マルウェア感染ステップのファイルタイプ遷移に基づいた Drive-by Download 攻撃検知手法

進藤 康孝 † 佐藤 彰洋 ‡ 中村 豊 ‡ 飯田 勝吉 †

† 東京工業大学大学院 通信情報工学専攻
152-8550 東京都目黒区大岡山 2-12-1
shindo@netsys.ce.titech.ac.jp
iida@gsic.titech.ac.jp

‡ 九州工業大学 情報科学センター
804-8550 福岡県北九州市戸畑区仙水町 1-1
{satoh,yutaka-n}@isc.kyutech.ac.jp

あらまし 不正に改ざんされた Web サイトを閲覧したクライアント PC に秘密裏にマルウェアをダウンロードさせる Drive-by Download(DbD) 攻撃が巧妙化を続けている．特に，不正なスクリプトコードが高度に難読化されることでコードの特徴による攻撃検知は難しくなる場合があり，また難読化されたペイロード情報の解析にはコストがかかる．そこで本稿では，難読化情報の解析を用いず，ネットワーク管理者が容易に入手できる情報を用いた軽量な異常検知手法の開発をめざすため，DbD 攻撃の感染ステップに伴うファイルタイプ遷移の特徴に着目し，機械学習によって攻撃を検知する手法を提案，評価する．その結果，JavaScript による DbD 攻撃を平均 82.3%の精度で判定できた．

Detection of Drive-by Download Attacks Based on File Type Transition in Malware Infection Process

Yasutaka Shindo† Akihiro Satoh‡ Yutaka Nakamura‡ Katsuyoshi Iida†

†Department of Communications and Computer Engineering, Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, JAPAN
shindo@netsys.ce.titech.ac.jp
iida@gsic.titech.ac.jp

‡Information Science Center, Kyushu Institute of Technology
1-1 Sensuicho, Tobata-ku, Kitakyushu 804-8550, JAPAN
{satoh,yutaka-n}@isc.kyutech.ac.jp

Abstract Drive-by download (DbD) attacks have been increasing in recent years. The major problem of detecting DbD attacks is that the attackers use code obfuscation techniques to be difficult to detect the attacks. In this paper, we propose a lightweight approach to detect DbD attacks by using Content-Type information in malware infection process, which does not require decoding obfuscated code. With the information, we use a machine learning technique to distinguish between malicious and benign communications.

1　はじめに

クライアント PC を狙った Drive-by Download 攻撃 (以下 DbD 攻撃) の脅威が深刻化している．DbD 攻撃とは，改ざんされた Web サイトにアクセスしたクライアント PC が，不正なスクリプトコードによってマルウェアをダウンロードするよう誘導させられるサイバー攻撃である．DbD 攻撃は秘密裏に攻撃が進行するので，クライアントは感染したことに気づかない場合が多いため問題となる．

そのような DbD 攻撃が猛威をふるう中，各エンタープライズ，キャンパスネットワークの管理者にとって自ネットワーク内での感染を早急に検知することは重要である．しかし，悪性通信を発生させる不正なスクリプトコードは高度に難読化されており，一般的に解析は容易でない．そのため，組織ネットワーク下で通信ログとして容易に取得でき，かつ解析に多大なコストを要さない情報によって DbD 攻撃の検知が行えれば，ネットワーク管理者はより迅速に脅威への対応が可能となる．

本稿では，DbD 攻撃の進行に伴って HTTP リクエストで取得するファイルタイプが遷移していく [1, 2] ことに着目し，容易に取得できる情報として Conent-Type 情報を用い，ファイルタイプ遷移を特徴ベクトルとした機械学習によって悪性通信と良性通信を区別する DbD 攻撃検知手法を提案，評価する．2 章では，関連技術として DbD 攻撃の説明と先行研究について述べる．3 章では，ファイルタイプ遷移に基づいた DbD 攻撃検知手法について述べる．4 章では，提案手法の評価結果とその考察について述べる．最後に 5 章では，まとめと今後の課題を述べる．

