L-021

Proxy と ICAP を用いた Drive-by Download 攻撃防御システムの開発
The development of Drive-by Download attack prevention system using Proxy and ICAP

山崎 雅斗† 小林 孝史‡
Masato Yamazaki Takashi Kobayashi

1. はじめに

Web 技術の進歩により，JavaScript を用いることで，動きのある Web ページの作成や Web ページのリアルタイム更新等が可能になり，Web ページの表現の幅が広がった．しかしその反面，JavaScript を悪用することで Web ページの閲覧者に気づかれることなく秘密裏にマルウェアをダウンロードさせる攻撃が観測されている．

JavaScript の解析手法として，実際にスクリプトを動作させて解析を行う動的解析と，スクリプトを実行せずに解析を行う静的解析の 2 種類が存在する．しかし，Drive-by Download 攻撃（以下，DbD 攻撃）に用いられる JavaScript は，タイマーやイベント処理を用いた遅延処理や，ユーザが特定のブラウザを使用している場合のみ攻撃を実行する処理，攻撃に用いる JavaScript の難読化処理等により，動的解析・静的解析のどちらか一方だけでは解析が困難となっている．本研究では，二つの JavaScript 解析手法のうち静的解析に着目し，静的解析の課題の一つである JavaScript の難読化処理に影響されにくい解析手法の提案と実装，評価を行った．

2. 本研究のシステム

本節では，本研究のシステムの概要と，提案した JavaScript の良悪判定機構について述べる．

2.1 システムの概要

本システムでは，図 1 のように Proxy サーバと ICAP サーバを連携させて JavaScript の良悪判定を行う．サーバの OS には CentOS 7.4 を用い，Proxy サーバは Squid v3.5，ICAP サーバは pyicap [1]をベースに実装したものを用いて構築した．ユーザが Proxy サーバを経由して Web ページにアクセスした際の流れは次のようになる．まず，ユーザが Web サーバへの HTTP リクエストを送信すると，Proxy サーバがユーザの代わりに Web サーバへ HTTP リクエストを行い，Web サーバからの HTTP レスポンスを受け取る．次に，受け取った HTTP レスポンスに JavaScript が含まれていれば，ICAP サーバで JavaScript の解析を行う．そして，解析の結果悪性であると判定された場合はブラウザに警告を表示し，悪性でないと判定された場合は Web サーバからの HTTP レスポンスをそのままユーザへ返却する．

ICAP サーバでの処理の流れを図 2 に示す．ICAP サーバでは，まず Proxy サーバから受け取った ICAP のメッセージから HTTP レスポンスボディに相当する部分のみを抽出する．次に，抽出した HTTP レスポンスボディを HTML パーサを用いて解析し，JavaScript が含まれているかどうかを検査する．検査の結果，JavaScript が含まれていない場合，書き換えが必要ないという意味の「204 No Content」を Proxy サーバに返却し，ICAP サーバでの解析を終了する．

† 関西大学大学院 Kansai University Graduate School
‡ 関西大学 Kansai University

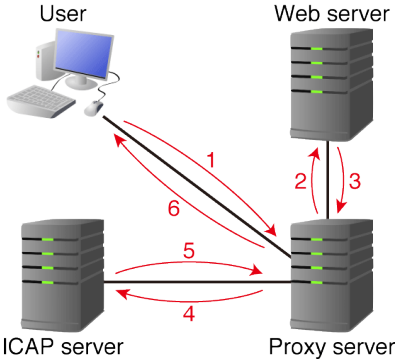

図 1 システムの概要

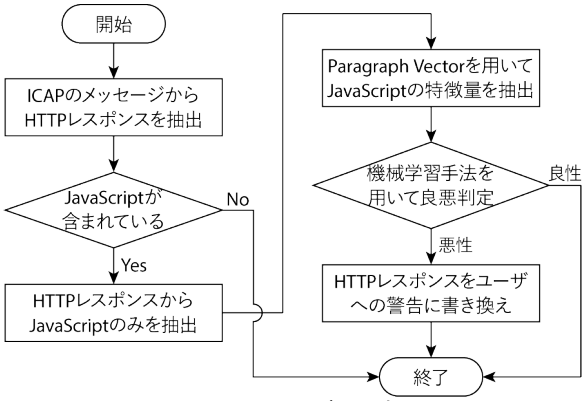

図 2 ICAP サーバでの処理

検査の結果，JavaScript が含まれている場合，続いて JavaScript の解析に移る．HTML から JavaScript のみを抽出し，本システムを用いて Paragraph Vector による特徴量の算出と良悪判定を行う．良悪判定の結果，悪性 JavaScript が含まれないと判定された場合は「204 No Content」を返却し，悪性 JavaScript が含まれると判定された場合は HTTP レスポンスの内容をユーザへの警告に書き換え，Proxy サーバへ返却する．

2.2 JavaScript の良悪判定

JavaScript の解析は，三須らの研究[2]を応用し，Paragraph Vector で算出した特徴量のベクトルをもとに行う．

JavaScript の良悪判定を行う事前準備として，Paragraph Vector を用いて良性・悪性 JavaScript の特徴量を算出し，算出された特徴量をもとに機械学習モデルを作成する．まず，JavaScript を処理する前処理として，関数名を_FUNC，変数名を_VAR にそれぞれ置き換える．前処理後，良性・悪性それぞれの JavaScript を Paragraph Vector を用いて特徴量を算出する．次に，機械学習を用いて学習を行うため，算出した特徴量に良性・悪性のラベル付けを行う．最後に，

