Drive-by Download攻撃の解析支援アプリケーションの開発と評価

青山 佑平[1]　吉井 章[2]　大倉 佳歩[1]　尾崎 幸也[1]　坂東 翼[3]　小林 孝史[1]

概要：Drive-by Download 攻撃によるマルウェアの自動取り込みが問題となっている．その問題点を明らかにするためには，攻撃フローを明らかにする必要があるが，その解析作業には専門的な知識と多大な時間を必要とする．我々は，ネットワーク上を流れる攻撃のためのトラフィックデータに含まれるデータ間の関係を明らかにするため，Drive-by Download 攻撃の解析作業を支援するアプリケーションを開発し，データ間の関係を木構造として視覚化することで攻撃の全体像を把握しやすくした．本アプリケーションの性能評価には，D3M データセットを活用し，全データの約 80%のトラフィックデータについて正しく関係を明らかにすることができた．

キーワード：MWS，Drive-by Download，作業工程の自動化，パケット間の関連性，攻撃の可視化

Development and evaluation of Analytical Support Application for Drive-by Download Attacks

Yuhei Aoyama[1]　Akira Yoshii[2]　Kaho Ohkura[1]　Koya Ozaki[1]　Tsubasa Bando[3]
Takashi Kobayashi[1]

Abstract: The automatic download of malware by Drive-by Download attacks becomes a big problem. It is necessary to clear the attack flow, however the analyzing work flow needs professional knowledges and massive time. We develop the application to support the analyzing workflows of Drive-by Download attacks, and all over the attack can be easier to understand by visualizing the relationship among the data as tree structure. We use the D3M datasets to evaluate the performance of this application. We can show the relationship of 80% traffic data of D3M datasets correctly.

Keywords: MWS, Drive-by Download, automation of analyzing workflow, relationship among traffic data, visualization of attacks

1. はじめに

インターネット・ブラウザを利用した情報システムやサービスが増えており，社会生活の中でもブラウザを使うことが非常に多くなっている．

Drive-by Download 攻撃は，ブラウザを使ってアクセスするWebサイト等に仕掛けられており，アクセスしたPCに自動的にマルウェア等がダウンロードされてしまうものである．

この攻撃手法は，様々な方法が絡み合って構成されているため，解析作業として，ネットワークを流れているトラフィックデータ（またはパケットともいう）の内容を調べ，その繋がりを明らかにする必要がある．この作業には専門的な知識と多大な時間を要するため，マルウェアのダウンロード元等の割り出しに非常に時間がかかる．

そこで，本研究では，トラフィックデータの間の関係を

[1] 関西大学
Kansai University
[2] MUS 情報システム株式会社
MUS Information Systems Co., Ltd.
[3] 関西大学大学院
Kansai University Graduate School

明らかにし，それらを木構造で表す可視化を行い，解析作業の従事者に結果のわかりやすさを提供するアプリケーションを開発した．このアプリケーションの使用により，解析時間の短縮・解析者の負担の軽減等を実現し，そして，マルウェア解析の研究分野での研究の発展の一助となることを目指す．

2. 関連研究

2.1 トラフィックデータ解析に関する研究

Drive-by Download 攻撃には効果的に攻撃対象者を絞るため多段階のリダイレクトが用いられているため，通信変遷を判定基準とした研究が多く行われている．寺田らの研究 [1] では，Drive-by Download 攻撃が正規のアプリケーションには見られない特有の通信挙動を示すことに着目し，攻撃通信における挙動と相関の高い通信を検知する手法が提案された．

佐藤らの研究 [2] では，Drive-by Download 攻撃が含まれる Web ページの判定基準として，Web ページの階層並びにドメインの生存期間などを勘案したアルゴリズムを提案し，本手法におけるアプローチが非常に有効であることが示された．

