3Z-9

Windows API フックを用いた通信監視による不正な PDF ファイルの検知

佐藤 両† 義則 隆之† 松井 拓也† 廣友 雅徳‡ 毛利 公美†† 神薗 雅紀‡‡ 白石 善明†
†名古屋工業大学 ‡佐賀大学 ††岐阜大学 ‡‡（株）セキュアブレイン

1. はじめに

標的型攻撃の 2011 年の発生件数は合計で 1277 件に上り，情報セキュリティ上の脅威であり続けている[1]．標的型攻撃では，特定の組織を狙って，攻撃コードを含む不正なファイルを添付した電子メールを送り，利用者の PC にマルウェアを感染させる．Portable Document Format ファイル(以下 PDF ファイルと称する)は悪用される主なファイル形式の一つである．添付された PDF ファイルには，PDF ファイル閲覧ソフトの脆弱性を突くコードが含まれており，利用者がファイルを開くとマルウェアをダウンロード・実行するようになっている．

不正な PDF ファイルに含まれる攻撃コードはアンチウイルスソフトに検知されないように難読化されていることが多い．JavaScript のフックを用いて攻撃コードの難読化を解除する方法が文献[2]，文献[3]で提案されているが，新たな難読化手法が現れるたびに解除方法を実装し直す必要がある．今後出現する新たな難読化に対応することを考えれば，難読化手法に依存しない検知方法が求められる．

本稿では，PDF ファイル閲覧ソフトの通信を監視することで不正な PDF ファイルを検知する方法を提案する．不正な PDF ファイルを開いた際に発生するマルウェア配布サイトへの通信を Windows API をフックすることで監視して検知する．正規の PDF ファイルも外部へ通信することがあるので，アクセス先の Web サイトの IP アドレスを取得する．提案する検知モジュールを実装し，Drive-by-Download Data by Marionette2012(以下 D3M2012)データセット[4]から取り出した不正な PDF ファイルを用いて検知率を評価した．

2. 悪性な JavaScript の難読化

PDF ファイルは動的なコンテンツを作るために，JavaScript のコードを組み込むことができる．PDF ではファイルを開く際に実行する JavaScript を定義することができ，標的型攻撃ではこの機能を悪用することでクライアント PC にマルウェアを感染させる．PDF ファイルを開く時に，閲覧ソフトの脆弱性を突く JavaScript を呼び出し，不正な引数を渡すことでバッファ・オーバーフローヒープスプレイを引き起こし不正な処理が実行される．不正な処理はシェルコードで実装されており，マルウェアを指定した URL からダウンロードする処理や，マルウェアを実行する処理がコードに記述されている．

JavaScript で記述された攻撃コードは，アンチウイルスソフトに検知されないように，難読化されていることが多い．難読化とは，ソースコードを人が読めない文字列に変換する処理であり，不正なリバースエンジニアリングからソースコードを保護するために使われる技術である．この難読化技術を悪用することで，コードの特徴的な部分を照合してマルウェアを検知するシグネチャベースのアンチウイルスソフトから攻撃コードが検知されないようにする．難読化手法は巧妙化しており，標準の JavaScript だけでなく，PDF 特有の JavaScript for Acrobat API(以下 Acrobat API)を使った方法が存在する．Acrobat API を用いた難読化では，PDF ファイル内に点在する情報を使って JavaScript が難読化される．

3. 難読化に対する既存のアプローチ

文献[2]では，JavaScript の関数フックを使って攻撃コードの難読化を解除する方法が提案されている．難読化された JavaScript を PDF ファイルから取り出し，関数をフックした JavaScript の実行環境で実行する．攻撃コードは難読化されたままでは実行できないので，必ず実行中に難読化が解除される点に着目している．難読化が解除された攻撃コードを JavaScript として実行する際に使われる関数をフックし，その引数からコードを得る．この方法の利点は，難読化が解除された JavaScript が得られるので，攻撃コードの有無だけでなく利用される脆弱性の種類が判定できる点である．しかし，解析する PDF ファイルが Acrobat API を使用して難読化している場合や，PDF のフォーマットに則っていない場合，自動で難読化を解除できないと述べられている．

文献[3]では，JavaScript の実行環境において一部の Acrobat API をエミュレートすることで Acrobat API を用いた難読化に対応している．Acrobat API が実行する処理と同じ処理を行う関数を実装することでエミュレートしている．しかし，Acrobat API は多数あり，全てをエミュレートすることは困難と述べられている．

