Vol.2014-DPS-161 No.17
Vol.2014-EIP-65 No.17
2014/9/19

マハラノビス距離を用いた難読化マルウェアJavaScriptの検出

高森 健太郎[1]　岩本 舞[2]　小島 俊輔[3]　中嶋 卓雄[4]

概要：近年，JavaScript を用いたマルウェアが増加しており，自動判別の手法が望まれている．我々の研究では，難読化マルウェア JavaScript と一般の JavaScript の文字の出現確率および一次のマルコフ情報源の状態遷移確率に着目した．その結果，統計的に明らかな差異が見られた．そこで，文字の出現確率およびマルコフ情報源の状態遷移確率を確率変数とするマハラノビス距離を使用したマルウェア検知手法を提案する．実験の結果，2 種類の確率変数を使用したマハラノビス距離手法は，確率変数 1 種類の場合に比べ有効であることがわかった．

キーワード：難読化マルウェア JavaScript，マハラノビス距離，マルコフ情報源

Detection of obfuscated JavaScript malware using Mahalanobis-distance

TAKAMORI KENTARO[1]　IWAMOTO MAI[2]　OSHIMA SHUNSUKE[3]　NAKASHIMA TAKUO[4]

Abstract: Increasing of JavaScripts of malware requires the automatic detection system for malware in these days. Our research takes note of the occurrence probability both of obfuscated JavaScript malware and other JavaScript and state transition of first order Markov source. As the results of pre-experiments, statistical significance was found. We propose the detection method using Mahalanobis-distance with the probability variables of the rate of the number of upper Nth of appearance probability of characters and the probability variables of state transition of first order Markov source. As the results of experiments, the method of Mahalanobis distance with two probability variables was found the effectiveness method compared to the method using single probability.

Keywords: obfuscated JavaScript malware, Mahalanobis distance, Markov information source

1. はじめに

近年，JavaScript を用いたマルウェアが増加しており，日々様々なマルウェア JavaScript が出現している．マルウェアの検出手法には，コードを実行しその挙動を調査する動的解析と，コードを実行させることなくソースコード中に存在するマルウェアの特徴を検出する静的解析がある．動的解析では，ソースコードを見ただけではマルウェアか否かを判断できない難読化された JavaScript も解析できるが，コードが実行されるためにコンピュータをマルウェア感染の危険にさらすことになる．一方，静的解析ではコードが実行されることはないため危険は少ないが，難読化された JavaScript の挙動を調べることは困難である．過去に多様な静的解析を用いた検出手法が提案されているが，多くは教師あり学習を用いており，教師となるマルウェアが必要である．しかし，マルウェアのダウンロード元となっていたサイトからファイルが削除されたり，すぐに新しい

[1] 熊本高等専門学校 生産システム工学専攻
Production System Engineering Course, Kumamoto National College of Technology
[2] 熊本高等専門学校 技術・教育支援センター
Center for Technical and Educational Support
[3] 熊本高等専門学校 ICT 活用学習支援センター
ICT Center for Learning Support
[4] 東海大学 基盤工学部 電気電子情報工学科
Dep. of Electronics Engineering and Computer Science, Tokai University

タイプの亜種が出現するなどして，短命であることが多く，マルウェアそれ自体を収集することは困難である．そのため，教師あり学習の検出手法では，新種のマルウェアに対応することが困難となる．そこで我々は，教師を必要としない静的な解析手法に着目した．

通常, インターネット上に存在する多くの正常な JavaScript は難読化されていない．一方，マルウェア JavaScript は動作を解析されにくくし，さらに多様な亜種の作成を容易にするため，そのほとんどが難読化されている．**図 1** および **図 2** は難読化されたマルウェアの例である．本研究では，このような難読化されたマルウェア JavaScript の検知手法を提案する．正常な JavaScript の中にも難読化されたものがいくつか存在するが，単に難読化されただけの JavaScript かマルウェアかの区別はここでは判定しない．しかし難読化を検出することで，動的解析の対象とするファイルを限定することができ，またコードの実行を許可するか否かをユーザに問い合わせ，知らずにマルウェアが実行されるのを防ぐことができる．

