情報処理学会論文誌 Vol.56 No.9 1693–1705 (Sep. 2015)

悪性文書ファイル内のROP攻撃コード静的判定手法

田中 恭之[1,a)] 後藤 厚宏[1]

受付日 2014年11月27日, 採録日 2015年6月5日

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．本稿では，特にここ数年の脆弱性においてホスト側の防御機構を突破することを目的として，多くの攻撃コードに付加される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，実際の検体を用いた実験結果を示す．また解析の現場においても，コストの高いROP動的解析に比べて本静的解析手法は有効である．

キーワード：マルウェア，Return-Oriented Programming，静的解析

ROP Attack Code Static Detection Method of Malicious Document File

Yasuyuki Tanaka[1,a)] Atsuhiro Goto[1]

Received: November 27, 2014, Accepted: June 5, 2015

Abstract: In recent targeted attacks, an attacker sends to victim malicious document files with embedded attack code. As the victim opens the file, malware infection occurs, then victim's confidential information is exploited. In this paper, we propose a method to detect statically malicious document files based on ROP (Return-Oriented Programming) attack code detection. We found ROP attack codes used in many exploits in order to break host defense mechanisms. We show how to statically detect these ROP attack codes, then show the results of experiments using actual samples. Our static analysis method is effective in malware analysis field because ROP dynamic analysis is difficult in general.

Keywords: Malware, Return-Oriented Programming, Static analysis

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23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攻撃コード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，攻撃コードが実行され，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．

[1] 情報セキュリティ大学院大学
Institute of Information Security, Yokohama, Kanagawa 221–0835, Japan
[a)] dgs155102@iisec.ac.jp

ユーザ環境に脆弱性がある場合，それを利用して，文書ファイルを閲覧しただけで攻撃コードを実行されてしまう．このため，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，ユーザ環境は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分からない状態となってしまう．

本稿では，対策が非常に困難であるゼロデイ攻撃対策を視野に入れ，標的型攻撃で多用される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する．次に，提案方式を実装し他のツールと比較した実験結果を通して有効性と課題を示す．本提案手法では，文書ファイルに埋め込まれたROP攻撃コードの検出により悪性文書ファイルを検出する．ROP攻撃コードは，ROP

(Return-Oriented Programming) [4] 技法によってホスト側の防御機構を突破することを目的とする攻撃コードである．Microsoft 社の資料によると，ここ数年に発生した新しい脆弱性に対する攻撃において 95%以上の割合で用いられ [19]，この中にはゼロデイ攻撃も含まれる．このため，ROP 攻撃を早期に検出することでゼロデイ対策につなげられると考えた．本手法では，実際の攻撃コードを分析した得られた特性：(1) ROP 攻撃コードの目的が共通している点，(2) シェルコード自体の暗号化対象とならない点，を利用して静的に ROP 攻撃コードの検出を行う．なお，本稿では，これ以降，この攻撃目的で用いられる ROP 攻撃コードを単に ROP コードと標記する．

本提案手法の利用シーンは次の 2 つである．1 つ目は，文書ファイル型検体解析の現場における解析者支援である．一般に，文書ファイル型検体の解析には，動的解析に加え詳細な機能を知るための人手による静的解析が行われる．動的解析では安全面の考慮に加えパッチレベル等環境を変化させ攻撃が発動する環境を探す必要がある．一方，静的解析では人手による解析コストが高いことから解析部分を絞る必要がある．さらに ROP を含む検体の場合，動的解析では環境のバリエーションが増加し，静的解析ではメモリ状態の解析等が加わることにより，両解析方式とも解析が困難となる．本提案手法により，対象検体に ROP が含まれているか否か，ROP が含まれている場合そのオフセット値や，ROPgadget のアドレス等の情報を得ることができる．これらの情報を参考にし，解析者は，さらに詳細な解析に進むことができる．2 つ目は，企業内の CSIRT 等の組織で，対象検体にゼロデイ攻撃の可能性がないかチェックをしたい場合である．6.6 節で示すように，同一の ROP コードが異なる脆弱性を攻撃するために用いられる傾向があるため，未知の脆弱性を突くゼロデイの悪性文書ファイルの早期検出に効果が見込める．

本提案手法の対象とする環境は，OS は 32 ビット Windows，文書ファイルは，Microsoft Word（拡張子 doc），Microsoft Excel（拡張子 xls），Microsoft PowerPoint（拡張子 ppt），Rich Text Format（拡張子 rtf）である．

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

悪性文書ファイルとは，ファイル内に文書データ，攻撃コード，マルウェアを含む文書ファイルである．悪性文書ファイルは閲覧ソフトウェアの脆弱性を突いて感染するが，exe 形式のマルウェアとは異なり実行形式ではないためユーザは不安を感じずに開いてしまうことが多い [3]．**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，主に，文書ファイル内に埋め込まれた実行形式のマルウェアを取り出して実行する．

図 1 基本的な攻撃の流れ
Fig. 1 The flow of basic attack.

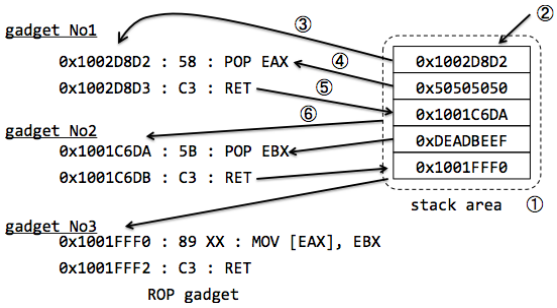

図 2 ROP コードの動作原理
Fig. 2 How the ROP code works.

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キック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な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）[4] 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．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（①）．スタックの最上位の値にリターンするように調整しておき（②），最初に gadget No1 が実行される（③）．レジスタ EAX には意図した値（0x50505050）が格納され（④），ret 命令でスタックを参照し（⑤），次の gadget No2 にリターンする形で gadget

図 3 高度な攻撃の流れ
Fig. 3 The flow of advanced attack.