PDF マルウェアを検知するために攻撃コードとなる JavaScript の難読化を解除する方法は，難読化手法に依存している．難読化手法の巧妙化に対応することを考えれば，これらに依存しない検知方法が求められる．

4. Windows API フックによる PDF 通信監視

PDF マルウェアに含まれる攻撃コードの難読化解除ではなく，発生する悪性の通信に着目する．検査環境下で PDF ファイルを開き，発生する外部への通信を監視することで，不正な PDF ファイルを検知するモジュールを提案する．通信監視による PDF マルウェアの検知を図 1 に示す．

4.1. 概要

PDF 通信監視モジュールを実現するためのアイデアは次の通りである．

(1) PDF 閲覧ソフトから発生する外部への通信を監視すること
(2) PDF 閲覧ソフトのバージョンに応じて動作を変える攻撃コードに対応すること
(3) アクセス先の IP アドレスを取得し確認すること

不正な PDF ファイルを開き脆弱性が突かれてから外部へアクセスする動作を検知するために，PDF 閲覧ソフトから発生する通信を監視する．通信の監視にはネットワーク入出力用の Windows API をフックする．

PDF 閲覧ソフトのバージョンごとに脆弱性の種類は異なり，バージョンを確認した上で突く脆弱性を変える攻撃コードを含む PDF ファイルが存在する．このタイプの PDF ファイルに対応するために，脆弱性が存在する PDF 閲覧ソフトのバージョンごとに仮想 OS を立ち上げる．

正規の PDF ファイルでも，開封時に閲覧権限の問い合わせ

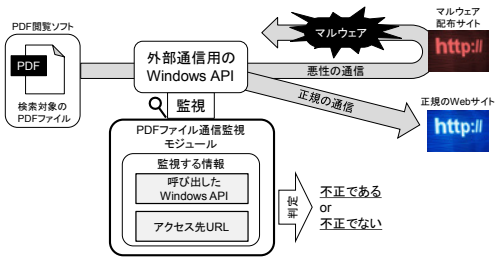

図 1 通信監視による PDF マルウェアの検知

Detection of Malicious PDF Files by Windows API Hook-based Network Monitoring
† Ryo Sato, Takayuki Yoshinori, Takuya Matsui and Yoshiaki Shiraishi・Nagoya Institute of Technology
‡ Masanori Hirotomo・Saga University
†† Masami Mohri・Gifu University
‡‡ Masaki Kamizono・SecureBrain Corp.

など正当な目的で外部と通信するものがある．正規の PDF による通信を判別するために，記録した外部への通信情報からアクセス先の IP アドレスを取得する．

4.2. PDF 検査環境とモジュールの構成

構築する検査環境とモジュールの構成を説明する．検査の際に PDF ファイルを実際に開くので，仮想 OS 上に環境を構築する．PDF 閲覧ソフトのバージョンごとに仮想 OS を作成し，各仮想 OS に PDF 通信監視モジュールを導入する．構築する PDF 検査環境とモジュールの構成を図 2 に示す．

[DLL ロード監視部]
PDF 閲覧ソフトにロードされる DLL 名を取得し照合する

[API フック部]
指定した Windows API をフックする

[API フック解除部]
Windows API のフックを外す

[API 検索部]
フックする API の場所を DLL の内部から見つけ出す

[API 呼出し記録部]
指定した Windows API が呼び出された場合，API 名をテキストファイル(API ログ)に記録する

[API フック関数]
フックした API が PDF 閲覧ソフトから呼出されたときに，処理の順序に割り込んで呼出される（フックする API 一つに対して API フック関数が一つ存在する）

4.3. 動作手順

PDF 通信監視モジュールの動作は次のようになる．

step1： PDF 閲覧ソフトにモジュールをロードする
step2： DLL をロードする Windows API を，フックした PDF 閲覧ソフト上で検査対象の PDF ファイルを開く
step3： DLL 監視部により，PDF 閲覧ソフトにロードされる DLL を監視する
step3-1： 通信用の Windows API を含む DLL のロードを検知した場合，API フック部により指定した Windows API をフックする
step4： 通信用の Windows API の呼び出しを監視する
step4-1： フックした Windows API が呼び出された場合，対応する API フック関数を呼出し，API 名を API 呼出し記録部へ通知する
step4-2： API 呼び出し記録部が API 名を API ログに記録する
step5： PDF 閲覧ソフトが終了するときに，API フック解除部によってフックを外す

5. 実装

PDF 通信監視モジュールを実装は，Microsoft 社の API フックライブラリである Detours[5]を使用した．Detours を利用することで Windows API をフックするモジュールを DLL として実装することができる．フックした Windows API の一覧は次の通りである．

