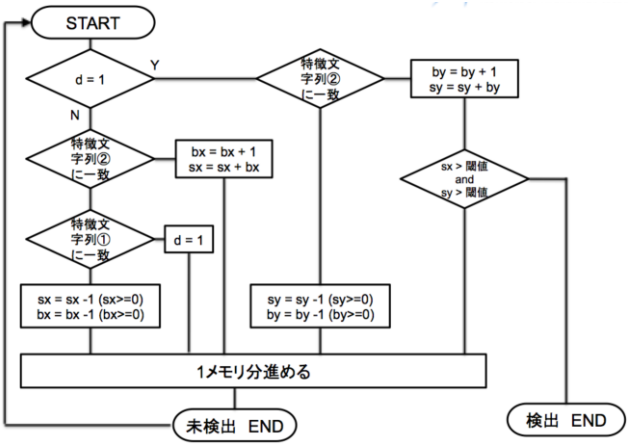

図 7 フローチャート

4.4 提案方式 2 の概要

提案方式 1 は，攻撃者が用いる既知のアドレスを特徴文

字列として蓄積しておき検出する方式であり，未知の物理アドレスは検出できない．ただ現時点では，流通する攻撃コードの多くが提案方式 1 に示すような既知のアドレスを利用しているため提案方式 1 でも一定の効果はあると考えられる．しかし，未知の ASLR 非対応物理アドレスを用いる攻撃手法，もしくは，ASLR 対応 dll でも，特定の dll のベースアドレスを動的に攻撃コード内で取得して用いる攻撃手法[12]が該当する．とくに後者の手法は，非 ASLR な dll は悪用されることが知られるようになり，ASLR 対応するように修正プログラムが早いタイミングでリリースされ，環境によっては非 ASLR な dll を見つけることができず攻撃が難しくなるため，攻撃者の立場からすると有効な手法である．このような攻撃手法に対抗するため，提案方式 2 では特定の dll 等の物理アドレスに依存しない方法で ROP 攻撃コードの検出を目指す．具体例として Metasploit から取得した JRE1.6 アプリケーションの ASLR 非対応 dll である msvcr71.dll から作られた実際の ROP 攻撃コードを図 8 に示す．

図 8 ROP 攻撃コードの例

msvcr71.dll のベースアドレスは非 ASLR であることから常に固定で，0x7C340000-0x7C396000 にロードされる．このことから作られる Gadget は上位バイトが 0x7C3 となり図のグレーで示すように 4 バイト周期で現れる．また，悪用される多くの dll はユーザ領域にロードされるものであるため，最大で 0x7FFFFFFF となることも特徴である．ここでリトルエンディアンで配置されることに注意する．また図で白で示した部分にも特徴がある．40 バイトから 47 バイト目の値が 0x00000040 であるが，これは VirtualProtect 関数での実行権を付与するのに必要な引数である．また図でグレーを除く白部分の文字列だが，word 文書ファイルでよく見られる 0x00 や 0xFF 等でなく，散らばりのある値となっている．

4.5 提案方式 2 の具体例

図 9 に疑似コードを示す．文書ファイルを先頭から 1 バイト毎にチェックする．先頭 4 バイトが ROP 被疑か否かをチェックし（行:03）被疑であれば，先頭 4 バイト含む 100 バイトをチェック対象とし（行:05），各 4 バイトの差分を取り，差分が閾値を下回る場合スコアを上げる（行:08,09）．スコアが閾値を超えた場合 ROP コードと判定する．(行:13)

```
(01) WHILE offset < file.size DO
(02)   dword = file[offset..offset+3]
(03)   IF dword > 0x7fffffff THEN offset +=1;NEXT
(04)   ELSE
(05)     dword[0..24] = [offset..offset+99]
(06)   ENDIF
(07)   FOR i = 0 to 24 DO
(08)     IF |dword[i]-dword[0] | < 0x100000 THEN
(10)       score += 1
(11)     ENDIF
(12)   ENDFOR
(13)   IF score > 10 THEN print "ROP code found."
(14)   ENDIF
(15)   offset +=1
(16) ENDWHILE
```

図 9 提案方式 2 の疑似コード

行:08 の閾値について，ROP に用いられる代表的な dll を調査した結果を表 4 に示し，この値を参考にサイズ 0x100000 未満とした．

表 4 ROP に用いられる代表的な dll 例

名前	開始アドレス	終了アドレス	サイズ
msvcr71.dll	0x7c340000	0x7c396000	0x56000
hxds.dll	0x51bd0000	0x51ca7000	0xd7000
msvcrt	0x77c10000	0x77c68000	0x58000