No2 が実行される（⑥）．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 に関連する技法として JOP（Jump Oriented Programming）[5] が提案されている．ret の代わりに jmp を用いるものだが，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）[6] 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．なお ROP，JOP，SOP は Code-Reuse 攻撃とも呼ばれる．

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．ASLR は，攻撃者が Return-to-libc や ROP の手法により API 関数やスタック・ヒープの固定な既知アドレスを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 32 bitOS 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 ASLR によるランダム化が行われないものが存在し，これを悪用した手法が最近の攻撃の主流となっている．さらに，ランダム化が行われている 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 コードを組み立てる手法も登場している [13]．ここで ASLR が適用されていない DLL を ASLR 非対応 DLL，ASLR が適用されておりロード時にアドレスがランダム化される DLL を ASLR 対応 DLL と呼ぶことにする．

2.6 防御メカニズムを回避する攻撃コード

DEP（2.2 節）や ASLR（2.4 節）といった防御メカニズムを回避するために，攻撃者は ROP 技法（2.3 節）を用いる．また IDS やアンチウイルス等の検出メカニズムを回避するために，攻撃者はシェルコードを暗号化する．この ROP コードや暗号化シェルコードが埋め込まれた悪性文書ファイルの動作順序を**図 3** に示す．図 1 と比べると ROP コード実行と復号コード実行が増えている．実行順序に着目すると，必ず ROP コードが実行された後で，暗号化されたシェルコードを復号するための復号コードが動作する．ROP コードは DEP を回避して復号コードに実行権限を与える目的に使われるためである．すなわち ROP コード部は復号コード以降に含めることができず，シェルコードの暗号化の対象にならない．本提案手法ではこの点に着目している．

3. 関連研究

3.1 悪性文書ファイル検出

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 [7]．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 o-checker 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 [8]．両手法ともにアンチウイルスソフトよりも高い検出性能を持っていると報告されている．鍵の探索機能を持つツールとして OfficeMalScanner がある [9]．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．いずれの方式も ROP コード検出の観点はなく，我々の提案手法とは異なる．

3.2 ROP コードの動的解析

次に ROP コードの検出分野での先行研究を示す．いずれもホスト側に実装し動的に検出する手法であり，静的に ROP コードを検出する先行研究はない．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よい．これは従来から提案されている方式で，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 の管理のオーバヘッドが大きくなり性能に影響が出る [10]．

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）を用いて ROP コードを検出を実現している [11]．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 コードを検出する．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 コードかを判定する．LBR の制約から履歴は 16 個に限られるが，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．さらに Cheng らは，ROPecker は，LBR に加え，さらにスライディングウインドウやコンディションフィルタリング等を用い kBouncer を改良している [16]．

一方で，動的解析である kBouncer や ROPecker の限界も示されている．Carlini らは，これらの方式が，ret 命令の戻りアドレスの 1 つ手前の命令が call 命令かどうかをチェックしている点に着目し，そのような ROP チェックを回避できる gadget をみつける等，代表的な 3 つの手法を提案し，動的解析検出を回避できることを示した [17]．また Goktas らは，kBouncer や ROPecker が ROP 判定閾値である個々の ROP gadget のサイズや ROP gadget のチェイン数を経験則により固定的に決めていることに着目し，その閾値を超えない ROP コードを構成することにより検出の回避を成功させている [18]．

3.3 ROP コードの静的解析

Tanaka らは，攻撃コードとして ROP 技法が用いられる際の，ROP コードの目的や動作順序に着目し，ROP 技法で用いられるアドレスを特徴文字列として蓄積しておき，パターンマッチによる検出を行うことを提案している．攻撃ツール Metasploit [*1] に限定した評価ではあるが約 84%の ROP 攻撃を検出可能であった [20]．

1 章後半で述べたが，検体詳細解析の現場での，MS-Office 文書ファイルの動的解析では安全面の考慮に加えパッチレベル等環境を変化させ攻撃が発動する環境を探す必要がある．さらに ROP を含む検体の場合，動的解析では環境のバリエーションが増加し，さらに解析が困難となる．文献 [10], [11], [16] の ROP 防御機構環境を用いた場合でも解析コストは同様に高い．これは ROP がメモリ状態に強く依存することから，正確に ROP が動作するスタックやヒープの状態を再現する必要があるためである．このような正確なメモリ状態構築には，解析対象の文書ファイルだけではできないケースも多く，他の適切な DLL 等の読み込みが必要になる等解析を困難なものとする．

4. 提案方式の着眼点

本提案では，ホスト側の防御機構を突破する ROP コードを静的に特定することで，悪性文書か否かの判定を可能とする手法を提案する．本提案では図 3 に示すように ROP コード部はシェルコードの暗号化ルーチンの外部に置かれることに着目する．ROP コードは復号コード自体に実行権限の付与する役割を持つため 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．このため ROP コード部はシェルコードのようにコード実行によって暗号化を解く必要がなく静的な解析が可能である．一般に静的解析は攻撃コードを実行しないため安全に解析できる利点があり，文献 [7], [8], [9] の手法が提案されているが，ROP コードに着目した静的解析の提案例はない．

[*1] Metasploit. http://www.metasploit.com

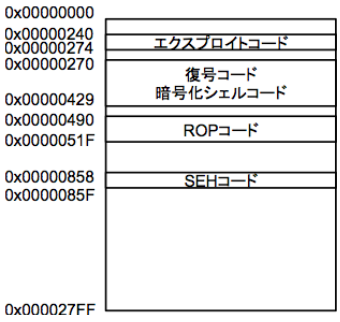

図 4 悪性文書ファイルの例
Fig. 4 An example of malicious documents file.

4.1 各コード部の考察

2.6 節および図 3 で示した各攻撃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．

これは，6.2 節に示すような実際の悪性文書ファイルを調査して得られた代表例である．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ごとに考察を行う．

- 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 コード部

DEP 回避を目的とする．後続する復号コード部の外に配置され，復号コードに実行権を付与する．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．4.2 節に示す方法で一般に流通する攻撃コードを調査したところサイズは平均 100 Byte 程度であった．

- SEH コード部

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あり安定して後続のコードを実行されること目的とする．SEH コードは ROP コード部と同様に復号コードの外側に配置され，シェルコードの暗号化の対象にはならない．調査結果，コードサイズは平均 10 Byte 程度であった．

- 復号コード部

暗号化シェルコードの復号を目的とする．XOR 等の論理演算を用いる単純なコードの場合は，数十 Byte 程度であった．ポリモーフィックコードを生成するエンコーダが用いられた場合，さらにサイズが大きくなった．

- 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 [12]．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ために 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

表 1 実行権限付与関数例
Table 1 Examples of typical Windows functions to grant execute permission.

関数名	
VirtualProtect	SetProcessDEPPolicy
VirtualAlloc	WriteProcessMemory
HeapCreate	NtSetInformationProcess

表 2 VirtualProtect 関数の引数
Table 2 Arguments of VirtualProtect function.