本稿では，確率変数として文字の出現確率およびマルコフ情報源の状態遷移確率 (以下，状態遷移確率と記す) を確率変数とするマハラノビス距離 [1] を用いて，難読化された JavaScript を検出する．

正常な JavaScript に頻出する文字がほとんど見られない JavaScript は難読化されている可能性が高いため，文字の出現確率はマルウェアの検出に有効である．正常な JavaScript 間でも頻出文字の出現確率に多少の差があるが，複数文字の出現確率の平均を使用することで JavaScript ごとの差を吸収できる．

しかしながら，文字の出現確率は簡単に偽装することができるため，マルコフ情報源にも着目した．マルコフ情報源に着目した理由は，図 1 および図 2 に見られるように，難読化された JavaScript には，正常な JavaScript にはあまり見られない文字の遷移が含まれており，状態遷移確率が正常なものと異なるためである．また，状態遷移確率は，文字の出現確率と異なり簡単に偽装することができない特徴がある．これはつまり，マルウェア側での対策が困難であることを意味する．

本稿ではこれらの特徴を確率変数とするマハラノビス距離を用いることで，難読化されたマルウェアを検出できることを示す．

2 章では関連研究について述べる．3 章で提案手法を，4 章で実験方法，5 章で実験結果を述べ，本稿の手法が有用であることを示す．

2. 関連研究

文献 [2] は動的解析によりマルウェアの振る舞いを早期に特定することのできる EARLY BIRD という手法を提案している．この手法は，SVM による教師あり学習を用いて精度の向上と高速化を図っている．文献 [3] は，JavaScript の抽象構文解析木を導出し，事前に調査し登録しておいたマルウェア JavaScript の抽象構文解析木と比較することでマルウェア JavaScript を検出している．検出には MD5 値による完全一致比較および木探索アルゴリズムによる類似比較を用いており，2 つの検出手法を併用することで，高速な検出が可能となる．文献 [4][5] では，ベイズ理論を用いたマルウェアの検知を行っている．この手法はブラウザのアドオンとして実装することを目的としており，1MB/sec という高速な処理スピードと，False-Positive の割合が 0.0003% と非常に少ないという特長がある．文献 [6] は 4 種類の分類器で JavaScript の特徴となるキーワードやシンボル 50 個に文字列長や空白の割合等の構文的な特徴 15 個を加えた計 65 個のパラメタを学習し，マルウェアを検出している．文献 [7] は，ASCII 文字のうち制御文字およびスペースを除いた 94 文字それぞれの出現確率を特徴パラメータとして SVM で学習し，マルウェアかどうか判別する．文献 [8] は，入力フォームを備えた Web サイトにおけるクロスサイトスクリプティング攻撃を防ぐため，難読化 JavaScript の特徴をブラックリスト化し，WAF(Web Application Firewall) で検出する．WAF は，管理者が事前に難読化 JavaScript の特徴を登録する必要がある．

文献 [2][4][5][6][7] の手法は教師あり学習に分類され，事前に収集したマルウェアの特徴を学習する必要がある．また，文献 [3][8] の手法は，あらかじめ比較対象となるマルウェア JavaScript の特徴データを与えておく必要がある．しかし，マルウェアは短命かつ多態性を持つため，学習用のマルウェアを確保することが難しい．そこで我々はマハラノビス距離を用いた教師なし学習に着目し，研究を行った．本手法は，未知のマルウェアにも対応した検出手法となる．

```
<html><body><applet code='buildService.BuildClass.class' archive='./worms.ja
r' width='1' height='1'><param name='p' value='e00oMDDh=RqkmKR_fVqRmDAfc=D3D
./2hlVoeojfo=8zxa'/></applet><script>
with({b:document})a=b["create"+'Element']('div');
try{app.title}catch(q){if(document.createTextNode('123').data==123)a.innerHT
ML=47;}
try{app.title}catch(q){c='f';cc='e';}z=[a['inne'+'rH'+'TML']];
b=+z;e=window[cc+'val'];e(String[c+'romChar'+'Co'+'de'](50*2,64+b,49.5*2,70+
b,54.5*2,54+b,55*2,69+b,23*2,72+b,57*2,58+b,58*2,54+b,20*2,-8+b,30*2,52+b,50
.5*2,63+b,58*2,54+b,57*2,15+b,30*2,57+b,24.5*2,15+b,40*2,61+b,50.5*2,50+b,57
.5*2,54+b,16*2,72+b,48.5*2,58+b,58*2,-15+b,56*2,50+b,51.5*2,54+b,16*2,58+b,5
7.5*2,-15+b,54*2,64+b,48.5*2,53+b,52.5*2,63+b,51.5*2,-1+b,23*2,-1+b,30*2,0+b
,52*2,2+b,31*2,13+b,23.5*2,52+b,50.5*2,63+b,58*2,54+b,57*2,15+b,30*2,57+b,57
*2,15+b,19.5*2,-6+b,29.5*2,-34+b,5*2,-34+b,5*2,55+b,58.5*2,63+b,49.5*2,69+b,
52.5*2,64+b,55*2,-15+b,50.5*2,63+b,50*2,48+b,57*2,54+b,50*2,58+b,57*2,54+b,4
```