2　関連技術

本章では関連技術として，DbD 攻撃の仕組みと先行研究について述べる．

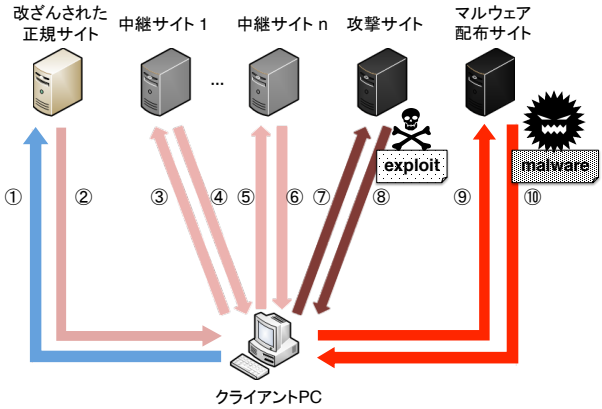

図 1: DbD 攻撃によるマルウェア感染ステップ

2.1　DbD(Drive-by Download) 攻撃

典型的な DbD 攻撃によるマルウェア感染ステップを図 1 に示す．DbD 攻撃は，攻撃の進行に伴ってフェッチ，すなわち HTTP リクエストにより取得されるファイルの種類が変わり，その種類によって大きく 3 つのステップに分けられる．

1. 不正スクリプトコード (①〜⑥)

   攻撃者は予め正規サイトを改ざんすることで，DbD 攻撃のトリガとなる入口サイトを用意する．改ざんされたサイトにアクセス (①) したクライアントは，不正な iframe タグを含む HTML ファイルをフェッチし (②)，JavaScript 等による中継サイトへの不正リダイレクト (③〜⑥) を経て，攻撃サイトへ誘導させられる．これら不正スクリプトコードは多くの場合，検知を回避するための難読化が施されている．

2. エクスプロイト (⑦〜⑧)

   攻撃サイトへ誘導させられたクライアント PC は，OS，Web ブラウザ又はそのプラグインの脆弱性を悪用するエクスプロイトをフェッチさせられる．悪用される脆弱性として，主に Adobe Flash Player，Adobe Reader，JRE/JDK，などが挙げられる．

3. マルウェア (⑨〜⑩)

   その結果制御を奪われたクライアント PC は，マルウェア配布サイトへリクエストを

送信する．その結果，トロイの木馬やスパイウェアなど，実行ファイル形式のマルウェアがダウンロードされ，攻撃は完了する．

2.2 先行研究

DbD 攻撃の検知に関する研究として，ペイロード情報を使った研究，ヘッダ情報を使った研究が挙げられる．

ペイロード情報を使った研究では，難読化された不正スクリプトコードを解析し，そのコードの特徴による攻撃の検知を目指している．Cova ら [4] らは，不正 JavaScript コードの特定の関数呼び出しや文字列上の特徴など，動的解析によるコードの包括的な情報を使い，機械学習によって悪意のあるコードや URL を検知する手法を提案している．

ヘッダ情報を使った研究では，DbD 攻撃の際の HTTP リクエスト，レスポンスヘッダから取得できる情報で攻撃検知することを目指している．寺田ら [5] は，D3M データセット 2010 に含まれる通信データの不正リダイレクトを分析し，Web アクセス遷移の考慮が検知に有効であることを示している．酒井ら [6] は，DbD 攻撃では HTTP レスポンスヘッダ中の X-Powered-By ヘッダに PHP のバージョン情報を高い割合で保有していることが多いという事実を用い，そのヘッダに基づいた検知手法を提案しており，不正スクリプトコードの難読化の影響を受けず軽量な HTTP ヘッダ情報の特徴による悪性通信の検知が可能であることを示している．北野ら [7] は，通信ログから得られるような GET 先ドメイン名や UserAgent が DbD 攻撃の進行に伴って遷移することから，検知ルールによって一定の攻撃検知が行えることを示している．

まとめとして，ペイロード情報を使った攻撃検知では，コードの解析による包括的な情報を特徴とした汎用的な検知が行える一方，解析に要するコストは高いと考えられる．また，ヘッダ情報を使った攻撃検知では，取得の容易な情報によって検知が行える一方，攻撃者の設定依存な情報や静的ルールに基づいた検知手法であるため，攻撃バターンの変化に追随することが困難と考えられる．

