PDMを用いたWEBブラウザ攻撃の動的解析

安藤　類央†　　外山　英夫‡

†情報通信研究機構ネットワークセキュリティ研究所
184-8795 東京都小金井市貫井北町４－２－１
ruo@nict.go.jp

‡(株）コムラッド
〒 168-0074 東京都杉並区上高井戸 1-8-17 Toya BUILDING7. 5F

あらまし 本論文では、Process Debug Manager (PDM) を用いたWEBブラウザの動的解析の手法を提案する。PDMはスクリプトを関数発行や変数代入時にハンドラを実装処理できる機構である。従来、悪意のあるWEBスクリプトは難読化あるいはバイナリコードが用いられることがあり、静的解析が難しい場合がある。PDMを用いることで攻撃をうけたWEBブラウザが発行する関数や変数の遷移を捉えることが可能であり、静的解析では難しい振る舞い抽出を行うことが可能である。提案手法では、いくつかのWEB攻撃化でのWEBブラウザの挙動をPDMを用いて可視化し、生成されたログの解析評価を行う。

Dynamic analysis of web browser exploitation using PDM

Ruo Ando†　　Hideo Toyama‡

†National Institute of Information and Communication Technology
4-2-1 Nukui-Kitamachi, Koganei, Tokyo 184-8795 Japan
ruo@nict.go.jp

‡Comrade Corp.
1-8-17 Kamitakaido, Suginami-ku, Tokyo, JAPAN

Abstract In this paper, we propose a new analyzing technique of obfuscated Java Script using process debug manager (PDM). Previously, many approaches of analyzing Web script is static which causes the obfuscation and metamorphism difficult to reveal. By using our module, we apply PDM and its step-execution components of IEPropertyInfo Interface to extract Property-Info Structure for tracking function call sequence and variables. Proposed system enables us to obtain the fine-grained log generated by high-level API. Step-execution and trace of our module can reveal the hidden behavior of obfuscated Java Script, which is difficult for static analysis

1　はじめに

近年、ＷＥＢアプリケーションの普及に伴い、ＷＥＢベースの攻撃やマルウェアの記述が急速に発達している。特に攻撃を成功させるための難読化やHeap Sprayなどの記述により、JavaScriptなどのコードは複雑化し、静的な解析の困難度が増している。本論文では、Process Debug Managerを用いてWindows OS上のプログラムやプロセスをSession Debug ManagerやDebug Engineを用いてトラック可能な手法を提案し、複雑なＷＥＢベースのScriptコード

の動的解析に適用する。評価実験では yahoo のサイト、ms09_072、ms10_002 などを対象として行った。

2 Process Debug Manager

Process Debug Manager (PDM) は Visual Studio の開発フレームワークから提供されるコンポーネントで、Windows OS 上のプログラムやプロセスを後述する Session Debug Manager (SDM) や Debug Engine (DE) との連動を可能にする。図 1 は、PDM の稼動位置と他のコンポーネントとの連携を示したものである。通常、シンボルの評価などは Debug Engine が行ない、Visual Studio のフレームワーク上で動作する PDM/SDM から操作し、トラッキングを行う。

3 処理フロー

提案モジュールの処理フローは、(1)PDM/SDM が発行したインターフェースを用いた (2) ブラウザのインスタンスの生成、(3) コールバック関数の登録、起動後のブレークポイントハンドラの設定と実行の 3 つに分けられる。

3.1 Web ブラウザの起動と injection

提案モジュールを process に inject するには、PDM.dll をロードし、ブラウザが起動しているプロセスを列挙する必要がある。この 2 つの動作双方とも IDebugProgramProvider2 系列の interface を用いる。PDM.dll のブラウザへのアタッチが成功した場合、自己実装した callback 関数を Web ブラウザの call chain にインサートすることが可能になる。

Browser launch and injection

PDM 操作のためのインターフェースを設定
$CComPtrIDebugProgramProvider2pdm$
PDM.dll のインスタンスを生成
$hr = pdm.CoCreateInstance$
ブラウザのプロセスをサーチ
$hr = pdm- > GetProviderProcessData$