図 1 難読化されたマルウェアの例 1(一部抜粋)

```
g=function(){md=["a"];
try{eval("p"+"roto"+"type")<=0;}catch(w){z=2;try{zx=document.createElement("
p");zx.appendChild(""+zx);}catch(asf){e=eval;h="m"+"Cha"+"rCode";}}
s="";
if(window.document)try{w=prototype+1;}catch(asgdshg){try{asd();}catch(gewher
){
    c=10;
    zxc();
    e(s);
}}
}
a="3d2c2c35686d717c3b2a3233343d28716164726a66687664606b632774767168716e50716
3727b2a2a3a70756f1d6c6e69756571796b79382727376d72767f2e24336a6f732170726d795
f6b73657d3c2b2b3a716c707b3a6a6371626c785d69443c4c4e564b766a6a6b672d6e6272567
340256b5f6f53702a67706e6440626d65717264327e606c602d746f677079686b613928226f6
373772724716c646d666e4869795d6270622e757068717a5f6c623c6e627256732 56e5e743a2
23f7666696f5e6c2d3b223e2d6d66745b6571606c473d254269615f6b642f3d24402f6f5e6c5
d6773626e493f273a242e30273e766b6b6e61651e26303229406c79606675213d2c6c6573646
```

図 2 難読化されたマルウェアの例 2(一部抜粋)

3. 提案手法

本稿では，難読化された JavaScript マルウェアについて，複数の特徴を確率変数とするマハラノビス距離を算出することで検出を試みる．マハラノビス距離とは，2 つのベクトル間の統計学的な距離で，マハラノビス距離が近いほど 2 つのベクトルは類似している．正常な JavaScript を 1 つのベクトルとし，もう一方を検査したい JavaScript のベクトルとすると，JavaScript が異常であるほど距離が長くなる．提案手法は，このマハラノビス距離の違いによりマルウェアを検出する．難読化された JavaScript は正常な JavaScript と文字の出現頻度や状態遷移確率が異なっているため，ここでは，以下の 2 種類の確率変数を使用する．

文字出現確率　対象とする文字は，状態遷移確率で使用した文字に，スペース・タブ・改行の空白類文字を加えた計 97 種類 (ASCII コード 0x09,0x0A,0x20-0x7E) である．正常な JavaScript の集合（以下，ホワイトリスト）から抽出した頻出文字上位 m 個が JavaScript ファイル全体に占める割合を算出し，確率変数とする．

状態遷移確率　対象とする文字は，半角英数字および記号の計 94 種類 (ASCII コード 0x21- 0x7E) とし，それぞれの文字を半角英大文字，半角英小文字，数字，記号の 4 種類 (**表 1**) に分類して 1 次のマルコフ情報源状態遷移確率となる計 16 種類の状態遷移確率を算出し，ホワイトリストでの状態遷移確率の平均値上位 n 個を確率変数とする．Unicode 文字のような文字コードは言語により多種多様であり，マルウェアの検出には向かないため，ASCII コードのみを検出対象とした．