3 ファイルタイプ遷移に基づいた検知手法

DbD 攻撃の早急な検知を行うには，組織ネットワーク下で容易に取得できる情報を使う軽量さが求められる一方，攻撃パターンの変動に対して耐性を持つ検知手法が求められる．そこで本研究では，組織ネットワーク下で容易に得られる情報のうち HTTP レスポンスヘッダに含まれる Content-Type 情報を用い，DbD 攻撃の進行に伴ってクライアント PC がフェッチするファイルの種類が変化していくことをファイルタイプ遷移として表現することで，攻撃パターンの変動に対する耐性を提供することを目指した軽量な検知手法を提案する．

3.1 検知手法概要

一般的に組織ネットワーク下にはプロキシサーバのようなノードが存在し，各クライアント PC の簡易的な通信ログが時系列で蓄積されている．例えばプロキシサーバなどに利用される squid[9] では，HTTP レスポンスヘッダの Content-Length フィールドや Content-Type フィールドの情報を記録できる．本手法では，そのような Content-Type フィールド情報が取得できる環境を想定し，そのファイルタイプ情報を用いる．

まず，学習フェーズで既知の悪性通信データと良性通信データからファイルタイプの遷移を時系列データとして抽出し，特徴ベクトルに変換する．それら特徴ベクトルの悪性サンプルと良性サンプルを訓練データとし，機械学習によって分類器を作成する．テストフェーズでは，クライアントの通信ログから特徴ベクトルのサンプルを抽出し，分類器に入力することでその通信が悪性か良性かを判定する．

3.2 指定不審ファイルと指定抽出ファイル

この節では，どのファイルタイプに着目してファイルタイプ遷移のサンプルを抽出するのかを説明する．

図 2 は，一連の通信からサンプルを抽出する例を示している．2.1 節で述べたように，DbD

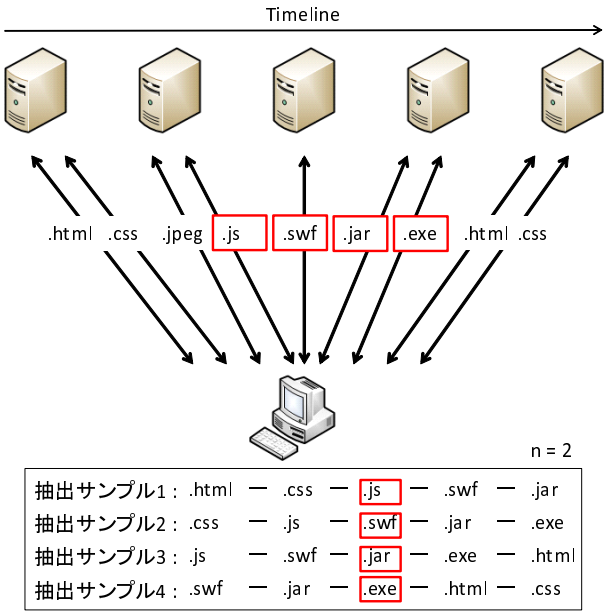

図 2: 特徴ベクトル化対象となる抽出サンプル例

攻撃はその進行に伴ってフェッチするファイルの種類が変化する．第 1 段階では，JavaScript 等の不正スクリプトコードをフェッチする．第 2 段階では，脆弱性を悪用するため，Flash，PDF，JAR ファイル等をフェッチする．第 3 段階では，トロイの木馬やスパイウェアなどの EXE ファイル等をフェッチする．それら DbD 攻撃に関連する図 2 赤枠のようなファイルを，指定不審ファイルと呼ぶこととする．指定不審ファイルがフェッチされた際，時系列的にその指定不審ファイルの前後 n 個のファイルタイプを含めてサンプルとして抽出する．この時，前後のファイルタイプとして抽出対象となるファイルを，指定抽出ファイルと呼ぶこととする．

3.3 特徴ベクトルへの変換

訓練データとテストデータはベクトルである必要があるため，抽出されたサンプルを特徴ベクトルへ変換する方法について述べる．

図 3 に，抽出サンプルの特徴ベクトルへの変換例を示す．サンプル 1 の各指定抽出ファイルは質的変数であり，これを量的変数とするため，各ファイルタイプをダミー変数化する．m 個の指定抽出ファイルが定義されている場合，各ファイルタイプは m 次元のダミー変数となる．指定不審ファイルを中心として前後 n 個を抽出した