3.2 コールバック関数の登録

コールバック関数の登録には、下記の WatchForProviderEvents を用いて行う。プロセス ID には、前述した GetProviderProcessData によってサーチしたブラウザの ID を指定し、第 6 引数に、実装したいコールバック関数の名前を取得する。コールバック関数の型は IDebugPortNotify2 であり、内部の任意のハンドラを格納することが可能である。

Browser launch and injection

$HRESULTWatchForProviderEvents$
$PROVIDERFLAGSFlags$,
$IDebugDefaultPort2pPort$,
$AD_PROCESS_IDprocessId$,
$CONSTGUIDARRAYEngineFilter$,
$REFGUIDguidLaunchingEngine$,
$IDebugPortNotify2pEventCallback$

3.3 ブレークポイントの処理

PDM/SDM が発行したインスタンスとインターフェースによりブラウザが起動されると、関数の実行やプロパティの変化時にハンドラが呼び出されることになる。

handling breakpoint

$HRESULTonHandleBreakPoint$
$IRemoteDebugApplicationThreadprpt$,
$BREAKREASONbr$,
$IActiveScriptErrorDebugpError$

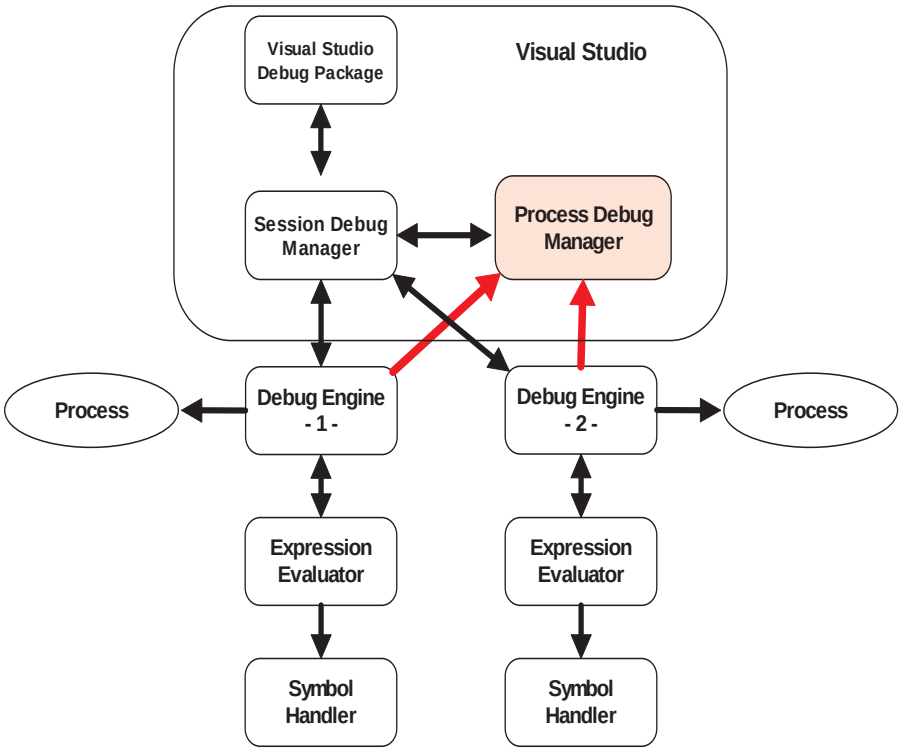

図 1: VSPackage of debugging components: VS Debug Package, Session Debug Manager, Process Debug Manger, Debug Engine, Expression Evaluator and Symbol Handler.

4 ログフォーマットと構造体

SDM/PDM ではファイル名の取得とプロパティの変化の記録用に下記の構造帯が利用可能である。ファイル名と実行された関数の情報取得には stack frame の記述子を、プロパティの変化には専用の構造体を用いる。

stack frame

$HRESULTNext$
$ULONGcelt,$
$DebugStackFrameDescriptorprgdsfd,$
$ULONGpceltFetched$

上記はIEnumDebugStackFrames Interfaceのメンバ関数から、取得可能である。これはスタックフレーム上を移動するもので、skip, next, clone などのメソッドがある。

property change

$typedefstructDebugPropertyInfo$
$DBGPROP_INFO_FLAGSdwValidFields$
$BSTRbstrName$
$BSTRbstrType$
$BSTRbstrValue$
$BSTRbstrFullName$
$DBGPROP_ATTRIB_FLAGSdwAttrib$
$IDebugProperty * pDebugProp$