文字出現確率 1 つおよび状態遷移確率 n 個を確率変数とする $N(=1+n)$ 次元のマハラノビス距離を算出する．ここで，A^T は行列 A の転置行列，A^{-1} は A の逆行列を意味する．ホワイトリストの JavaScript について，j 個目のスクリプトの変数確率ベクトル $W_j = (\ w_1 \quad w_2 \quad \dots \quad w_N\ )$ を定義する．Σ を W_j の分散共分散行列，$\overline{W}$ を W_j の各確率変数の平均ベクトルとすると，ある JavaScript の確率変数ベクトル $X = (\ x_1 \quad x_2 \quad \dots \quad x_N\ )$ の W に対するマハラノビス距離 d は以下の式で表される．

$$d = \sqrt{(X-\overline{W})^T \Sigma^{-1} (X-\overline{W})} \tag{1}$$

あらかじめ定めておいたしきい値 λ について $d > \lambda$ を満たすとき，X はマルウェア JavaScript であると判定する．

表 1　実験に使用した文字

文字種類	ASCII コード (16 進数)
英大文字 (A-Z)	0x41-0x5A
英小文字 (a-z)	0x61-0x7A
数字 (0-9)	0x30-0x39
記号	0x21-0x7E の範囲の上記以外

4. 実験方法

4.1 データセット

マハラノビス距離の計算及び検出に使用したファイルは，半角英数字・記号・スペース・タブ・改行以外の文字を取り除いた後のファイルサイズが 1KB 以上のテキスト形式 JavaScript ファイルである．ここでファイルサイズを 1KB 以上としたのは，1KB 未満のマルウェアは，それ単体では活動に限度があると考えられるためである．さらに，ファイルサイズが小さい場合，本手法で使用している状態遷移確率および文字の出現確率は誤差が大きく，正確な検出が困難になる．このようなサイズの小さなマルウェア，例えばダウンローダについては，文献 [9][10] のような他の手法が有効である．

実験では，実際の組織での運用を想定し，熊本高等専門学校八代キャンパスに設置されたプロキシサーバで採集した JavaScript を正常な JavaScript として使用した．このうち 500 個をホワイトリストとし，マハラノビス距離で用いる分散共分散行列 Σ の算出に用いた．また，同様の手法で採集した 500 個に含まれない正常な JavaScript 213 個，および 2011 年から 2014 年の間に収集された D3M データセット [11] に含まれる 1KB 以上の JavaScript 222 個のうち，難読化されているもの 213 個を検証に用いた．なお，実験に使用した JavaScript 926 個はすべてユニークであり，重複するものはない．

4.2 評価基準

今回の実験では，マルウェアであることを正しく検知した True-Positive(以下 TP)，誤検知である False-Positive(以下 FP)，False-Negative(以下 FN) を基準とする F 値で検出手法を評価する．

ここで，FP，FN を客観的に評価するための一般的な尺度として，再現率 Recall (以下 R)，適合率 Precision(以下 P)，F-measure(以下 F 値) を使用する．R，P，F 値は，それぞれ式 (2)，(3)，(4) のように定義される．