5. 実験

5.1 誤検出回避ロジックの追加

未知の攻撃への検出可能性のある提案方式 2 について，実験を行った．図 9 の疑似コードをスクリプト化し，悪性文書ファイル及び良性文書ファイルを対象に実験を行った．良性文書ファイルを誤検出したため，図の疑似コードに以下のロジックを追加した．

- 0x00FFFFFFFF 以下の除外

0x00 は終端文字列等で使われることが想定され，正常文書ファイル内に一定周期で現れ誤検出となっていた．図 9 の 03 行に AND 条件で加える．このことによって上位 1 バイトが 0x00 で始まるメモリ空間の ROP コードは検出できなくなるが，0x00000000-0x7FFFFFFF のなかの 1/128 であることから今回は除外条件としても効果の影響は少ないと判断した．

- 0x****0000 を除外

これも良性文書内に多くみられた．表 3 で示したように，下位 2 バイトがともにゼロとなる ROP コードは非現実かつ目にしていない点と一般に関数のベースアドレスとなる値を ROPgadget の起点するケースはほぼ無いと考えられることから除外しても影響は無いと考える．

- 1 バイト毎に同じ値が出現を除外

検査対象のバイト列（ここでは 100 バイト）について，1 バイト毎に同じ文字列が並ぶバイト列が半数以上（50 バイト）ある場合，そのような制約下で ROP コードの構築は無理と考え除外した．

- 近すぎる値の除外

図 9 の行 08 で 0x10000 未満という閾値をもうけたが，これに 0x100 以上という閾値を追加した．ROP コードを構築する上で下位 1 バイトのみを変化させて行うのは困難であるためである．

5.2 実験結果

表 5 に悪性文書に対する検出結果を示す．入手できた検体について，誤検出回避ロジックを組み込んだ提案方式を実装したスクリプト，Ochecker（以下 OC），OfficeMalScanner（以下 OM）で検出可能かを実験した．（検体 No4 については検体の入手ができなかったが[13]から一部の内容を確認できたので机上チェックに検出可能かを判断した．）また良性文書ファイルとして MS 文書ファイルを合計 10 ファイル用意し，同様に判定を行い誤検出が無いことを確認した．

No1-4 は提案方式で検出でき，OC，OM ともに検出できなかった．No1 については metasploit のシェルコードエンコードオプションで bloxor を指定している．これを外したところ OM のシェルコード検出機能で fs レジスタ(0x30)へのアクセスコードで捕らえることができた．これは多くのシェルコードで見られる特徴であり暗号化されていないコードを捕らえるのに有効な手法である[11]．OC で検出できなかった理由として，ファイル構造に問題が無い形で各種悪性コードが埋め込まれていたからであった．これは大坪らも CVE-2010-3333 として言及している[14]．本実験では CVE-2010-3970,CVE-2012-0158 の検体が検出不可となったが，これは検体の作りに依存すると考えられる．

No5 は 3 者が検出できた．OM は No1 と同様のシェルコード検出ロジックで検出したため，同様に暗号化がかかった場合は検出不可と考えられる．

No6,7 は 2014 年 3 月 25 日に発表された MS-word のゼロデイ脆弱性を利用する検体である．提案方式では検出できなかった．これは脆弱性を発動させるために array 形式でエンコードしておく必要があり図 8 に示すようなバイトコードとは形式が異なる為であった[15]．No6 について OC 及び OM は検出可能であった．OM は rtf のファイル構造異常で検出した．No7 については 3 者とも検出できなかった．

表 5 実験結果

No	ファイル名	CVE	入手先	提案	OC	OM	備考
1	msf11_006.doc	2010-3970	Metasploit	○	×	×	-e bloxor
2	44.doc	2010-3970	contagio	○	×	×	
3	msf12_027.doc	2012-0158	Metasploit	○	×	×	
4	Presi_.doc	2012-0158	securelist	○	×	×	机上チェック
5	cve_.doc	2010-3333	exploit-db	○	○	○	
6	msf14_017.doc	2014-1761	Metasploit	×	×	×	array形式
7	e378_.bin	2014-1761	malwr	×	○	○	array形式

6. 議論

本提案方式は ROP コードの静的検出に特化したものであり OC や OM 等の他の方式と組み合わせて用いることで，ゼロデイを用いるような悪性文書ファイルを判定出来る可能性があると考えるが，以下の課題がある．

- エンコードへの対応

CVE-2014-1761 のように ROP コードがエンコードされた場合，実験に示したように提案方式で検出ができない．ただ一意のルールでエンコードされる場合，図に示すアルゴリズムを改善することで検出できる可能性がある．難読化された場合は本案では対応できない．

- 64bit 対応

今回は windows32bit 環境を想定したが，類似アルゴリズムで 64bit 対応も可能と考えられる．ただ，実際に 64bit の ROP コードの流通を確認できていない．