2.2 JavaScript に関する研究

Drive-by Download 攻撃には難読化が施されている JavaScript が多いという特徴に着目し，これを元に攻撃検知を行う研究が多く行われている．Cova らの研究 [3] では攻撃に使用されている JavaScript を HtmlUnit[4] を用いたサンドボックスで実行し，実行される脆弱性を特定することで攻撃の判定を行った．しかし，既知の脆弱性とのパターンマッチングを利用して攻撃判定を行っているため，未知の脆弱性に対しては攻撃検知が不可能である．また，HtmlUnit と本物のブラウザでの挙動の差異による攻撃の検出漏れなどが指摘されている．藤原らの研究 [5] では，難読化が施された JavaScript の静的解析を行い，難読化の特徴である文字列の変化に着目して攻撃検知の提案を行ったが，誤検知率が高いとの課題が挙げられた

また，Likarish らの研究 [6] 並びに西田らの研究 [7] では，難読化が施された悪性 JavaScript から特徴点を抽出し，文字出現頻度をパラメータとした機械学習を行うことで，高適合率の結果が示されている．Su らの研究 [8] では，情報理論的指標を用いることで Likarish らの研究と比較して高速なシステムを提案したが，文字数が少ない難読化が施された悪性 JavaScript は検知できないという課題が挙げられている．

3. 提案手法

3.1 アプリケーション概要

本研究では，Drive-by Download 攻撃の解析支援アプリケーションを開発する．本アプリケーションは，Drive-by Download 攻撃に関する解析を行う際，トラフィックデータを基に Web ページへアクセスを行った際発生する HTTP セッションを木構造として表し，視覚化することで解析者の作業を補助することを目的とする．また，本論文において提案するアプリケーションを PADDAR(Packet Analyzer for Drive-by Download Attack Researcher) と命名した．

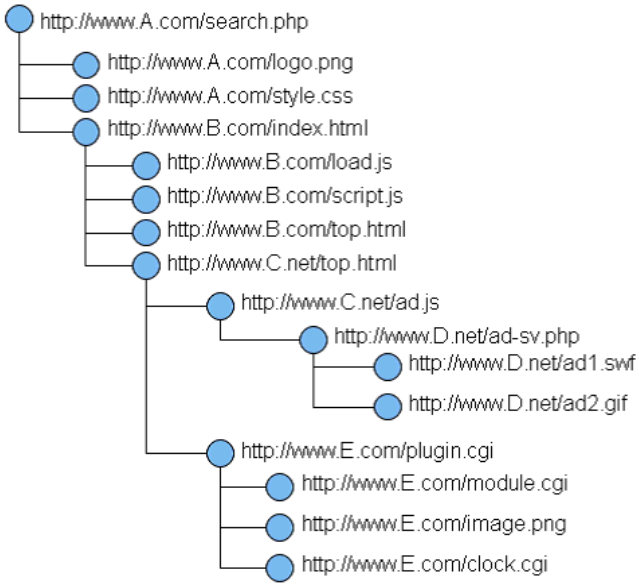

図 1 Web 関連性の木構造視覚化のイメージ
Fig. 1 visualizing example of the tree structure of web relationship

3.2 木構造による視覚化

HTTP セッションの関連性を明確にする解析の終了後，解析結果をもとにトラフィックデータ内の HTTP セッションを木構造として視覚化し，どの Web ページからどの Web ページへのアクセスが発生しているかを明確にする．木構造として視覚化した際のイメージを図 1 に示す．

PADDAR のシステムの流れを図 2 に示す．はじめに，解析者はリアルタイムにパケットをキャプチャし解析を行う「キャプチャモード」か，pcap 形式のトラフィックデータを読み込んで解析を行う「インポートモード」の選択を行う．どちらのモードを選択しても，トラフィックデータから TCP パケットを抽出した時点で同様のトラフィックデータとなり，その後 JavaScript の実行等の解析が行われる．解析結果として，HTTP セッションの関連性から構築した木構造を視覚化する．

3.2.1 キャプチャモード

キャプチャモードでは，リアルタイムにトラフィックデータの収集を行う．ユーザはまずはじめにパケットキャプチャを行うネットワークインタフェースを選択する．ネットワークインタフェースの選択後は四つの設定を行う．