上記はプロパティ情報の構造体である。BSTR型の変数に値などの情報が格納される。

図 2 は提案システムが稼動している状態で yahoo のサイトにアクセスした際のログである。プロセスＩＤやアクセス先のアドレスなどの他に、実行された関数名や、プロパティの変化などが記録されている。

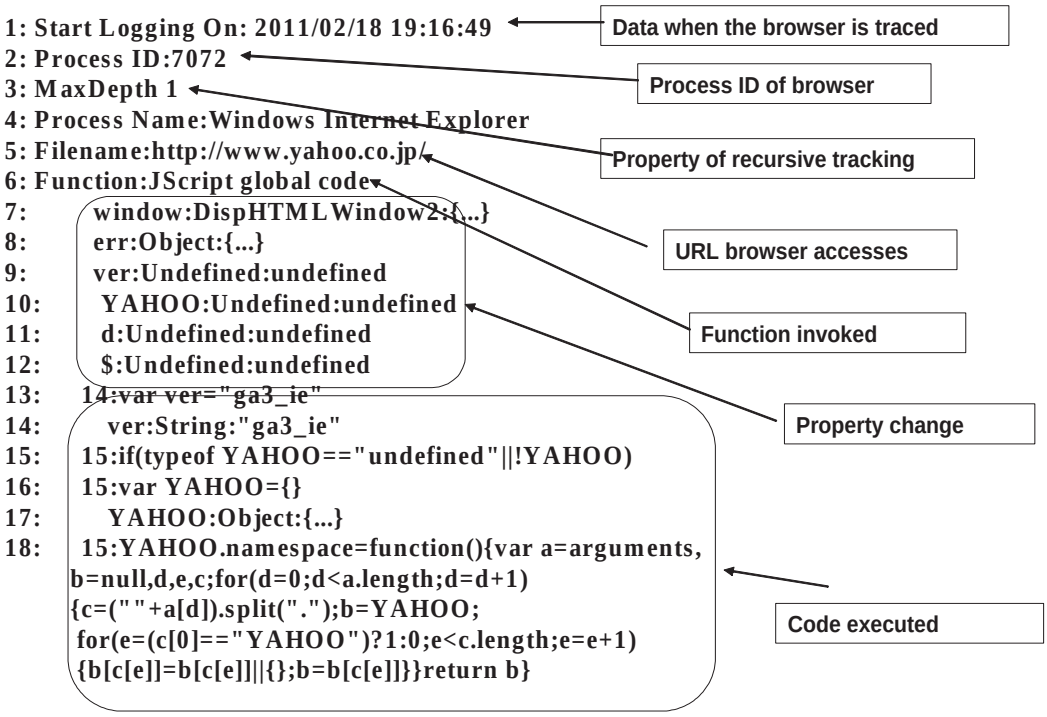

図 2: 提案システムが稼動している状態で yahoo のサイトにアクセスした際のログ

5 評価実験

本節では、Internet Explorer の脆弱性を突いた攻撃を 2 つ上げ、提案モジュールが組み込まれた状態でロギングを行った結果を示す。1 つは ActiveX コントロールの脆弱性 (MS09-072), もう 1 つは Google Aurora operation と認知されている脆弱性を扱う。

5.1 MS09-072

MS09-072 は、Microsoft Active Template Library (ATL) ヘッダーに組み込まれた ActiveX コントロールにより、リモートでコードが実行される脆弱性である図 3 は、提案システムが稼動している状態で MS09-072 の脆弱性を突いたサーバにアクセスした際のログである。ログの特徴は、文字列長が線形に増加する長いループが後半部分で展開されることである。

5.2 MS10-002

MS10-002 は、Internet Explorer に存在する 7 つの非公開で報告された脆弱性と 1 つの公開された脆弱性を指し、通称で Aurora Operation と呼ばれることがある。

- (1)XSS フィルターのスクリプト処理の脆弱性 - CVE-2009-4074
- (2)URL の検証の脆弱性 - CVE-2010-0027
- (3) 初期化されていないメモリの破損の脆弱性 - CVE-2010-0244
- (4) 初期化されていないメモリの破損の脆弱性 - CVE-2010-0245
- (5) 初期化されていないメモリの破損の脆弱性 - CVE-2010-0246
- (6) 初期化されていないメモリの破損の脆弱性 - CVE-2010-0247
- (7)HTML オブジェクトのメモリ破損の脆弱性 - CVE-2010-0248