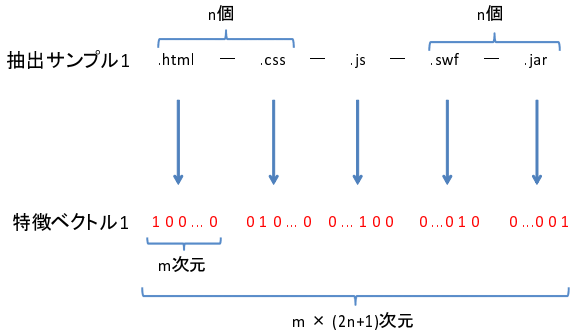

図 3: 特徴ベクトルへの変換例

場合，特徴ベクトルは m × (2n+1) 次元となる．

4 評価

提案手法の正答率を評価するため，悪性通信データと良性通信データを使った SVM による機械学習結果とその考察について述べる．

4.1 概要

評価の概要として，用いる通信データと機械学習アルゴリズム，そして今回の評価で定義する指定不審ファイルと指定抽出ファイルについて述べる．

4.1.1 通信データ

ある URL にアクセスした際発生する一連の HTTP 通信を，セッションと呼ぶこととする．

悪性通信データとして，MWS Dataset[8] の D3M2013，2012 を用いた．D3M ではシードサイトの URL が掲載されているため，それを基に D3M 攻撃通信データを各セッションに分けた．

良性通信データとして，ALEXA による Web サイト上位 100 万 [10] を用いた．これらの Web サイトは必ずしも良性とは限らないが，ここでは良性通信とした．Mozilla FireFox を用いて各 URL にアクセスし，発生する HTTP 通信をそれぞれセッションごとに取得した．

各通信データから取得した全セッション数を表 1 に示す．悪性セッション数は 200 セッション，良性セッション数は 9911 セッションであった．

表 1: 用いる通信データの内訳

	悪性通信データ	良性通信データ
データ元	D3M2013,2012	ALEXA
全セッション数	200	9911
セッション数 (JavaScript)	28	8767
サンプル数 (JavaScript)	101	204351
セッション数 (Flash)	15	3154
サンプル数 (Flash)	25	10488

表 2: 定義する指定不審ファイルと指定抽出ファイル

指定不審ファイル	指定抽出ファイル
1. application/javascript (application/x-javascript) (text/javascript) 2. application/x-shockwave-flash	text/html text/css text/plain text/* application/javascript (application/x-javascript) (text/javascript) application/x-shockwave-flash application/pdf application/java-archive (application/x-java-archive) application/x-msdownload application/octet-stream application/* N/A

4.1.2 機械学習アルゴリズム

機械学習アルゴリズムとして，教師あり学習の中でも比較的精度の高いとされる [11]Support Vector Machine (SVM) を用いた．SVM の実装には R 言語のパッケージである e1071 を用いた．

4.1.3 定義する指定不審ファイルと指定抽出ファイル

3.2 節で説明したように，訓練データやテストデータの抽出は指定不審ファイルと指定抽出ファイルの定義によって決定される．今回の評価で定義した指定不審ファイルと指定抽出ファイルを表 2 に示す．

まず指定不審ファイルの定義は，1. JavaScript のみ　の場合と 2. Flash のみ　の場合の 2 パターンとした．その理由は，良性通信データは一般的な Web サイトのトップページにアクセスした際の通信データであるため，その良性通信データからは JavaScript，Flash ファイル以外で指定不審ファイルになり得る PDF，JAR ファイル等がほとんど取得できていない．そのため，それらのファイルを指定不審ファイルと定義してしまうと，悪性通信データのみにそれら指定不審ファイルを中心としたサンプルが出現してしまい，良性通信データとの公平な比較ができないためである．

また，指定抽出ファイルの定義は，表 2 の通りとする．これは，悪性通信データにおける全 HTTP レスポンスの Content-Type フィールド値で頻度分布を調べた際の上位 10 位のファイルタイプ群と，それ以外の text, application である．type/subtype で subtype を「*」と表記しているものは，指定ファイルで定義されていない MIME Type を表している．N/A は，リダイレクトなどで Content-Type フィールドのない HTTP レスポンスを表している．