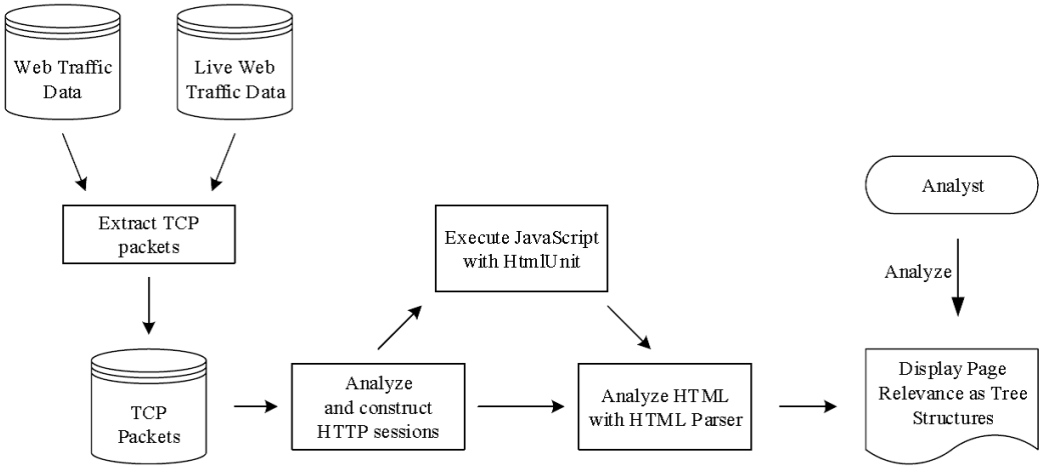

図 2 システムの流れ
Fig. 2 analyzing workflow

一つ目はパケットキャプチャ時のトラフィックデータの視覚化を行うかどうかの選択である．視覚化を行わない場合，後のトラフィックデータの出力設定を行うことで，PADDAR を単なるパケットキャプチャアプリケーションとしても使用することもできる．二つ目は HTTP セッションの解析精度を高める設定である．HTTP セッションの解析を行う際に，HTTP パケット情報を基に関連性を構築した後に JavaScript を実行するが，環境やトラフィックデータのサイズによっては,JavaScript の実行に多大な時間を要することが考えられる．そのため，JavaScript の実行を行わないで解析を行うことも可能にしている．三つめはキャプチャの開始とともに Web ブラウザを起動するかの設定である．本設定を有効にすることで，トラフィックデータの収集開始前に余計なトラフィックが混じることによる解析作業の負担を無くすことが可能である．最後は，キャプチャしたトラフィックデータの出力設定である．本設定を有効にしてトラフィックデータを出力することで，より詳細な解析を行うことができる．

また，これによって出力したデータを「インポートモード」で読み込むことで，同じ木構造の視覚化結果を得ることができる．

3.2.2 インポートモード

インポートモードは，既存のトラフィックデータを読み込み，HTTP セッションの木構造での視覚化を行う．これによって，ユーザはトラフィックデータ中のアクセスについて把握することが可能である．インポートモードでユーザが行う設定は，読み込むトラフィックデータのパスと解析精度に関する設定である．

解析精度に関する設定は「キャプチャモード」と同様のものである．

4. システムの実装

本アプリケーションは，JavaSE version 1.8 を用いて実装した．また，基幹となるライブラリとして，トラフィックデータのキャプチャには jNetPcap version 1,3,0, JavaScript の実装には HtmlUnit version2.19，HTML のパースには Jericho HTML Parser version 3.4 を使用した．jNetPcap の利用にあたっては，実行には Windows OS の場合は WinPcap，Unix/Linux の場合は libpcap のインストールが必要となる．PADDAR は Java を用いて開発されているが，jNetPcap が Mac OS には対応していないため，Windows OS 或いは Unix/Linux 環境が必要となる．