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)

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．

上記の考察より，脆弱性に依存せず，暗号化や難読化にも影響されない部分は，ROP コード部，SEH コード部，復号コード部である．SEH コード部や論理演算ベースの復号コード部はサイズが小さく特徴をとらえるのは容易ではない．エンコーダベースの復号コード部はそのサイズから特徴をとらえられる可能性はあるが，本提案では ROP コード部を対象とした．

4.2 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つな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，次に示す方法で調査した結果，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．調査方法は，最初に，Metasploit の攻撃コードから，Windows プラットフォーム用のすべての攻撃コードのうち，ROP が用いられるものを抽出し（CVE 番号ユニークで約 70 個あった）目視による解析を行った．ここで Windows プラットフォームすべてを対象としたので MicrosoftOffice 以外の脆弱性も含まれる．次にこの攻撃コードの CVE 番号から exploit-db [*2] の検索機能を使い，exploit-db に登録のある Metasploit 以外の攻撃コードを入手し同様に解析を行った．また，インターネットから入手できる悪性文書検体や独自に入手した悪性文書検体で ROP が含まれるものも解析対象とした．この調査から，ROP コードの目的は，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るという事実を発見した．

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．またその 1 つである VirtualProtect 関数の引数について**表 2** に示す．

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

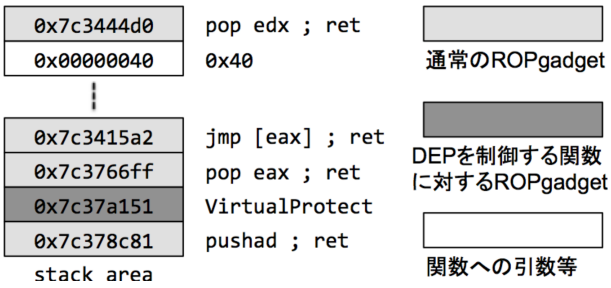

図 5 ROP コードが積まれたスタック
Fig. 5 An example of stack area after ROP code is loaded.

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．

図 5 で DEP を制御する表 1 に示す関数のアドレス（図では VirtualProtect 関数）に関する ROPgadget 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 を通常の ROPgadget 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 およ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 が回避されると復号コードの実行等の処理が開始する．

攻撃者は安定して攻撃を成功させるためにアドレスが固定値として存在する ROPgadget 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，固定値として用いることが困難となるが 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，ROPgadget のアドレス数は限定的であり，3.3 節で示した Tanaka らの従来方式 [20] は，この点を利用して検出を行う．

5. 提案方式

5.1 非シグネチャ依存方式

3.3 節で示した従来方式 [20] は，攻撃者が用いる既知の ROPgadget 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，未知のアドレスは検出できない．ただ攻撃ツール Metasploit の攻撃コードを利用した攻撃も多数観測されており脅威であることから [21]，Metasploit 攻撃コード検出に効果が示されている従来方式 [20] 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 DLL のアドレスを用いる攻撃手法，または，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 [13] に対してはあらかじめシグネチャを作れないため検出できない．特に文献 [13] の手法は ASLR 非対応 DLL を必要としないため攻撃者の立場からすると有効である．ASLR 非対応 DLL は悪用されることが知られるようになると，ASLR 対応した修正プログラムがリリースされるためである．このような新たな攻撃手法に対抗するため，提案方式では，従来方式のように特定の DLL 等の固定アドレスのシグネチャに依

```
6 6 4 C 3 6 7 C 6 6 4 C 3 6 7 C D C 4 E 3 4 7 C 1 0 0 2 0 0 0 0
0 5 1 E 3 5 7 C E 3 3 6 3 5 7 C F F F F F F F 5 5 5 2 3 4 7 C
8 E 2 1 3 5 7 C 3 7 5 9 3 4 7 C 0 0 0 0 0 0 4 0 B 1 1 E 3 5 7 C
B 9 C 5 3 6 7 C 6 7 1 E 3 9 7 C 5 8 2 E 3 4 7 C 0 2 D 2 3 4 7 C
F 4 F 8 3 4 7 C A 2 1 5 3 4 7 C D C 4 E 3 4 7 C 5 1 A 1 3 7 7 C
8 1 8 C 3 7 7 C 3 0 5 C 3 4 7 C
```

図 6 ROP コードの例
Fig. 6 An example of ROP code.

存しない方法で ROP コードの検出を目指す．

提案方式を説明するために，ASLR 非対応 DLL から作られた ROP コードを**図 6** に示す．これは Metasploit から取得した JRE1.6 アプリケーションである msvcr71.dll から作られた実際の ROP コードである．msvcr71.dll は ASLR 非対応 DLL のためベースアドレスはつねに固定で，メモリ空間 0x7C340000 から 0x7C396000 にロードされる．攻撃者はこの空間内から ROPgadget を探してそれらを組み合わせて ROP コードを作成する．このため，作られる ROPgadget はすべて上位バイトが 0x7C3 となる．この ROPgadget が連結された固まりである ROP コードは図 6 のようになる．したがって図のグレーで示すように上位バイトが共通して 4 バイト周期で現れるという特徴がある．図の各バイト列はリトルエンディアンで配置されることに注意する．未知の ASLR 非対応 DLL や ASLR 対応 DLL から作られた ROP コードの場合，ベースアドレスは推測できず，対応したシグネチャは作成できないが，特定のメモリ空間に DLL はロードされるため，図のグレーで示すような 4 バイト周期で同一文字列が現れるという特徴は共通する．つまり提案手法の要点は ROPgadget は特定の DLL のアドレス空間から作られ ASLR 対応か非対応かにかかわらず，上位バイトが等しい ROPgadget が固まりとなって ROP コードとして文書ファイル内に含まれることに着目し検出を行う．この周期的に現れる特徴をベースに経験則から次節以降で示す複数の条件や閾値を決定し，シグネチャに依存しない方式で ROP コードを判定する．

5.2 提案方式：全体フローチャート

図 7 に提案方式の全体フローチャートを示す．判定対象のファイルは文書ファイルである doc 形式，xls 形式，ppt 形式，rtf 形式である．バイナリ形式でファイルオープンする．rtf 形式の場合，攻撃コードは ASCII 文字エンコードされているためデコードを行う．ここで 1 ワードを 4 バイトと定義する．図 7 の (※ 1) で 1 ワードを読み込み，後述する PRE check 部で判定を行い，No であれば，offset に 1 バイト加算し，再度，図 7 (※ 1) で 1 ワードを取り出し同様の判定を行う．PRE check 部で Yes となった場合，図 7 (※ 2) で offset から 25 ワード取り出し，BASIC check 部で判定を行う．No の場合は offset に 1 バイト加算し同

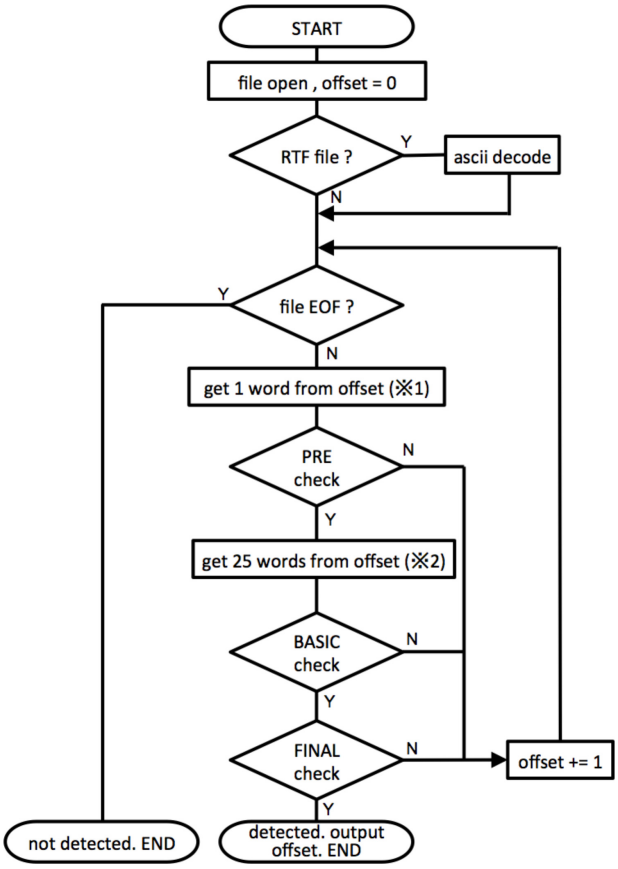

図 7 全体フローチャート
Fig. 7 The entire flowchart.