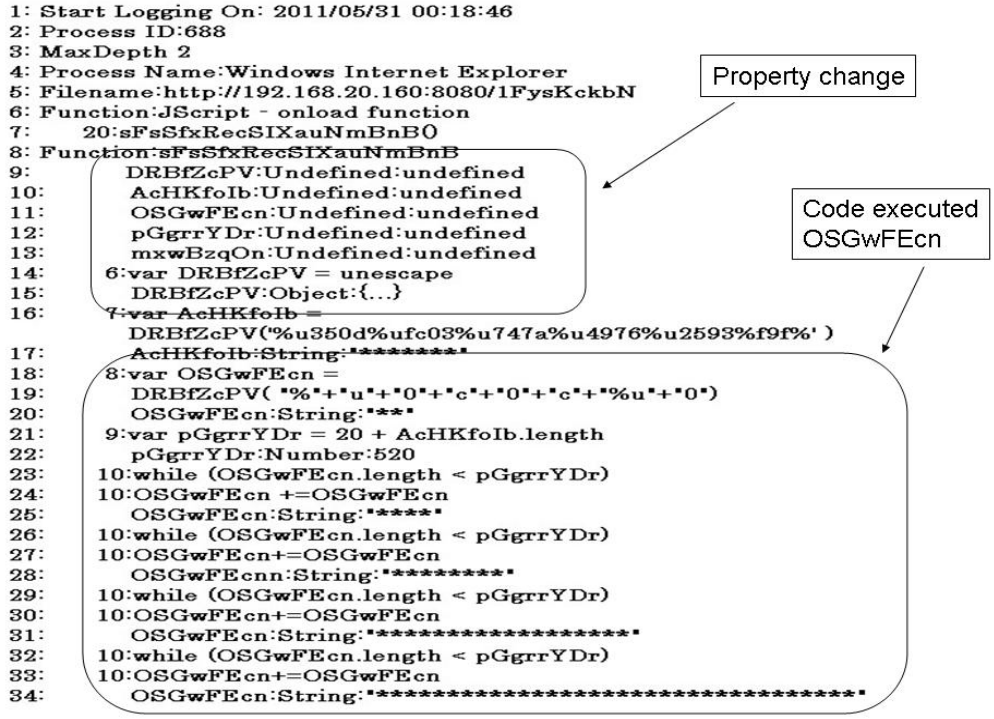

図 3: 提案システムが稼動している状態で MS09-072 の脆弱性を突いたサーバにアクセスした際のログ

- (8)HTML オブジェクトのメモリ破損の脆弱性 - CVE-2010-0249

図４は、提案システムが稼動している状態で MS10-002 の脆弱性を突いたサーバにアクセスした際のログである。ログの特徴は、前半部分で既知の property change、未知の property changeが実行された後、$parseInt(substring(i, i+2), 16))$ の長いループが展開される。

6　関連研究

WEB ベースのマルウェアの検出防御で代表的な研究に [1] がある。特に、動的解析は [2] で提案されている。難読化については [3] で提案されている。[3] は、polymorphic code の signature を自動的に生成するというものである。定理証明系を用いた難読化コードの解析は [4][5] で提案されている。[4] は導出法、[5] は項書き換えの方法が適用されている。[6] は web application のコードを静的に解析し、動的に（runtime）防御を行うものである。[7] ではランタイムにＷＥＢの脆弱性に対処する方法が提案されている。

7　まとめと今後の課題

本論文では、process debug manager (PDM) を用いた WEB ブラウザの動的解析の手法を提案した。PDM はスクリプトを関数発行や変数代入時にハンドラを実装処理できる機構である。従来、悪意のある WEB スクリプトは難読化あるいはバイナリコードが用いられることがあり、静的解析が難しい場合がある。PDM を用いることで攻撃をうけた WEB ブラウザが発行する関数や変数の遷移を捉えることが可能であり、静的解析では難しい振る舞い抽出を行うことが可能である。提案手法では、いくつかの WEB 攻撃化での WEB ブラウザの挙動を PDM を用いて可視化し、生成されたログの解析評価を行った。今後の課題としては、提案システムにメモリダンプなどの機能を付加することなどが上げられる。