4.1 HTTP セッションを元にした木構造構築

Drive-by Download 攻撃は，攻撃が発生した Web ページからリダイレクトが繰り返されるため，階層の深い木構造が構築される．また，リダイレクトは時間差ではなく同時に行われることが多いため，後にトラフィックデータを基にアクセスの流れを再構築しようとしても，非常に困難なことが多い．しかし，これらを木構造として視覚化することで，どの Web ページからどの Web ページへとアクセスが発生したかというアクセスの流れを容易に把握することが可能である．Drive-by Download 攻撃は悪意のあるリダイレクトが発生している Web ページを特定し公開を停止させるなどの処置をとることで被害の拡大を抑えることができるため，木構造を構築し，Drive-by Download 攻撃の全体像を把握することを容易にする本手法は本攻撃の解析に対して有効であると言える．

4.2 トラフィックデータの解析

トラフィックデータ解析では，既存のトラフィックデー

タを読み込み，各トラフィックのパケットを基にトラフィックを再構築し，Web ページへアクセスした結果発生した HTTP セッションの流れを明らかにする．読み込むトラフィックデータの形式は，一般的なトラフィックデータ形式である pcap(packet capture) ファイルとする．

4.3 リアルタイムトラフィック解析

リアルタイムトラフィック解析では，Web ページへのアクセスとトラフィックデータの収集を同時に行う．収集したトラフィックデータを pcap ファイルとして出力することも可能とする．おもな用途としては研究者や Web ページの管理者が新たな Drive-by Download 攻撃を発見した際に，攻撃が仕掛けられている Web ページへアクセスした際の視覚的な挙動を把握しつつ，pcap ファイルを元に詳細な攻撃の解析を行うなどが挙げられる．

4.4 HTTP セッションの関連性の解析

PADDAR は，トラフィックデータを解析し，HTTP セッションの関連性を明らかにするために，複数の解析を順に行う．はじめにトラフィックデータ内の TCP パケットをすべて抽出し，それらのパケットから HTTP セッションを抽出する．その後，HTTP セッションに含まれる HTTP リクエスト並びに HTTP レスポンスパケットを参照し，暫定的な関連性を明らかにする．最後に各 HTTP セッションデータを復元し，JavaScript の実行並びに HTML 解析を行うことで，最終的な関連性を明確にする．

4.4.1 HTTP セッションの抽出

HTTP セッションの抽出は，トラフィックデータ内に含まれている TCP パケットから HTTP リクエストパケット及び HTTP レスポンスパケットを抽出することで行われる． はじめに TCP パケットを順に参照し，HTTP リクエストパケットが存在した場合は，当該パケットを暫定的に HTTP セッションの起点とする．その後，HTTP リクエストパケットに対する ACK パケットを受信していた場合は，1 件の HTTP セッションが発生したと判定する．ACK パケットを受信せずに RST パケットを受信した場合は，HTTP セッションは発生していないと判定する．PSH フラグを含むパケットを受信した場合は当該パケットを暫定的に HTTP セッションの終点とする．同一セッションに属する FIN フラグを含むパケットの存在が把握できるか，最後まで対応するパケットが確認できなかった場合は，暫定的な HTTP セッションの終点を HTTP セッションの終点として確定する．

4.4.2 関連性の解析

HTTP セッションの関連性の解析は，HTTP セッションに含まれる HTTP リクエストパケット並びに HTTP レスポンスパケットに含まれている情報を参照して行う．

HTTP リクエストパケットには，クライアントからサーバに対して行われる要求や，接続環境などの情報が記載されている．ステータス部には，サーバに対して行う要求の種類をあらわすリクエストメソッド (Request Method)，要求の対象となるファイルアドレス (Request URI)，通信を行う HTTP のバージョン (Request Version) の三種類の情報が含まれている．ヘッダ部には，クライアントが使用しているブラウザの種類や OS の情報を表したエージェント (User-Agent)，HTTP リクエストヘッダが発生した URL が示されたリファラ (Referer)，ユーザのログイン状態を管理するクッキー (Cookie) などの情報が含まれている．このうち，解析には，ステータス部の情報並びに Referer を用いる．