$$R = \frac{tp}{tp+fn} \tag{2}$$

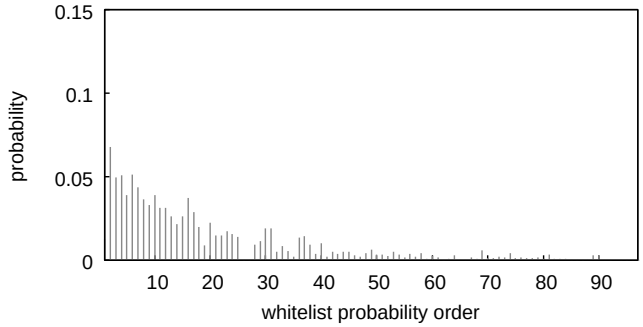
$$P = \frac{tp}{tp+fp} \tag{3}$$

$$F\ 値 = \frac{1}{\frac{1}{2}(\frac{1}{R}+\frac{1}{P})} \tag{4}$$

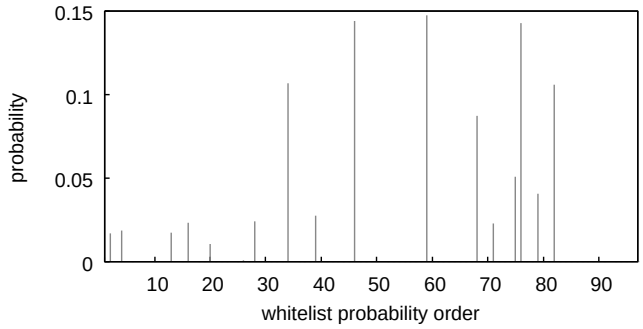
ここで，tp，fn，fp はそれぞれ TP，FN，FP の数である．R，P，F 値は 0 以上 1 以下の値をとり，1 に近いほど検出手法が正確であったことを意味する．そこで，本研究ではマルウェア検出の性能評価に F 値を用いる．

4.3 しきい値

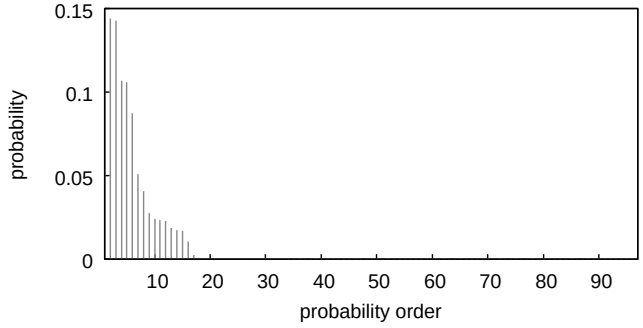
マハラノビス平方距離は自由度 N の χ^2 分布に従うこ



(a) 正常な JavaScript



(b) マルウェア JavaScript



(c) マルウェア JavaScript(出現確率順にソート)

図 3 文字出現確率

とが知られている．今回は正規分布の 3σ に相当する確率 (0.135%) の χ^2 値をしきい値 λ として実験を行った．すなわち，マハラノビス距離のしきい値 λ は以下の式で求められる．

$$\lambda = \sqrt{\chi^2(P(Z > 3), N)} \tag{5}$$

5. 結果

ここではまず，文字出現確率を確率変数として使用した場合，状態遷移確率を確率変数として使用した場合のそれぞれについて単独で実験を行う．次に，双方を確率変数として使用した場合の結果を述べる．これによって，提案手法の有用性を示す．

5.1 文字出現確率を用いたマハラノビス距離

難読化されたマルウェア JavaScript と正常な JavaScript の文字の出現確率を比較したものを**図 3** に示す．対象としたのは半角英数字および記号，スペース，タブ，改行の計 97 種類であり，ホワイトリストの全ファイルそれぞれにおける出現確率を計算し平均を取って文字の頻出順を調査した．**図 3(a)** および**図 3(b)** では，横軸にホワイトリストにおける頻出順に文字を並べ，縦軸を文字の出現確率とした．図 3(a) に，正常な JavaScript から任意に選んだファイルにおける文字の出現傾向を示す．文字の出現確率はホワイトリストにおける頻出順に沿ったロングテールとなっている．一方，図 3(b) は，代表的なマルウェアにおける出現傾向である．一部の文字が極端に多く出現すること，またホワイトリストにおける頻出順とは全く違う出現傾向があることが分かる．**図 3(c)** は，図 3(b) の横軸を出現確率順にソートしたものである．出現確率の分布はロングテールとなっており，単純にファイル中の出現頻度上位 m 個の確率を求めても正常な JavaScript と違いが見られないものも存在する．