表 3 PRE check 条件
Table 3 PRE check conditions.

条件 No	条件	例
P1	値が 0x7FFFFFFF より大きい	0x953B235F
P2	値が 0x01000000 より小さい	0x0001F325
P3	値が 0x10000 で割り切れる	0x1B420000

様に頭から処理を行う．Yes の場合，次の FINAL check 部での判定を行い，Yes であれば ROP コードが見つかったとして ROP コード部のオフセット値を出力して終了，No であれば，同様に頭から処理を行う．ここで 25 ワードの値は，実際の攻撃コードを観察し得た知見から決定した．

5.3 提案方式：PRE check 部

PRE check 部では先頭 1 ワードが ROP コードであるかどうか判定を行う．**表 3** で示す条件のいずれかに該当する場合 ROP でないと判断し図 7 で No に進む．表 3 の条件 P1 は，悪用される DLL はユーザ空間にロードされる経験則から，ROPgadget として利用されるメモリ空間は 0x00000000 から 0x7FFFFFFF の範囲となり，この範囲外であれば ROP コードと判断しない．条件 P2 は，本方式の制約事項となるが，上位 1 バイトが 0x00 で始まるメモリ空間の ROP コードの判定は，誤検出を誘発することから行わないこととした．つまり 0x00000000 から 0x00FFFFFF

表 4 BASIC check 条件
Table 4 BASIC check conditions.

条件 No	条件	閾値	例
B1	1バイト毎に同一の値が閾値以上出現	15	0x0B340B7A 0x0B880B55
B2	各ワードの上位 2 バイト目が 0x00 となるワードが閾値以上出現	12	0x3D001283 0x2200DEAD
B3	各ワードの下位 1 バイト目が同一となるワードが閾値以上出現	12	0x4D12836D 0x458E546D
B4	各ワードの下位 1 バイト目が単調増加するワードが閾値以上出現	6	0x340B2E01 0x9B6EB702 0x831E6603
B5	各ワードの下位 1 バイト目が 0x00 となるワードが閾値以上出現	6	0x3D321200 0x22DE4500

のメモリ空間で ROP コードを組み立てられた場合，本提案手法では検出できない．これは MS-Office 文書内に 0x00 で始まる ROP コードに類似したコードが存在するためである．MS-Office 文章は CFB 形式 [22] であり，その仕様から一部の 0x00 を含む固定長フィールドの繰返しが影響すると推測される．この類似部分を判定する他の条件が見つけられるのが理想ではあるが，この制約空間（0x00000000 から 0x00FFFFFF）は ROP 技法が用いられるユーザ空間全体（0x00000000 から 0x7FFFFFFF）のなかの 1/128 の割合であるため，今回は制約事項としても本提案手法の利用価値への影響は少ないと判断し本研究の対象外とした．条件 P3 は，下位 2 バイトがともにゼロとなる場合で，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皆無と考え可能性なしとする．

5.4 提案方式：BASIC check 部

BASIC check 部では，攻撃コードを観察した知見から，該当 25 ワードが，ROP コードの可能性があるかの判断を行う．**表 4** に示す条件のいずれかに該当する場合 ROP の可能性なしとして図 7 で No に進む．いずれも ROP コードを組み立てるうえでの障害となる条件であり除外して問題ないと考えられる．

5.5 提案方式：FINAL check 部

FINAL check 部では，該当 25 ワードが ROP コードと断定できるかどうかの最終判断を行う．**図 8** にフローチャートを示す．先頭 1 ワードと後続する各ワードとの差分を diff 配列に格納する．次に図 8 (F1) で diff 配列で値がゼロのものや同じ値のものは ROP の組み立てに非現実であるため，ROP コードではないと判断し，diff 配列から除外する．アドレスの差分が同じ ROPgadget を複数用いて ROP コードを組み立てることは，CTF [*3] 等の競技において問題作成者が意図的に制約として課すケースがある．しかし，ROPgadget が得られるメモリ空間が広い必要があり，現実世界では利用可能なメモリ空間は狭いため，実攻撃コードとして経験上目にすることがなく，ROP コードの可能性は低いとして除外している．次に図 8 (F2) で diff の値が値 0x100000 以上のもの，つまり個々の ROPgadget のアドレスが離れすぎているものは ROP として非現実であるとして除外する．この閾値とした値については，ROP に用いられる代表的な DLL を調査した結果を**表 5** に示し，この値を参考に決定した．図 8 (F3) では，各 ROPgadget のアドレスが逆にあまりにも近い場合は ROP を組み立てるのが困難になり非現実であるので，閾値 10 個より大きい場合 ROP ではないと判断する．具体例として，F3 条件に該当する ROP コードの例は，0x12345611, 0x12345633, 0x123456ab, 等で下位 1 バイトのみ変化する異なる ROPgadget が 10 個以上用いられる ROP コードである．5.1 節で説明したように，攻撃者は特定の固定アドレスのメモリ空間（図 6 の例で，msvcr71.dll が読み込まれ

*3 CTF（Capture the flag）：コンピュータセキュリティ技術を競う競技

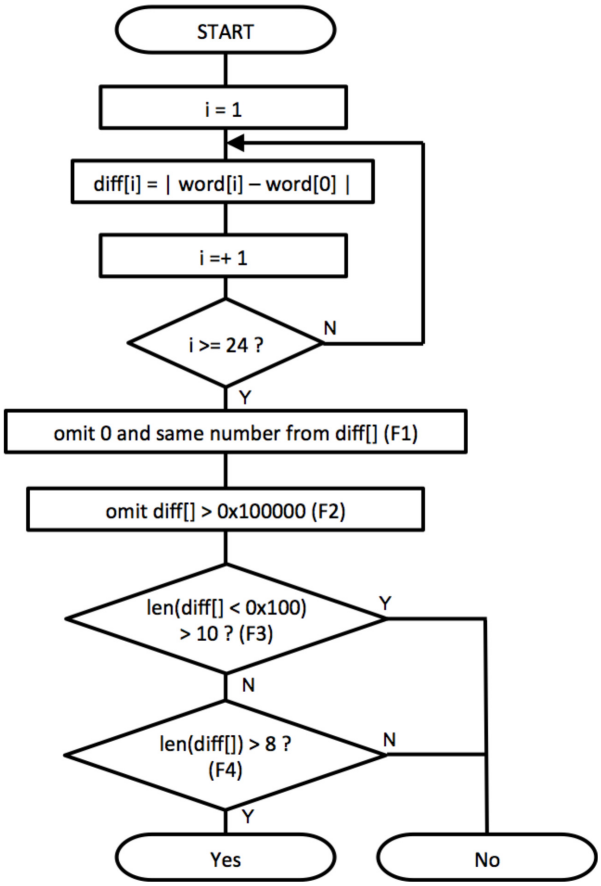

図 8 FINAL check 部フローチャート
Fig. 8 The flowchart of FINAL check.