HTTP レスポンスパケットには，サーバからクライアントに対して，サーバの応答やデータの詳細などの情報が含まれている．ステータス部には，クライアントに対して通信を行う HTTP のバージョン，コンテンツ要求に対する応答コードであるステータス番号 (Status Code) 及びにその略称（Status Message）が存在する，ヘッダ部には HTTP サーバの詳細 (Server) や，最終更新日 (Last-Modifed)，転送先の URL が示されたロケーション (Location) などの情報が含まれている．メッセージ部には，Web サーバからクライアントに対して送信される HTML データなどが含まれている．これらのうち，解析に使用するのは，ステータス部及び Location の値，そしてメッセージ部の情報を用いる．

HTTP リクエストパケットに記載されている Referer には，どの URL から発生したアクセスであるかが示されている．また，HTTP レスポンスパケットに記載されているどこの URL へアクセスを行うかを表したリダイレクト先が示されている．HTTP セッションの関連性解析には，これらの Referer 及び Location 情報を用いる．

一方で，これらの情報は HTTP パケットに記載されていないことも多い．Location 情報は，リダイレクトが発生するときにのみ記載されるのに加え，Drive-by Download 攻撃では解析妨害のため，サーバが意図的に Location 情報を付与しないこともある．Referer 情報は，ユーザが Web ブラウザを開き，アドレスバーに URL を入力して Web ページにアクセスした時や，JavaScript などによって新規ウインドウが開かれ，Web ページへのアクセスが発生した場合など，HTTP リクエストが発生した URL が存在しない場合は記載されない．以上の場合，Location 並びに Referer 情報のみを使って HTTP セッションの関連性を明らかにすることはできない．よってでは，HTTP パケットに含まれる HTTP メッセージ内の Referer 及び Location 情報以外にも解析を行うことで，HTTP セッションの関連性の解析を行う．

表 1　HTTP リクエストヘッダ情報例
Table 1　example of HTTP Request Header

ヘッダ名	意味
Host	宛先の IP または FQDN
Connection	持続接続機能情報
Accept	受信可能なファイルタイプ
User-Agent	クライアント情報
Referer	リクエストが発生した URL

表 2　HTTP レスポンスヘッダ情報例
Table 2　example of HTTP Response Header

ヘッダ名	意味
Server	HTTP サーバ情報
Connection	持続接続機能情報
Date	応答日時
Last-Modified	最終更新日時
Location	リダイレクト先 URL

4.4.3　HTTP セッションデータの復元

暫定的な HTTP セッションの関連性が明らかになった後は，各 HTTP セッションデータを復元する．復元の手順としては，まず HTTP セッションデータに含まれるシーケンス番号を基にペイロードを組み立て，仮のデータとして復元する．次にデータがいつ圧縮されているかについて確認し，圧縮されている場合はデータの解凍を行う．解凍を行うファイルは HTML ファイルに限定し，解凍後のデータを HTTP セッションとして確定させる．圧縮の判断方法については，仮データはバイナリ配列となっているため，この配列の先頭の値を確認することで，圧縮されているかの判断を行う．データが圧縮されていない場合は，仮データを HTTP セッションとして確定させる．

5.　評価実験

5.1　評価方法

PADDAR による攻撃解析支援の有用性を確認する指標には，Web ページへアクセスした際の各 Web ページ間の関連性を木構造として正常に構築できた割合を用いる．評価に用いるトラフィックデータには D3M(Drive-by Download Dataset by Marionette)[9] を使用する．D3M には，Drive-by Download 攻撃が確認された Web ページに対して，ハニーポットを用いてアクセスを行った際のトラフィックデータが含まれている．このアクセスは，対象となる URL を Web ブラウザへ入力し，新規のアクセスを行ったものであるため，Drive-by Download 攻撃の入り口 Web ページへアクセスした結果となる．よって，トラフィックデータ内にて発生したほかの Web ページへのアクセスは，D3M に付録されている入り口 Web ページの URL のいずれかを親に持つ木構造で表すことができる．なお，