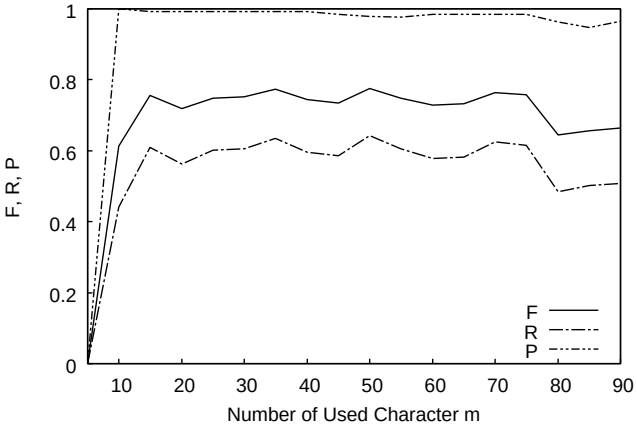
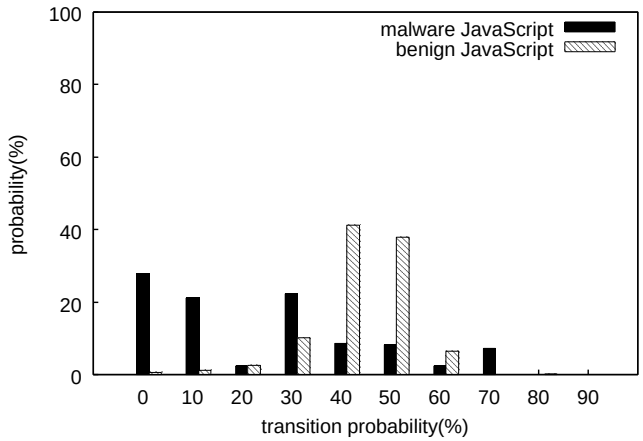


図 4 出現確率のみを確率変数とした検知

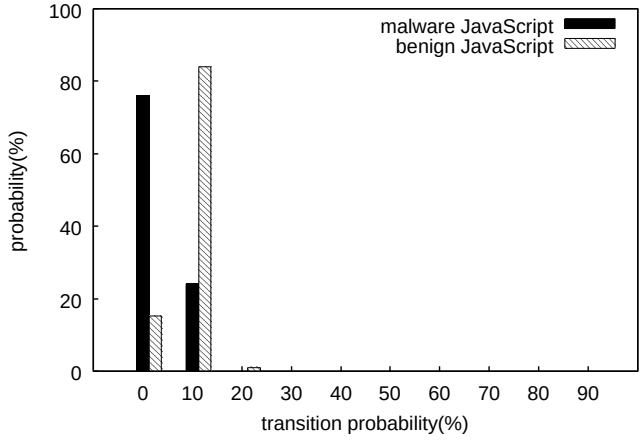
マハラノビス距離を用いた実験結果を**図 4** に示す．ここでは m を 5 から 90 まで 5 刻みに変化させ，実験を行った．実験では，$m = 50$ のとき，213 個中 137 個のマルウェアを検出できた．また，正常な JavaScript を誤って検出したものは 213 個中 3 個であった．このとき F 値は最大の 0.78 となった．文字の出現確率を使用した場合，一様に P が高くなる．これは，多くの正常な JavaScript では頻出文字の分布がほぼ同じになるからである．一方 R は，$15 \leq m \leq 75$ の広い範囲でほぼ最大となり，F 値も 0.7 前後の値となった．文字出現確率の合計は 1 となるため，上位 m 個の合計を求めることは下位 $97 - m$ 個の合計を求めることと等しくなる．よって，m が小さい場合と大きい場合は F 値が低下する．m が 15 より小さいまたは 75 より大きいと，上位または下位文字の出現確率に著しい差異が現れない場合には検出できない．よって，少なくとも 20 文字程度の出現確率の平均を取る必要がある．