表 5 ROP に用いられる代表的な DLL 例
Table 5 Typical DLL examples used in ROP.

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.dll	0x77c10000	0x77c68000	0x58000

る 0x7C340000 から 0x7C3960000）をできる限り活用して ROPgadget を見つける．このとき，たとえば 0x7C340000 から 0x7C3400FF の中からのみ ROPgadget を見つけ DEP を解除するコードを組み立てるのはほぼ不可能であると経験則から判断し，この条件を入れた．最後に図 8 (F4) で，残った diff 配列の要素数が閾値 8 個より大きい場合 ROP コードと判定する．つまり，F4 条件は最終的に，これまでの処理で ROPgadget と考えることができるワードが閾値個以上のこった場合，該当コード部分が ROP コードと最終判定している．これらの閾値も経験則によって決定した．

6. 実験

6.1 ツール試作

提案方式についてツールを試作し実験を行った．ツールの仕様は以下のとおりである．

- OS：Mac OS X v10.9 Mavericks
- 言語：Python 2.7.5
- 対応ファイル形式：doc, xls, ppt, rtf

6.2 実験検体

手動解析で ROP コードが埋め込まれていることが確認できた検体を準備した．内訳は，**表 6** に示すインターネット[*4]で入手できた in the wild 検体含む約 150 個の悪性文書ファイルから ROP を含むもの 6 検体，独自ルートで確認した悪性文書ファイルから 8 検体である．悪性文書ファイル自体の流通が少なく，また，ROP コードが埋め込まれている検体は比較的新しい検体に限られることから，これ以上の検体母数を増やすことは困難であった．また一部のファイルではファイル名を MD5 ハッシュ値の先頭 5 文字のみの標記としている．これらの検体はあらかじめ解析を行い CVE 番号が判明したものは標記した．これは異なる CVE 番号でも類似した ROP コードが用いられる場合があるのではないかという仮説を検証するために行った．

表 6 ROP 検体
Table 6 ROP samples.

検体 No	ファイル名	形式	CVE	入手元	in the wild
1	msf11_006.doc	doc	2010-3970	Metasploit	
2	44.doc	doc	2010-3970	contagio	✔
3	msf12_027.doc	rtf	2012-0158	Metasploit	
4	cve_.doc	rtf	2010-3333	exploit-db	✔
5	msf14_017.doc	rtf	2014-1761	Metasploit	
6	e378_.bin	rtf	2014-1761	malwr	✔
7	d309e_	doc	2012-0158	独自	✔
8	c2b68_	rtf	2012-0158	独自	✔
9	cf548_	doc	2014-1761	独自	✔
10	006fc_	rtf	2012-0158	独自	✔
11	47a92_	doc	不明	独自	✔
12	6653a_	rtf	2012-0158	独自	✔
13	b61ee_	doc	不明	独自	✔
14	eda67_	rtf	2012-0158	独自	✔

[*4] Metasploit. http://www.metasploit.com
contagio malware dump. http://contagiodump.blogspot.jp/
The Exploit Database. http://www.exploit-db.com/
malwr. https://malwr.com/

6.3 False Positive 検査

正常なファイルを誤判定しないことの確認として，正常なファイルを，doc 形式：15 ファイル，rtf 形式：15 ファイル，xls 形式：10 ファイル，ppt 形式：10 ファイルの合計 50 ファイルを主にインターネットからダウンロードし正常検体として用意した．試作ツールで検査を行い ROP コード判定されないことを確認した．

6.4 実験結果 1

各機能部の条件と閾値が適切に機能していることを確かめるために，ROP 検体および正常検体の一部（6.3 節の検体から ROP 検体と同数をランダム抽出）に対して，各条件に一致した回数と配列要素数を**表 7**，**表 8**，**表 9**，**表 10**，**表 11**, **表 12** にまとめた．PRE check および BASIC check について，Yes に進んだ回数と No に進んだ内訳回数を示

表 7 ROP 検体 PRE check 結果
Table 7 PRE check results of ROP samples.

検体 No	Yes	No		
		P1	P2	P3
1	272	758	132	19
2	289	741	132	19
3	124	1465	741	60
4	852	762	57	8
5	1975	0	0	0
6	9551	8739	9437	1202
7	203867	148828	42961	5774
8	1301	5642	4387	180
9	9446	8685	9407	1202
10	124	1465	741	60
11	1817	2013	13649	1120
12	203867	148828	42961	5774
13	1828	2015	13661	1123
14	205512	147222	42878	5737

表 8 ROP 検体 BASIC check 結果
Table 8 BASIC check results of ROP samples.

検体 No	Yes	No				
		B1	B2	B3	B4	B5
1	256	16	0	0	0	0
2	273	16	0	0	0	0
3	1	122	0	1	0	0
4	586	264	0	2	0	0
5	1975	0	0	0	0	0
6	6381	3112	15	13	22	9160
7	161323	42143	260	20	121	53
8	688	613	0	0	0	727
9	6212	3176	15	30	13	22
10	1	122	0	1	0	0
11	231	1542	18	26	0	0
12	161323	42143	260	20	121	53
13	232	1551	18	27	0	0
14	161797	43293	260	20	142	87

表 9 ROP 検体 FINAL check 結果
Table 9 FINAL check results of ROP samples.

検体 No	F1	F2	F3	F4
1	259	2839	1	9
2	276	3245	1	9
3	4	1	1	12
4	907	7636	2	12
5	18443	32295	4	5
6	9160	90904	2	5
7	189093	1950658	4	12
8	727	8884	1	10
9	9042	88466	2	6
10	4	2	1	12
11	337	2414	2	12
12	189093	1950658	4	12
13	339	2422	2	11
14	190200	1973647	6	12

表 10 正常検体 PRE check 結果
Table 10 PRE check results of normal samples.