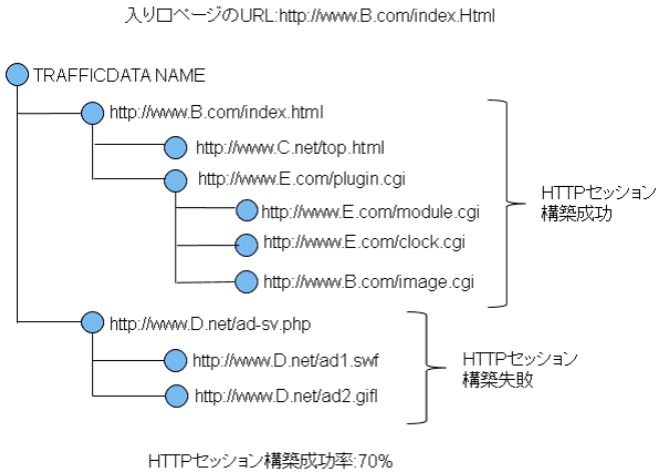

図 3　本評価手法による構築成功率 70%の例
Fig. 3　analyzed example by proposed method,successful rate:70%

本稿では，入り口 Web ページのことを入り口ページと定義し，HTTP セッションをセッションと定義する．

PADDAR における性能評価では，1 件の入り口ページへアクセスした結果発生した総セッション数に対して，木構造の構築が成功したセッション数の割合を表す構築の成功率を評価指標として用いる．なお，本評価では，1 件の Web ページへアクセスを行った結果発生した Web ページへのアクセスを，1 件のセッションと定義する．具体的には，入り口ページにアクセスした結果，入り口ページを除いて 9 件の Web ページへのアクセスが発生した場合は，合計 10 件のアクセスが発生したと表す．各セッションをノードとして考え，各ノードの親をたどって行き，その結果入り口ページの HTTP セッションノードが存在していた Web ページへのアクセスノードは構築成功とし，入り口ページが存在していない Web ページへのアクセスノードは構築失敗とする．例えば，合計 10 件のセッションが存在するトラフィックデータを解析した結果，7 件のセッションが親をたどると入り口ページの HTTP セッションノードであった場合の構築成功率は 70 %とする．なお，入り口ページの URL リストには記載されているものの，DNS の名前解決ができていないトラフィックデータなどは，HTTP セッションが作成されていないと判断し，評価結果から除外する．

D3M は Drive-by Download 攻撃のトラフィックデータを用いたものであるので，その中にはマルウェア等が含まれている可能性が非常に高い．そのため，評価実験はネットワークインタフェースを設定していない仮想マシンを作成し，インターネットから隔離した環境で行った．

5.2　評価結果

D3M を PADDAR の性能評価データとして用い，各デー

表 3　D3M を評価データとして用いた際の総合評価結果
Table 3　total evaluation results using D3M Datasets as evaluation data

データセット名	入り口ページ数	総セッション数	正常構築数	構築成功率
D3M2010	557	5,935	4,182	70.46%
D3M2011	283	2,362	1,319	55.84%
D3M2012	158	2,046	1,055	51.56%
D3M2013	46	326	211	64.72%
D3M2014	58	609	591	97.04%
D3M2015	312	12,763	11,851	92.85%
データセット総合	1,391	24,041	19,209	79.90%

タの入り口ページから派生したセッションの木構造構築成功率を確認する実験を行った．その結果，D3M2010 から D3M2015 のデータセットの約 80%のセッションの関連性を明確にし，関連付けることができた．各年度のデータセットにおける評価結果を表 3 に示す．

D3M の総入り口ページは 1,391 件，総セッション数は 24,041 件，構築成功セッション数は 19,209 件，総合構築成功率は 79.90%であった．また D3M2014，D3M2015 では 90%を超える成功率を示したが，D3M2010 から D3M2013 のデータセットにおいては，構築成功率が 70%を下回る結果となった．