なお，指定不審ファイルを中心として前後 n 個の指定抽出ファイルを抽出したものがサンプルであるが，ここでは n=2 とする．つまり，指定不審ファイルがフェッチされた数だけサンプルは存在し，各サンプルは 5 つのファイルタイプ情報を含む．ただし，指定抽出ファイルが存在しない場合は 5 未満になる場合がある．

表 1 に示す通り，JavaScript を含む悪性セッションは 28 セッションあり，そこから合計 101 サンプルを抽出した．また，JavaScript を含む良性セッションは 8,767 セッションあり，そこから合計 204,351 サンプルを抽出した．一方，Flash を含む悪性セッションは 15 セッションあり，そこから合計 25 サンプルを抽出した．また，Flash を含む良性セッションは 3,154 セッションあり，そこから合計 10,488 サンプルを抽出した．

4.2 サンプル単位の評価

抽出されたサンプルそれぞれに対して正答率を評価する．

学習フェーズでは，悪性サンプルと良性サンプルの数を同数にすることとし，悪性サンプル数分の良性サンプルをランダムに選ぶ．今回の

表 3: サンプル単位の正答率

	正答率 [%] (JavaScript)	正答率 [%] (Flash)
最大	67.3	80.0
最小	58.4	10.0
平均	63.6	37.4
標準偏差	3.0	20.7

表 4: セッション単位の正答率 (JavaScript)

		Active			Passive		
		最大	最小	平均	最大	最小	平均
悪性	正答率 [%]	100.0	96.4	98.0	89.3	75.0	82.1
	誤答率 [%]	0.0	3.6	2.0	10.7	25.0	17.9
良性	正答率 [%]	71.4	60.7	65.5	85.7	82.1	82.8
	誤答率 [%]	28.6	39.3	34.5	14.3	17.9	17.2
両方	正答率 [%]	85.7	78.6	81.8	87.5	78.6	82.3
	誤答率 [%]	14.3	21.4	18.2	12.5	21.4	17.7

検証では 10 通りの良性サンプル群のランダムサンプリングを行い，それぞれ別の試行として検証した．

10 回の各試行では交差検定を行う．ここでの交差検定は，サンプルの中から 1 つをテストデータとして選び，それ以外のサンプルを学習させることでテストデータの正誤判定を行う一個抜き交差検定によって評価する．サンプルそれぞれに対して正誤判定結果が出るが，最終的な正答率はその平均とする．

JavaScript，Flash それぞれの場合について正答率の最大，最小，平均，標準偏差を表 3 に示す．JavaScript の正答率平均は Flash よりも高いことがわかり，また 10 回の試行においての正答率のばらつきも小さい．一方，Flash の正答率の標準偏差は比較的大きく，良性サンプルの選び方によるばらつきが大きいことがわかる．

4.3 セッション単位の評価

前節では各サンプルを対象にして正答率を計算したが，実用上は悪性セッションの検知率や良性セッションであるにも関わらず悪性と判断されてしまうセッションの誤検知率が重要である．そのため，各サンプルが属しているセッション単位で正答率を導出する．

セッション単位で考える場合，あるセッションから抽出されたサンプルが (a) 全て悪性，(b) 一部悪性で一部良性，(c) 全て良性の 3 パターンが存在する．この時，(b) のパターンに属するセッションを悪性と判断するか良性と判断するかで正答率が変わってくる．そのため，そのセッションを悪性と判断する判定方法を「Active」，良性とする判定方法を「Passive」とする．つまり Active では，サンプルのうち 1 つでも悪性と判定されれば悪性セッションであると判断するため，積極的に警告アラートを出すような False Negative を軽減するシステムポリシーとなる．一方 Passive では，False Positive を軽減するシステムポリシーとなる。

前節と同様に 10 回の試行を JavaScript，Flash それぞれについて行った．良性セッション数，良性サンプル数は，悪性セッション数，悪性サンプルと同数とする．つまり，JavaScript の場合は悪性良性それぞれ 28 セッション 101 サンプル，Flash の場合は 15 セッション 25 サンプルとなる．