検体種別	Yes	No		
		P1	P2	P3
doc	52443	6817	55925	4010
doc	12407	11666	3010	146
doc	17431	2961	27212	2983
doc	40579	20249	68146	8141
rtf	45085	47563	158355	8356
rtf	18522	9054	426	2
rtf	27241	193661	101043	7955
rtf	20988	136353	7643	711
xls	17643	3893	26311	3252
xls	1368118	116843	574812	29086
xls	15171	8918	20728	1674
ppt	56616	89609	46839	7027
ppt	277766	276022	12874	1114
ppt	676143	642041	79341	9862

表 11 正常検体 BASIC check 結果
Table 11 BASIC check results of normal samples.

検体種別	Yes	No				
		B1	B2	B3	B4	B5
doc	5587	44454	1721	674	7	75
doc	11472	935	0	0	0	0
doc	40	17323	65	5	0	0
doc	22384	14120	2811	984	280	758
rtf	18099	26984	0	0	2	1
rtf	17836	685	0	0	1	0
rtf	14851	11638	143	582	27	6
rtf	12378	8593	0	17	0	0
xls	4095	13493	35	20	0	39
xls	76217	1164496	105229	22162	14	48
xls	5607	9564	0	0	0	0
ppt	36715	19324	454	123	0	3
ppt	269703	7763	81	200	19	33
ppt	637680	36224	1845	363	31	55

した．FINAL check について，F1，F2 に各条件で除外された要素数合計値を示した．全検体で F3 閾値である 10 を超えず Yes に進まなかったため，F3 に配列要素数の最大値（図 8 に示す 0x100 より小さい diff 配列の要素数の最大値）を示した．F4 に条件に一致した際の値（図 8 に示す diff 配列の要素数）を示した．表 12 については F4 閾値を超えなかったため，ファイルを終端まで検査して，配列要素数の最大値を示した．

表 12 正常検体 FINAL check 結果
Table 12 FINAL check results of normal samples.

検体種別	F1	F2	F3	F4
doc	18434	86826	2	5
doc	11481	135859	1	2
doc	91	492	0	0
doc	45365	305443	3	6
rtf	18990	183904	5	5
rtf	42716	259569	3	3
rtf	21125	253134	3	6
rtf	18032	132823	2	4
xls	11782	74713	2	5
xls	184428	484180	3	5
xls	12443	135076	1	3
ppt	237126	376870	5	5
ppt	292122	3161531	2	4
ppt	662944	7498679	4	5

条件 F3 以外は，経験則から決定した条件や閾値にヒットしたことから適切に機能したと判断できる．しかし条件 F3 は全検体で閾値を超えず，今回のサンプルでは不要な条件となり期待した効果を発揮しなかった．

6.5 実験結果 2

提案方式を実装した試作ツールの結果のほかに，関連研究である 3.1 節の o-checker，OfficeMalScanner，およびアンチウイルスソフトである ClamAV での検出結果を**表 14** に示した．

試作ツールは No5，6，9 を除く検体について，ROP コード判定を行い，ROP コードのオフセット値を出力した．また 5.5 節の F4 条件で残ったワードを参照することで ROPgadget のアドレス情報を得ることができた．例として，検体 No3 および No4 でから得られた ROPgadget のアドレス情報を 6.6 節の**表 15** に示す．オフセット値が正しいかは各検体をバイナリエディタで閲覧し確認した．次に ROPgadget のアドレス情報が正しいかの確認は動的解析が必要となる．実際にはデバッガを用い，各検体を動作させ，得たアドレス情報が，メモリ領域上で 2.3 節の図 2 で示した ROP コードを構成する意味のある値であり確かに ROPgadget であることを確認した．さらにこれらのコードを観察したところ，4.2 節の図 5 で示した特定の DLL アドレス空間から構成される DEP を解除する目的とした ROP コードであることが分かった．このことからも特定 DLL アドレス空間に着目した 5 章で示す提案方式の有効性が確認できた．

一方で，検体 No5，6，9 は試作ツールで判定できなかった．これらの検体は CVE-2014-1761 で脆弱性を発動させるためにエンコードしておく必要があり図 6 に示すような

表 13 エンコード詳細
Table 13 Encoding details.

工程	データ	説明
1	0x12345678	オリジナルデータ
2	0x1234 0x5678	上位・下位バイト分離
3	0x5678 0x1234	上位・下位バイト反転
4	22136 4660	10 進数変換
5	-18040 -564	0x1000 から工程 4 の値を引く
6	18040 564	絶対値取得
7	?¥u-18040 ?¥u-564	“ ?¥u-“ を先頭に付加

表 14 実験結果
Table 14 Experimental results.

検体 No	試作ツール	o-checker	OfficeMal Scanner	ClamAV
1	✔	NA	NA	NA
2	✔	NA	NA	NA
3	✔	NA	NA	✔
4	✔	✔	✔	✔
5	NA	NA	NA	NA
6	NA	✔	✔	✔
7	✔	NA	NA	NA
8	✔	✔	✔	NA
9	NA	✔	✔	✔
10	✔	NA	NA	✔
11	✔	✔	NA	NA
12	✔	NA	NA	NA
13	✔	✔	NA	NA
14	✔	NA	NA	NA

表 15 使われた ROPgadget の例
Table 15 Examples of ROP gadgets in samples.

検体 No3	検体 No4
0x3f2233cc	0x3f2233cc
0x3f2b745e	0x3f2b745e
0x3f2cb9e0	0x3f2cb9e0
0x3f2d59df	0x3f2d59df
0x3f2f18cc	0x3f2f18cc
0x3f389ca5	0x3f389ca5
0x3f39795e	0x3f39795e
0x3f39afcf	0x3f39afcf
0x3f39cb44	0x3f39cb44
0x3f39cb46	0x3f39cb46
0x3f3b3dc0	0x3f3b3dc0
0x3f3b3dc4	0x3f3b3dc4
0x3f10115c	0x3f398267
0x3f2cb9e1	0x3f3a16de
計 14 個	計 14 個

バイトコードとは形式が異なるため検出できなかった [15]．エンコード工程の詳細を**表 13** に示す．工程 1 のデータがオリジナルデータで工程 7 がエンコード後のデータである．エンコード対象は，攻撃コードすべて（エクスプロイトコード，ROP コード，復号コード，シェルコード）で，エンコードなしでは，攻撃者が意図する脆弱性が発動しない．このようにエンコードとは暗号化や難読化と異なり一定のルールに基づいて型変換することを示す．なお，このエンコード下でも ROP コード部はシェルコードの暗号化ロジックの外に配置されるというルールは成り立っている．

表 16 ROPgadget のユニーク数
Table 16 Unique numbers of ROP gadgets in samples.