データセットによって構築成功率にバラつきが発生した理由として，データセットそのものの特徴が原因であると考えられる．D3M2010 から D3M2013 では，同じ手法によって改ざんされたと思われる入り口ページが多く含まれていた．そのため，使用されている難読化や，リダイレクトの手法に対応できなかった場合は，ほかの同じ手法が用いられている HTTP セッションに対応できないので，HTTP セッション関連性構築成功率が 70%を下回る結果になったのではないかと考えられる．一方，以前から Drive-by Download 攻撃の研究者からデータセットに同じ手法での Drive-by Download 攻撃が含められているという声が上がっており，これを踏まえて D3M2015 では異なる手法を用いた攻撃が多く収集されている．これが D3M2015 での構築成功率が高かった理由だと考えられる

PADDAR を用いた本評価実験において，D3M2010 から D3M2015 までの合計では HTTP セッション構築成功率が約 80%，特に D3M2015 では HTTP セッション構築成功率が 90%を超えている．よって PADDAR は Drive-by Download 攻撃において，HTTP セッションの関連性構築を行い，本研究の目的である本攻撃に対する解析支援に活用できるといえる．

6. 考察

6.1 関連性を発見できなかった HTTP セッションへの対策

評価実験の際関連性を発見できなかった HTTP セッションについては，複数の原因が考えられる．その中の一つとして考えられるのが，Java Applet に用いられる Jar ファイルが，ほかのファイルを呼び出しているものである．PADDAR は，解析において HTML タグに実装された Java Applet オブジェクトが Jar ファイルを呼び出した HTTP セッションには対応している．しかし，その Jar ファイルから呼び出される class ファイル等の解析には対応していないため，Java Applet を用いた攻撃が存在した場合は，HTTP セッションにおける木構造構築成功率が落ちてしまう．

本課題に対しては，Java Applet で呼び出された Jar ファイルの解析を行い，さらに呼び出される class ファイルの HTTP セッションについても補足する必要がある．しかし誤ってファイルを実行してしまうと解析者の環境がマルウェアに感染する可能性も考えられるので，PADDAR にサンドボックス機能を持ち合わせるなどの対策が必要である．

6.2 実行に失敗する JavaScript への対策

評価実験を通して，PADDAR には HTTP セッションに含まれている JavaScript を実行する際に，一部の JavaScript が実行できないという問題が発見された．本課題を解決するためには，JavaScript の実行に現在使用している HtmlUnit を用いるのではなく，他の手法を用いる必要がある．

考えられる対応策としては，Mozilla Developer Network によって作成された Java で記述された JavaScript エンジンである Rhino[10] と，John Resig によって開発された env.js[11] を組み合わせたシステムを構築する手法が考えられる．env.js はブラウザの挙動をエミュレートする,，JavaScript によって記述されたライブラリである．JavaScript の実行方法としては，Rhino 用にコンパイルした env.jp を Rhino に読み込ませておき，さらに実行した JavaScript が含まれたファイルを読み込ませることで，JavaScript を実行しその結果を得ることができる．本稿における評価実験の中で，HtmlUnit を用いて実行できなかった難読化された JavaScript コードを Rhino 及び env..jp を用いたプログラムで実行したところ，正常に実行できたも

のがいくつか存在した．そのためこの手法を用いることで本課題の改善に繋がると考えられる．

しかし env.jp はすでに公式 Web サイトが閉鎖されており，開発がすでに中断，あるいは終了している可能性がある．そのため，今後 JavaScript に新たな仕様が追加され，それを利用した難読化が行われた場合，これに対応できない可能性がある．そのため，最も現実的な対応策としては，HtmlUnit と Rhino 及び env.jp を用いた実行を平行して行うことが考えられる．

6.3 HTTPS 通信への対応

PADDAR は HTTP セッションの解析にのみ対応しており，HTTPS セッションの解析には対応していない．HTTPS は暗号化された通信であり，キャプチャを行ったとしてもその内容を得ることはできず，関連性解析を行えない．Drive-by Download 攻撃の攻撃者はこのことを利用し，リダイレクトの途中に HTTPS 通信を含めることで，解析を妨害することもある．