JavaScript，Flash それぞれの場合について正答率の最大，最小，平均を表 3 に示す．悪性セッションの正答率は，JavaScript の場合 Active で平均 98.0%，Passive で平均 82.1%であり，Flash の場合 Active で平均 61.8%であり，Passive で平均 44.3%である．そのため，悪性セッションの検知率を向上させたい場合は Active による判定を用いると良いことがわかる．一方，良性セッションの正答率は，JavaScript の場合は Active で平均 65.5%，Passive で平均 82.8%であり，Flash の場合は Active で平均 34.4%であり，Passive で平均 45.0%である．そのため，良性セッションの誤検知率を低減させたい場合は Passive による判定を用いると良いことがわかる．また，悪性と良性の両方をあわせた場合の正答率は，Active において JavaScript は平均 81.8%，Flash は 48.1%であり，Passive において JavaScript は平均 82.8%，Flash は平均 44.6%である．サンプル単位の評価の場合と同様，Flash の場合の誤判定が多いことがわかる．特に，悪性セッションが検知できないことは問題となるため，以下の節では悪性セッションの誤判定原因について考察する．

表 5: セッション単位の正答率 (Flash)

		Active			Passive		
		最大	最小	平均	最大	最小	平均
悪性	正答率 [%]	100.0	26.7	61.8	86.7	26.7	44.3
	誤答率 [%]	0.0	73.3	38.2	13.3	73.3	55.7
良性	正答率 [%]	66.7	0.0	34.4	80.0	6.7	45.0
	誤答率 [%]	33.3	100.0	65.6	20.0	93.3	55.0
両方	正答率 [%]	83.3	13.3	48.1	83.3	16.7	44.6
	誤答率 [%]	16.7	86.7	51.9	16.7	83.3	55.4

表 6: 誤判定された代表的な悪性セッションとそのサンプル

	セッション	代表的なサンプル
JavaScript	(b)-1	.js .js .js .js .js
		.js .js .js .pdf .js
		.js .pdf .js .js .css
	(b)-2	.html .pdf .js .js .html
	(c)-3	.html .html .js — —
Flash	(c)-4	.html .html .swf .swf .swf
	(c)-5	.swf .js .swf .html .swf
		.swf .html .swf N/A .jar
	(c)-6	.html .html .swf .html —
	(c)-7	.html .pdf .swf .jar .jar

4.4 悪性セッションの誤判定原因

JavaScript,Flash それぞれで正答率が一番悪かった試行のうち，誤判定されたサンプルが属する悪性セッションについて考察する．

表 6 に，誤判定された代表的な悪性セッションとそれに属するサンプルを示す．(b) は「抽出されたサンプルの一部が良性」，(c) は「全てが良性」と判定された悪性セッションを示し，その後に続く数字は悪性セッションの識別番号を示す．「—」はセッション中で HTTP レスポンスが存在せず，Content-Type フィールドが取得できなかったことを示す．「N/A」は HTTP レスポンスは存在したが，Content-Type フィールドが存在せず取得できなかったことを示す．

以下では，表 6 の JavaScript，Flash について誤判定原因の考察を述べる．

4.4.1 JavaScript の場合

セッション (b)-1 のように，同一ファイルタイプの割合が多いと誤判定を起こしやすい．これは，良性データから JavaScript の割合が大きいサンプルが抽出されており，それによって誤判定が起きている．良性データは一般の Web サイトのトップページから取得されたものであるため，トップページの HTML ファイルが多くの JavaScript ファイルを呼び出し，それら JavaScript ファイルが時系列的に近傍でフェッチされていると考えられる．そのため，良性データには JavaScript の割合の大きいサンプルが抽出されていると考えられる．同様に，HTML ファイルも良性通信においてフェッチされる割合が大きいため，(b)-2 や (c)-3 も良性通信であると判断されて，誤判定を起こす．

4.4.2 Flash の場合

JavaScript の場合と同様，Flash ファイルの近傍に Flash ファイルや HTML ファイルが存在する良性データが多いことが原因だと考えられる．また，Flash の場合はサンプル数が少なく，(c)-7 のように悪性通信として特徴的であるサンプルでも誤判定を起こしてしまう．サンプル数が少ないことで正答率が良性データの変化に依存しやすくなり，正答率の分散が大きくなってしまうことにも繋がる．