検体 No	1	2	3	4	5	6	7
gadget 数	10	10	14	14	NA	NA	13

検体 No	8	9	10	11	12	13	14
gadget 数	11	NA	14	13	13	12	13

o-checker は，No1，2，3，5，7，10，12，14 が検出できなかった．この理由として，ファイル構造に問題がない形で各種攻撃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．一方，試作ツールで検出できなかった No6，9 を o-checker はファイル構造の特徴を使って検出することができた．

OfficeMalScanner は，No4，6，8，9 についてシェルコード検出機能により検出できた．検出できなかった No1 の検体は，metasploit のシェルコードエンコードオプションで bloxor を指定して作成したものである．これを外したところ OfficeMalScanner 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と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とらえるのに有効な手法である．ただし最近のシェルコードは暗号化がされていることからこの手法での検出には限界がある．

最後に ClamAV については，No6，9 の比較的新しい脆弱性を検出できたにもかかわらず，他の古い脆弱性の検出はできないという結果になった．パターンマッチング方式での検出は難しいことが推測される．

6.6 ROP コードの解析と汎用性考察

提案手法の有効性をさらに考察するために，試作ツールで ROP 検出可能であった検体の ROP コードの解析を行った．表 15 に検体 No3 および No4 で用いられた ROPgadget を示す．それぞれユニークな ROPgadget は 14 個使われており，両者で共通しているものをグレーで示す．**表 16** に各検体で用いられた ROPgadget のユニークな個数を示した．**表 17** に検体どうしで共通の ROPgadget が使われた割合を示す．分母の数字は比較 2 検体合計のユニーク ROPgadget 数であり，分子は比較 2 検体で共通した ROPgadget 数である．たとえば，表 15 の検体 No3 と No4 で分母は 16，分子は 12 となり，表 17 には 12/16 と標記される．表 17 より No1 と No2，No3 と No10，No7 と No12 と No14 は用いられた ROPgadget が完全に一致していることが分かる．また No3 と No4，No4 と No10，No11 と No13 は，それぞれ 12/16（75%），12/16（75%），11/14（約 79%）と高

表 17 同一 ROPgadgets が使われた割合
Table 17 Percentages of the same ROP gadgets in samples.

No	1	2	3	4	5	6	7
1	-	10/10	0/24	0/24	NA	NA	0/23
2	-	-	0/24	0/24	NA	NA	0/23
3	-	-	-	12/16	NA	NA	0/27
4	-	-	-	-	NA	NA	0/27
5	-	-	-	-	-	NA	NA
6	-	-	-	-	-	-	NA
7	-	-	-	-	-	-	-

No	8	9	10	11	12	13	14
1	0/21	NA	0/24	0/23	0/23	0/22	0/23
2	0/21	NA	0/24	0/23	0/23	0/22	0/23
3	0/25	NA	14/14	0/27	0/27	0/26	0/27
4	0/25	NA	12/16	0/27	0/27	0/26	0/27
5	NA	NA	NA	NA	NA	NA	NA
6	NA	NA	NA	NA	NA	NA	NA
7	0/24	NA	0/27	2/24	13/13	1/24	13/13

No	8	9	10	11	12	13	14
8	-	NA	0/25	1/23	0/24	1/22	0/24
9	-	-	NA	NA	NA	NA	NA
10	-	-	-	0/27	0/27	0/26	0/27
11	-	-	-	-	2/24	11/14	2/24
12	-	-	-	-	-	1/24	13/13
13	-	-	-	-	-		1/24
14	-	-	-	-	-	-	-

表 18 ROPgadget による分類
Table 18 Classifications by ROP gadgets.

検体 No	1	2	3	4	5	6	7
タイプ	I	I	II	II	NA	NA	III

検体 No	8	9	10	11	12	13	14
タイプ	IV	NA	II	V	III	V	III

い割合で同一の ROPgadget が用いられていることが分かる．ここで，表 6 より No3 は CVE-2012-0158 脆弱性を，No4 は CVE-2010-3333 脆弱性を利用するもので，用いられる脆弱性は異なるが，用いられる ROPgadget は高い割合で一致していることが分かる．これは，冒頭で述べたように，ゼロデイ脆弱性で，その脆弱性に対するシグネチャが存在せず対処ができない場合でも，異なる脆弱性でも共通して使われる可能性が高い ROP コードを検出することで，ゼロデイ対策になる可能性を示している．

さらに，**表 18** に，表 17 での ROPgadget の類似度を参考にして，各検体の ROP コードを分類した．結果，5 つのタイプに分類できた．これは，6.2 節で示したように ROP 検体の多数収集は困難である状況下で限られたサンプル数ではあったが，本提案手法は，異なる 5 つのタイプの ROP コードを検出できたことを示しており，有効性が確認できた．

7. 議論

本提案方式の特徴は以下である．

- シグネチャに依存せず ROP コードを検出

本方式により，高度化する攻撃手法に対抗するためシグネチャに依存せず ROP コードを検出し，ROP コードのオフセット値や ROPgadget のアドレス情報を得ることができる．ROP コード部の目的は DEP の解除でありシェルコードの暗号化ロジックの外側に配置される点に着目し検出する．ROP コードは新しい脆弱性において多用される点 [19]，および，同一 ROP コードが複数の脆弱性を攻撃するために用いられる傾向がある点から，未知の脆弱性を突くゼロデイ検体検出に効果が見込める．

- ツールによる静的解析検出

本方式は，他の関連研究では行われていない静的解析による ROP コード検出を行う．一般に，文書ファイル型検体の解析には，動的解析に加え詳細な機能を知るための人手による静的解析が行われる．本方式はツールとして動作し，解析者に有益な出力情報を提供する．

- 既存方式との併用

本提案方式の検出方法は既存方式と独立しているため組み合わせて用いることができ，攻撃が高度化するなか多層防御の観点でも貢献できる．

一方，本提案方式の課題として以下がある．

- エンコードへの対応

CVE-2014-1761（検体 No5，6，9）では ROP コードがエンコードされていたため，本提案方式で検出ができなかった．ただ脆弱性を発動させるためのエンコードは難読化とは異なり，一定の制約に基づいて型変換されるため，検出できる可能性はあるが，アルゴリズムは図 7 とは異なるものを考える必要がある．

- 64 bit 対応

今回は windows32 bit 環境を想定したが，類似アルゴリズムで 64 bit 対応も可能と考えられる．ただ，実際に 64 bit の ROP コードの流通を確認できていない．

- 未知 DLL 名の判定

今回は実装していないが，既知の DLL から作られた ROP の場合は DLL 名を判定することが方式上，可能である．しかしながら未知 DLL の場合は，アドレスは分かっても DLL 名は判定できない．DLL 名を判定するためには，候補 DLL を読み込み動的解析を繰り返す等の解析が必要であり，その自動化も課題である．

- 制約事項の解消

今回上位 1 バイトが 0x00 から始まる ROP コードの判定は誤検出を誘発する可能性があることから行わないこととし，本方式の制約事項とした．これを判定できる条件，たとえば，今回は利用しなかったが DEP 制御関数への引