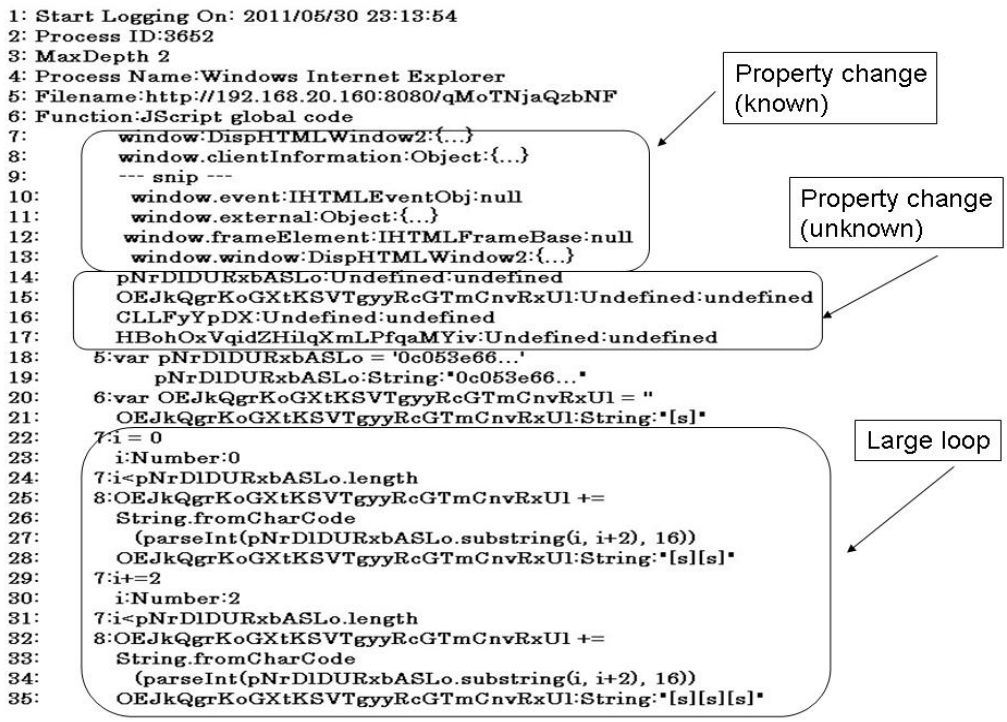

図 4: 提案システムが稼動している状態で MS10-002 の脆弱性を突いたサーバにアクセスした際のログ

参考文献

[1] V. Benjamin Livshits, Weidong Cui: Spectator: Detection and Containment of JavaScript Worms. USENIX Annual Technical Conference 2008: 335-348

[2] Emre Kiciman, Helen J. Wang: Live Monitoring: Using Adaptive Instrumentation and Analysis to Debug and Maintain Web Applications. HotOS 2007

[3] J. Newsome, B. Karp, and D. Song. Polygraph: Automatically generating signatures for polymorphic worms. In Proceedings of the 2005 IEEE Symposium on Security and Privacy, May 2005.

[4] Ruo Ando, Yoshiyasu Takefuji, ”Faster resolution based metamorphic virus detection using ATP control strategy ”, WSEAS TRANSACTIONS ON INFORMATION SCIENCE AND APPLICATIONS, Issue 2, Volume 3, February 2006. ISSN 1709-0832,pp260-2266,February 2006

[5] Gregory Blanc, Ruo Ando, and Youki Kadobayashi. Term-Rewriting Deobfuscation for Static Client-Side Scripting Malware Detection. In Proceedings of the 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS 2011), February 2011.

[6] Yao-Wen Huang, Fang Yu, Christian Hang, Chung-Hung Tsai, Der-Tsai Lee, Sy-Yen Kuo: Securing web application code by static analysis and runtime protection. WWW 2004: 40-52

[7] M. Martin, B. Livshits, and M. S. Lam. SecuriFly: Runtime vulnerability protection for Web applications. Technical report, Stanford University, Oct. 2006.