5 おわりに

本稿では，キャンパスネットワーク等の組織ネットワーク下で容易に得られる情報のうち，HTTP レスポンスの Content-Type フィールド情報を用いることで，クライアント PC が通信するファイルタイプの遷移を特徴ベクトルで表現し，機械学習によって悪性通信と良性通信の区別をする手法を提案，評価した．

評価では，通信データから抽出されたサンプルによって JavaScript ファイルと Flash ファイルを対象とした分類器を作成し，サンプルに対する悪性良性判定を行った．その結果，サンプル単位の正答率よりも実用上重要なセッション単位で正答率を導出することで，正答率が大幅に改善可能なことが明らかになった．

また，Active と Passive を比較すると，悪性セッションの検知率向上に対しては Active を，良

性セッションの誤検知率低減に対してはPassiveを用いることが良いことがわかった．悪性良性の両方を合わせた正答率としては，Passiveを用いたJavascriptの場合，平均82.3%となり，十分な正答率が提供できることが明らかになった．しかし、Flashの場合はActiveを用いて平均48.1%と，改善の余地を残した．今後の課題は以下である．

- 通信データの改善

  Flashの場合のように，良性サンプルによって正答率が大きく変動してしまうため，良性通信データの質の向上が必要である．また，悪性通信データにおいては良性通信データに比べサンプル数が少ないため，量の向上が必要である．

- JavaScript, Flashファイル以外の評価

  JavaScriptファイルとFlashファイルに対して評価を行ったが，PDFファイルやJARファイルなど悪性通信に割合の多いファイルタイプに対しても評価を行う必要がある．そのため，良性サンプルとして用いる良性通信データの取得の方法を考える必要がある．

- 他の機械学習アルゴリズムの適用

  機械学習アルゴリズムとしてSVMを用いたが，他の機械学習アルゴリズムを用いた場合の精度と比較し，より精度の良い手法を検証する必要がある．

謝辞

本研究はJSPS科研費 26730066の助成を受けたものです．

参考文献

[1] V. L. Le, et al., “Anatomy of Drive-by Download Attack,” Proc. Australasian Information Security Conference 2013 (AISC’13), vol. 138, pp. 49–58, Feb. 2013.

[2] J. Chang, et al., “Analyzing and defending against web-based malware,” ACM Computing Surveys(CSUR), vol. 45, no. 4, pp. 49:1–49:35, Aug. 2013.

[3] C. Grier, et al., “Manufacturing Compromise: The Emergence of Exploit-as-a-Service,” Proc. ACM Conference on Computer and Communications Security, pp. 821–832, Oct. 2012.

[4] M. Cova, et al., “Detection and analysis of drive-by-download attacks and malicious JavaScript code,” Proc. International Conference on World Wide Web, pp. 281–290, Apr. 2010.

[5] 寺田剛陽, 他, “検知を目指した不正リダイレクトの分析,” 情報処理学会シンポジウム論文集, vol. 2010, no. 9, pp. 765–770, 2010年10月.

[6] 酒井裕亮, 佐々木良一, “Drive By Download攻撃に対するHTTPヘッダ情報に基づく検知手法の提案,” 情報処理学会研究報告. マルチメディア通信と分散処理研究会, vol. 2013, no. 29, pp. 1–6, 2013年3月.

[7] 北野美紗, 他, “Drive-by-Download 攻撃における通信の定性的特徴とその遷移を捉えた検知手法,” コンピュータセキュリティシンポジウム2013論文集 CSS2013, vol. 2013, no. 4, pp. 595–602, 2013年10月.

[8] 神薗雅紀, 他, “マルウェア対策のための研究用データセット ～MWS 2013 Datasets～,” コンピュータセキュリティシンポジウム2013論文集 CSS2013, vol. 2013, no. 4, pp. 1–8, 2013年10月.

[9] squid-cache.org,
`http://www.squid-cache.org`

[10] Alexa, The top 100 sites on the web,
`http://www.alexa.com/topsites/global`

[11] C. C. Chang, and C. J. Lin, “LIBSVM : A Library for Support Vector Machines,” ACM Transactions on Intelligent Systems and Technology (TIST), vol. 2, no. 3, pp. 27:1–27:27, Apr. 2011.