WSAStartup	通信の初期化
Connect	サーバへ接続
Send, WSASend	データの送信
Recv, WSARecv	データの受信
URLDownloadToFile	ファイルのダウンロード
LoadLibrary	DLL のロード

これらの Windows API をフックする機能を実装した DLL を，DLL インジェクションによって Adobe Reader へロードする．DLL インジェクションとは，対象のアプリケーションに特定の DLL をロードさせる方法である．はじめに Adobe Reader へ通信監視用の DLL をロードしておき，その上で検査対象の PDF ファイルを開き動作を監視する．PDF 通信監視モジュールを構成する機能群の実装について説明する．DLL ロード監視部は，LoadLibrary のフック関数であり，Adobe Reader がロードする DLL の名前を照合する．通信用の DLL である

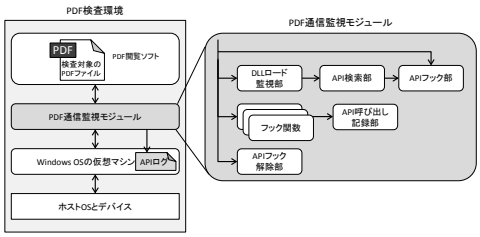

図 2　構築する PDF 検査環境とモジュールの構成

WS2_32.dll をロードしたことを検知すると，API 検索部を呼び出す．API 検索部は WS2_32.dll に含まれる API である WSAStartup，connect，send，recv，WSASend，WSARecv の場所を特定する．connect に対応する API フック関数では，接続先の IP アドレスを取得し，API 名と共に API 呼び出し記録部に通知する．また，DLL ロード監視部は WS2_32.dll とは他に，Urlmon.dll という DLL が Adobe Reader にロードされるか監視する．Urlmon.dll に含まれる API である URLDownloadtoFile は，指定した URL からファイルをダウンロードする際に使われ，標的型攻撃ではマルウェアのダウンロード時に使われる．

6. 評価

D3M 2012 データセットから取り出した PDF ファイル 100 個を使用して提案モジュールを評価する．これらの PDF ファイル全てをアンチウイルスソフト(NOD32)で検査した結果，98 個をマルウェアと判定した．残りの 2 個の PDF ファイルを Wepawet[6]で調べた結果，マルウェアであることを確認した．PDF 閲覧ソフトは Adobe Reader のバージョン 8 および 9 を使用し，仮想 OS を個別に立ち上げて検査環境を用意した．仮想化ソフトには VMware Player 5.0.0 を，仮想 OS には Windows XP Professional SP2 を使用し，この OS 上で実装したモジュールを動作させた．

実験方法は，検査環境上で対象の PDF ファイルを開き，API ログに書き込まれた情報（呼び出された API の名前や IP アドレス）を見ることで通信発生の有無を確認し，通信が発生していた場合に不正な PDF ファイルと判定した．

結果は，どちらか一方のバージョンでのみ通信を発生する PDF ファイルが多数存在することがわかった．両バージョンの検査結果を合わせると，100 個中 90 個の PDF ファイルの通信発生を検知できた．NOD32 で検知できなかった 2 個の PDF ファイルの通信発生を検知できた．

7. おわりに

本稿では，標的型攻撃で悪用される不正な PDF ファイルを検知するために，PDF 閲覧ソフトから発生する通信を監視するモジュールを提案した．提案モジュールを実装し，D3M 2012 データセットから取り出した PDF マルウェアを用いて検知率を評価した．検知率は 90%であり，アンチウイルスソフトでは検知できなかった PDF マルウェアを検知できることを確認した．

参考文献

[1] 警察庁情報通信局情報技術解析課：情報技術解析平成 23 年報～平成 23 年中のインターネット観測結果等～，平成 24 年 3 月
[2] 神薗雅紀，西田雅太，星澤裕二，“動的解析を利用した難読化 JavaScript コード解析システムの実装と評価”，MWS2010
[3] F.Schmitt, J.Gassen, E.Gerhards-Padilla, “PDF SCRUTINIZER: Detecting JavaScript-based Attacks in PDF Documents”, 2012
[4] MWS2012 実行委員会，研究用データセット MWS2012 Datasets について，http://www.iwsec.org/mws/2012/about.html#datasets
[5] Microsoft Research , Detours Express 3.0 : http://research.microsoft.com/en-us/projects/detours/
[6] University of California: Wepawet, Wepawet(online), http://wepawet.iseclab.org/index.php