本課題への対応策としては，PADDAR を MITM Proxy（Man In The Middle Proxy）として実装するものが考えられる．MITM Proxy の仕組みは，あらかじめクライアントに対して MITM Proxy が発行する証明書をインストールしておき，クライアントに代わって MITM Proxy が HTTPS サーバと通信を行うようになっている．MITM Proxy が介在したネットワークでは，二つの HTTPS 通信が行われることになる．そして，MITM Proxy は両方の HTTPS 通信の当事者であるため，PADDAR に MITM Proxy を実装することで，クライアントが送信したデータ及びサーバから受信したデータを PADDAR が解析することが可能になる．また，MITM Proxy は中間者攻撃を行う Proxy サーバと言い換えることもできるが，本アプリケーションは一般的な用途で使用するアプリケーションではなく，研究者並びに解析者が，攻撃の解析を行う際に使用するものであるため，問題は無いと考える．

しかし，MITM Proxy を通さずに HTTPS 通信を行ったトラフィックデータは解析を行うことができない．そのため，本手法は実際に Web ページへアクセスし，トラフィックデータを収集して解析を行う「キャプチャモード」でのみ有効であると言える．

7. まとめ

本研究では，Drive-by Download 攻撃の解析支援アプリケーション PADDAR を開発し，D3M を用いた評価実験を通してその有用性を明らかにすることができた．評価実験で見えた改善点として，関連性を発見できなかった HTTP セッション，HTTPS 通信に対応していないこと，実行に失敗する JavaScript 等があるため，改善していく必要がある．今後は，前述した点の改善とともに，さらに最新の Drive-by Download 攻撃に対応させ，広く活用されることを目標とし，開発を続けていく．

参考文献

[1] 寺田成吾，小林峻，小出和弘，羽藤逸文，瀬戸口武研，道根慶治，山下康一，“ネットワーク通信の相関性に基づく Drive-by Download 攻撃検知手法”，Computer Security Symposium 2015，pp. 1-7，2015.

[2] 佐藤祐磨，中村嘉隆，高橋修，“通信遷移と URL の属性情報を用いた悪性リダイレクト防止手法”，Computer Security Symposium 2015，pp. 8-15，2015.

[3] Marco Cova, Christopher Kruegel, Giovanni Vigna, “Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code”，19th international conference on World wide web，pp. 281-290，2010.

[4] Gargoyle Software，“HtmlUnit”，http://htmlunit.sourceforge.net/，2016 年 8 月 12 日確認.

[5] 藤原寛高，Gregory Blanc，櫨山寛章，飯村卓司，門林雄基，“難読化の特徴を利用したドライブバイダウンロード攻撃検知方式の実装と評価”，電子情報通信学会 信学技法 IEICE Technical Report ICSS2014-71，pp. 49-54，2015.

[6] Peter Likarish, Eunjin Jung, Insoon Jo, “Obfuscated Malicious JavaScript Detection using Classification Techniques”，Malicious and Unwanted Software 4th International Conference，pp. 47-53，2009.

[7] 西田雅太，星澤裕二，笠間貴弘，衛藤将史，井上大介，中尾康二，“文字列出現頻度をパラメータとした機械学習による悪質な難読化 JavaScript の検出”，情報処理学会研究報告 Vol.2014-DPS-158 No21/Vol.2014-CSEC-64 No.21，pp. 1-7，2014.

[8] Jiawei Su，Katsunari Yoshiaki，Junji Shikata，Tsutomu Matsumoto，“Detecting obfuscated malicious JavaScript based on information-theoretic measures and novelty detection”，Computer Security Symposium 2015，pp. 226-233，2015.

[9] 神薗雅紀，秋山満昭，笠間貴弘，村上純一，畑田光弘，寺田真敏，“マルウェア対策のための研究用データセット～MWS Datasets 2015～”，情報処理学会研究報 Vol.2015-CSEC-70 No.6/Vol.2015-SPT-14 No.6，pp. 1-8，2015.

[10] Mozilla Developer Network，Rhino | MDL，https://developer.mozilla.org/ja/docs/Rhino，2016 年 8 月 12 日確認.

[11] Web Archive，Envjs，http://www.envjs.com/（Web Archive にて確認），2016 年 8 月 12 日確認.