ラベル付けした特徴量を機械学習手法の入力として与えて学習を行い，学習結果をモデルとして出力する．

事前準備で作成したモデルを用いて，解析対象となる JavaScript の良悪判定を行う．JavaScript の良悪判定は次のような手順で行う．まず，機械学習モデルの作成と同様の手順で JavaScript の特徴量を算出する．その後，算出された特徴量と事前に作成した機械学習モデルを用い，JavaScript の良悪判定を行う．

3. 実験と考察

本研究で構築したシステムの性能を検証するため，実験を行った．実験では，本システムで JavaScript の良悪を正しく判定できるかどうかと，ユーザが本システムを用いて悪意あるスクリプトを含む Web ページにアクセスしたとき正常に警告を表示できるかどうかの 2 点を検証する．この 2 点について検証を行うため，本研究で実装した良悪判定機構を用いて JavaScript の良悪を判定する実験と，Proxy と ICAP を用いて構築した本システムの動作を検証する実験を行った．実験では，Paragraph Vector の実装に Doc2Vec を，Paragraph Vector から出力されたベクトルを用いて二値分類を行う機械学習手法にはニューラルネットワークの一つである多層パーセプトロンを用いた．

3.1 JavaScript 良悪判定機構の性能評価

Paragraph Vector を用いて実装した JavaScript 良悪判定機構の精度を検証するため，実装した良悪判定機構を用いて JavaScript の良悪を判定する実験を行った．実験では，良性 JavaScript として通常のブラウジングから得た JavaScript を，悪性 JavaScript として MWS Datasets[3]内の Drive-by Download Data by Marionette Dataset（以下，D3M Dataset）に含まれる JavaScript をそれぞれ 1242 件ずつ使用した．

交差検定の結果，悪性 JavaScript の検知率は 99.56%，False Positive が 0.40%，False Negative が 0.48%となり，実装した良悪判定機構により高い精度で JavaScript の良悪判定を行うことができた．

3.2 Proxy と ICAP を用いたシステムの性能評価

ユーザが本システムを用いて悪意あるスクリプトを含む Web ページにアクセスしたとき正常に警告を表示できるかどうかを検証するため，ユーザ側で本システムを有効にした状態で実験用に準備した 4 種類の悪意あるスクリプトを含む Web ページへアクセスを行い，正常に警告画面が表示されるかどうかを検証した．作成した Web ページに含まれる悪意あるスクリプトの挿入方法と本システムを用いて Web ページへアクセスしたときの判定結果を表 1 に示す．実験の結果，用意した 4 種類の Web ページの内 3 種類について正常に警告画面を表示することができた．Web ページ 3 に含まれる悪意あるスクリプトは，変数名や関数名等が変わるような難読化処理は施されておらず，スペースや改行を取り除くことで読みにくくする minify と呼ばれる処理が加えられているのみであった．そのため，Web ページ 3 に含まれるスクリプトの特徴量と DbD 攻撃に用いられる他のスクリプトの特徴量の類似度がそれほど高い値にならず，良性と判定されたと考えられる．JavaScript に難読化処理を施すと，変数の値は直接格納されずに文字列置換等の処理を経てから格納される．そのため，難読化処理が施されているものと施されていないものの間で特徴量に差が生じる．機械学習モデルを作成する時に使用した悪性 JavaScript には，難読化処理が施されていないものや minify 処理のみが加えられているものも含まれていた．

表 1　作成した Web ページへの悪意あるスクリプト挿入方法とアクセス後の判定結果

Web ページ	悪意あるスクリプトの挿入方法	アクセス後の判定結果
Web ページ 1	<script> タグに直接記述	悪性
Web ページ 2	<script> タグに直接記述	悪性
Web ページ 3	src 属性を用いて外部から読み込み	良性
Web ページ 4	src 属性を用いて外部から読み込み	悪性

3.3 考察

上記 2 種類の実験に用いたデータを調査したところ，悪性 JavaScript で用いられる難読化手法は数が限られており，同じ手法で難読化処理を施しているものが多かった．そのため，難読化処理によって悪性 JavaScript の中で似たようなソースコードを持つものが多くなり，Paragraph Vector で算出される特徴量の類似度が高くなったために高い精度で悪性 JavaScript の検知を行うことができたと考えられる．また，実験結果より，D3M Dataset に含まれる攻撃と類似する DbD 攻撃に対しては高い精度で悪性 JavaScript を検知できると推測できる．

4. おわりに

本システムでは，Paragraph Vector を用いて JavaScript の特徴量を算出し，その特徴量をもとに機械学習手法を用いて二値分類を行うことで，高い精度で JavaScript の良悪判定が行えることを検証した．Paragraph Vector を用いることで，動的解析だけでは対応できなかった悪性 JavaScript の検知も可能になると考えられる．しかし，今回の実験では Paragraph Vector から算出された特徴量を学習する機械学習手法を仮に多層パーセプトロンと決定したため，他の機械学習手法との比較検討を行う必要がある．また，本研究では悪性 JavaScript として D3M Dataset を用いているが，D3M Dataset に含まれない種類の DbD 攻撃には対応できない可能性がある．そのため，Paragraph Vector での計算前に行う下処理の段階で JavaScript の抽象化を行う等，データセットに影響されにくい検知システムの検討を進める．

参考文献

[1] netom/pyicap: A lightweight python framework for writing ICAP services，https://github.com/netom/pyicap，2018 年 6 月 20 日確認.
[2] 三須 剛史，巻島 和雄，岡田 晃市郎，岩本 一樹，“ソースコードの類似度に基づく悪性 JavaScript の分類に関する一検討”，Computer Security Symposium 2017 pp.378-384，2017.
[3] 高田 雄太，寺田 真敏，村上 純一，笠間 貴弘，吉岡 克成，畑田 光弘: マルウェア対策のための研究用データセット～MWS Datasets 2016～情報処理学会研究報告 2016-CSEC-74，pp.1-8，2016.