- False Negative の改善

今回は利用しなかったが DEP 制御関数への引数や間に入る JUNK コードを特徴文字列として判断材料に加えることは FN の向上に貢献すると考えられる．引数を即値として用いないように ROP コードを組むことは可能だが，コードが長くなり脆弱性発動要件を満たさなくなる等の理由からあまり見かけないためである．

- False Positive の改善

誤検知回避ロジックとして下位バイトのエントロピーや分散値を用いて判定するとさらに精度が高くなると思われる．今回は文書ファイルを対象としたが，例えば PE ファイルを対象とした場合，コンパイラに依存するが，関数アドレス参照等で連続した値が出現する部分がある．ただ機械的に単調増加する等の特徴があるため計算量が少ない方式で見分けられる可能性がある．

- 実験検体数の増加

主に contagio*から検体入手を行ったが，ROP 技法が含まれるものは少なかった．また文書ファイルは個人・企業の秘密情報が含まれるため一般に流通しにくい点も研究を進める上での課題である．

7. まとめ

本稿では，ゼロデイ脆弱性が標的型攻撃で用いられた場合でも検出ができることを目標として，不正プログラムを埋め込んだ悪性文書ファイルを静的解析により安全かつ効率的に検出する手法を提案した．提案方式では，昨今のゼロデイ攻撃ではホスト側の防御機構を突破する ROP 技法が多用されるため，ROP 攻撃コードの目的が共通している点やシェルコード自体の暗号化対象とならない点を利用し検出を行う．提案方式を実装し他のツールと比較した実験結果と課題を示した．今後は課題を検討しながら，引き続き最新の攻撃コードやマルウェアの特徴を調査・研究し防御に繋げたい．

参考文献

[1] シマンテック：2013 年 インターネットセキュリティ脅威レ

* contagion malware dump. http://contagiodump.blogspot.jp/

ⓒ 2014 Information Processing Society of Japan

ポート 第 18 号，シマンテック（オンライン）（2014），入手先 (http://www.symantec.com/ja/jp/security_response/publications/) (参照 2014/05/30)

[2] 情報処理推進機構：2013 年版 10 大脅威 身近に忍び寄る脅威（オンライン）（2014）入手先 (http://www.ipa.go.jp/security/vuln/10threats2013.html)（参照 2014/05/30）

[3] Hovav Shacham. The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86). In Proceedings of the 14th ACM conference on Computer and Communications Security (CCS), 2007.

[4] Tyler Bletsch, Xuxian Jiang, and Vince Freeh, “Jump-Oriented Programming: A New Class of Code-Reuse Attack,” in Proceeding ASIACCS. ACM, 2011, pp. 30-40.

[5] Mathias Payer, Thomas R. Gross, “String oriented programming: when ASLR is not enough”, in Proceeding PPREW '13 Proceedings of the 2nd ACM SIGPLAN Program Protection and Reverse Engineering Workshop.

[6] 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，pp.1211-1219(Mar. 2013).

[7] 大坪 雄平，三村 守，田中 英彦：ファイル構造検査による悪性 MS 文書ファイル検知手法の検知，情報処理学会研究報告, Vol.2013-IOT-22, No.16 (2013).

[8] Boldewin, F.: Analyzing MSOffice malware with Office-MalScanner (2009), available from (http://www.reconstructer. org/papers/Analyzing%20MSOffice%20malware%20with%20OfficeMalScanner.zip) (accessed 2014/05/30)

[9] Lucas Davi, Ahmad-Reza Sadeghi, and Marcel Winandy. ROPdefender: A practical protection tool to protect against return-oriented programming. In Proceedings of the 6th Symposium on Information, Computer and Communications Security (ASIACCS), 2011.

[10] Vasilis Pappas, Michalis Polychronakis, and Angelos D. Keromytis. Transparent ROP Exploit Mitigation Using Indirect Branch Tracing. USENIX Security, 2013.

[11] 田中　恭之，後藤　厚宏：攻撃コードの特徴から見た対策の検討, SCIS2014, 4C2-4, 2014.

[12] http://www.fireeye.com/blog/uncategorized/2014/04/new-zero-day-exploit-targeting-internet-explorer-versions-9-through-11-identified-in-targeted-attacks.html (参照 2014/05/30)

[13] https://www.securelist.com/en/analysis/204792298/The_curious_case_of_a_CVE_2012_0158_exploit (参照 2014/05/30)

[14] http://www.iwsec.org/mws/2013/presentation/3B1-3-slide.pdf（参照 2014/05/30)

[15] http://h30499.www3.hp.com/t5/HP-Security-Research-Blog/Technical-Analysis-of-CVE-2014-1761-RTF-Vulnerability/ba-p/6440048 （参照 2014/05/30）

ⓒ 2014 Information Processing Society of Japan