数等の特徴文字列等を併用する，もしくは，4.1 節で示した復号コード部に現れる特徴文字列を併用する等により制約事項を解消できる可能性がある．

- 閾値を回避する検体

本提案方式では入手可能な ROP 検体の解析結果に基づく経験則から閾値を選定している．意図的に閾値を回避するような検体の場合検出できない．

- 実験検体数の増加

悪性文書ファイルの検体は個人・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うえでの課題である．

8. まとめ

本稿では，対策が非常に困難であるゼロデイ攻撃対策を視野に入れ，標的型攻撃で多用される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．本提案手法では，文書ファイルに埋め込まれた ROP コードの検出により悪性文書ファイルを検出する．ゼロデイを含む攻撃で昨今多用される ROP 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．

謝辞 本研究を進めるにあたりご協力いただいた JPCERT コーディネーションセンターに感謝します．

参考文献

[1] シマンテック：2014 年インターネットセキュリティ脅威レポート第 19 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 2014-11-24）．

[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，情報処理推進機構（オンライン）(2014)，入手先 ⟨http://www.ipa.go.jp/security/vuln/10threats2013.html⟩（参照 2014-11-24）．

[3] ITPro セキュリティ：添付ファイルに工夫の数々，日経コンピュータ（オンライン）(2014)，入手先 ⟨http://itpro.nikkeibp.co.jp/article/COLUMN/20140317/544186/⟩（参照 2014-11-24）．

[4] Shacham, H.: The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86), *Proc. 14th ACM conference on Computer and Communications Security* (*CCS2007*) (2007).

[5] Bletsch, T., Jiang, X., Freeh, W.V. and Liang, Z.: Jump-Oriented Programming: A New Class of Code-Reuse Attack, *Proc. 6th ACM Symposium on Information, Computer and Communications Security* (*ASIACCS2011*) (2011).

[6] Payer, M. and Gross, R.T.: String oriented programming: when ASLR is not enough, *Proc. 2nd ACM SIGPLAN Program Protection and Reverse Engineering Workshop* (*PPREW 2013*) (2013).

[7] 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 (2013).

[8] 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 文書ファイルの検知，情報処理学会研究報告，Vol.2013-IOT-22, No.16 (2013).

[9] Boldewin, F.: Analyzing MSOffice malware with Office-MalScanner (2009), available from ⟨http://www.reconstructer.org/papers/Analyzing%20MSOffice%20malware%20with%20OfficeMalScanner.zip⟩ (accessed 2014-11-24).

[10] Davi, L., Sadeghi, A. and Winandy, M.: ROPdefender: A practical protection tool to protect against return-oriented programming, *Proc. 6th Symposium on Information, Computer and Communications Security* (*ASIACCS 2011*) (2011).

[11] Pappas, V., Polychronakis, M. and Keromytis, D.A.: Transparent ROP Exploit Mitigation Using Indirect Branch Tracing, *Proc. 22nd USENIX Security Conference on Security* (2013).

[12] 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，電子情報通信学会，2014 年暗号と情報セキュリティシンポジウム（SCIS2014），4C2-4 (2014).

[13] FireEye Threat Research Blogs: New Zero-Day Exploit targeting Internet Explorer Versions 9 through 11 Identified in Targeted Attacks, FireEye (online) (2014), available from ⟨https://www.fireeye.com/blog/threat-research/2014/04/new-zero-day-exploit-targeting-internet-explorer-versions-9-through-11-identified-in-targeted-attacks.html⟩ (accessed 2014-11-24).

[14] 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評価（発表スライド），情報処理学会，マルウェア対策研究人材育成ワークショップ 2013（MWS2013）（オンライン）(2013)，入手先 ⟨http://www.iwsec.org/mws/2013/presentation/3B1-3-slide.pdf⟩（参照 2014-11-24）．

[15] HP Security Research Blog: Technical Analysis of CVE-2014-1761 RTF Vulnerability, HP (online) (2014), available from ⟨http://h30499.www3.hp.com/t5/HP-Security-Research-Blog/Technical-Analysis-of-CVE-2014-1761-RTF-Vulnerability/ba-p/6440048⟩ (accessed 2014-11-24).

[16] Cheng, Y., Zhou Z., Yu, M., Ding, X. and Deng, H.R.: ROPecker: A generic and practical approach for defending against ROP attacks, *Proc. 2014 Network and Distributed System Security Symposium* (*NDSS 2014*) (2014).

[17] Carlini, N. and Wagner, D.: ROP is Still Dangerous: Breaking Modern Defenses, *Proc. 23rd USENIX Security Conference on Security* (2014).

[18] Goktas, E., Athanasopoulos, E., Polychronakis, M., Bos, H. and Portokalidis, G.: Size Does Matter: Why Using Gadget-Chain Length to Prevent Code-Reuse Attack is Hard, *Proc. 23rd USENIX Security Conference on Security* (2014).

[19] 日本のセキュリティチーム：攻撃コードのトレンド（Exploitation Trend）—セキュリティインテリジェンスレポート第 16 版から，Microsoft（オンライン）(2014)，入手先 ⟨http://blogs.technet.com/b/jpsecurity/archive/2014/06/11/exploitation-trend-16.aspx⟩（参照 2014-11-24）．

[20] Tanaka, Y. and Goto, A.: n-ROPdetector: Proposal of a method to detect the ROP attack code on the network, *Proc. 7th Workshop on Cyber Security Analytics, Intelligence and Automation* (*Safeconfig 2014*) (2014).

[21] Ramirez-Silva, E. and Dacier, M.: Empirical Study of the Impact of Metasploit-Related Attacks in 4 Years of Attack Traces, *Proc. 12th Asian Computing Science Conference on Advances in Computer Science: Computer and Network Security* (*ASIAN'07*) (2007).

[22] Microsoft: Compound File Binary File Format, Microsoft (online) (2015), available from ⟨http://download.microsoft.com/download/9/5/E/95EF66AF-9026-4BB0-A41D-A4F81802D92C/%5BMS-CFB%5D.pdf⟩ (accessed 2015-05-23).

付　　録

表 12 において記載対象にできなかった他の正常検体の F4 の値について以下に示す．

検体種別	各正常検体の F4 の値
doc	6, 5, 5, 5, 6, 6, 7, 6, 2, 6, 5
rtf	5, 5, 3, 6, 4, 6, 6, 6, 6, 4, 6
xls	5, 6, 6, 5, 3, 5, 5
ppt	5, 6, 3, 5, 7, 6, 5

田中 恭之 （正会員）

1995 年立教大学理学部物理学科卒業．2015 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修士課程修了．同年同大学同研究科博士課程入学．1995 年日本電信電話株式会社入社．現在，NTT コミュニケーションズ株式会社技術開発部勤務．ネットワークセキュリティ技術，特に攻撃コードやマルウェア解析に関する研究開発に従事．

後藤 厚宏 （フェロー）

1984 年東京大学大学院工学系研究科情報工学専攻博士課程修了（工学博士）．NTT 研究所にてインターネットセキュリティ技術，高信頼クラウドコンピューティグ技術の研究開発等に従事．2011 年 7 月より情報セキュリティ大学院大学教授．情報処理学会理事．IEEE Computer Society Board of Governor．