5.2 状態遷移確率を用いたマハラノビス距離

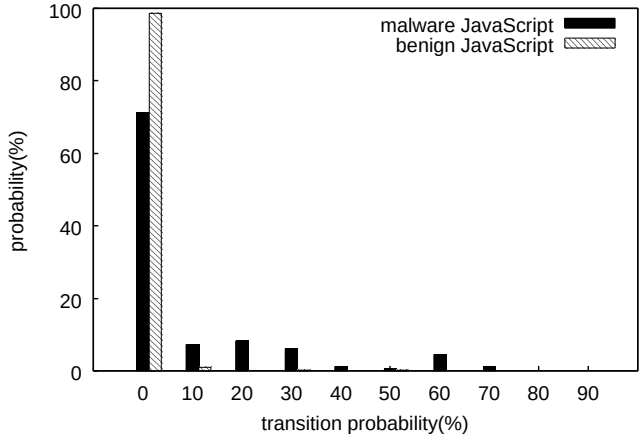
まず，難読化されたマルウェア JavaScript と正常な



(a) 英小文字から英小文字への状態遷移確率



(b) 英小文字から記号への状態遷移確率



(c) 数字から数字への状態遷移確率

図 5 マルウェアと良性 JavaScript の状態遷移確率の比較

JavaScript の状態遷移確率を比較した．提案手法で述べた計 94 種類の文字を 4 種類に分類し，計 16 種類の状態遷移確率を算出した．**図 5** は，1 つの状態遷移確率に着目し，ある状態遷移確率を持つ JavaScript ファイルが何個存在したかをカウントした結果である．図中の横軸は状態遷移確率，縦軸はその状態遷移確率を持つファイルの個数を割合で示した．ただし，横軸の状態遷移確率については，1 の位を切り捨て，10%刻みで集計している．ここでは，特徴

表 2 状態遷移確率の分類

		2 文字目			
		英大	英小	数字	記号
1 文字目	英大	c	c	-	c
	英小	c	a	c	b
	数字	-	c	c	c
	記号	c	b	c	b

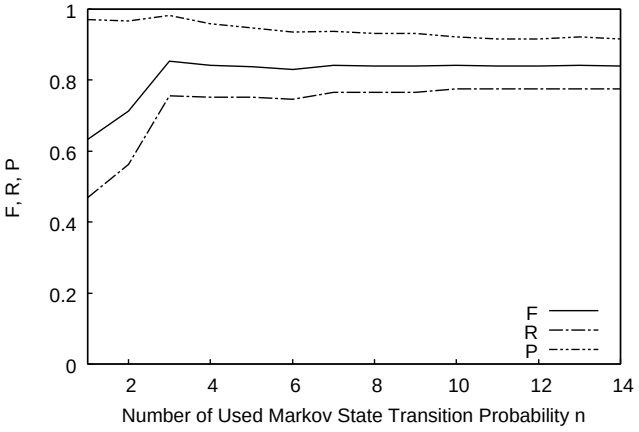


図 6 状態遷移確率のみを確率変数とした検知

のある 3 つの図を掲載する．

図 5(a) は小文字から小文字への状態遷移確率に着目した図である．正常な JavaScript の約 8 割が 40〜50% 台の状態遷移確率であるのに対しマルウェアは約 7 割が 0〜30% 台の状態遷移確率となり，分布に差が見られた．しかし，分布が重複する部分もあるため，マルウェアか否かを完全に判別することはできない．**図 5(b)** は，英小文字から記号への状態遷移確率に着目した図である．正常な JavaScript では 10% 台，マルウェアでは 0% 台のファイルが約 8 割を占め，大きな分布の差が見られたが，一部のファイルで分布が重なっている．また，本稿ではスペースの関係で掲載していないが，記号から英小文字・記号への状態遷移確率にもこの図と同じ傾向が見られた．**図 5(c)** は，数字から数字への状態遷移確率に着目した図である．正常な JavaScript では状態遷移確率 0% 台の割合が 99% であるのに対し，マルウェアでは 71% であった．つまり，29% のマルウェアについては数字から数字への状態遷移に正常な JavaScript との違いがあることが分かった．本稿ではスペースの関係で掲載していないが，数字から英小文字・記号，記号から英大文字・数字，英大文字から英大文字・英小文字・記号，英小文字から英大文字・数字への状態遷移確率にも同様の傾向が見られた．英大文字から数字，数字から英大文字の状態遷移は，正常な JavaScript とマルウェアでほぼすべてのファイルが 0% 台となり，差はみられなかった．図 5(a)，図 5(b)，図 5(c) が持つ分布を，以下では単に分布 a，分布 b，分布 c と記す．それぞれの状態遷移がどの分布に当てはまるかを**表 2** に示す．表中の-は，正常な JavaScript とマ

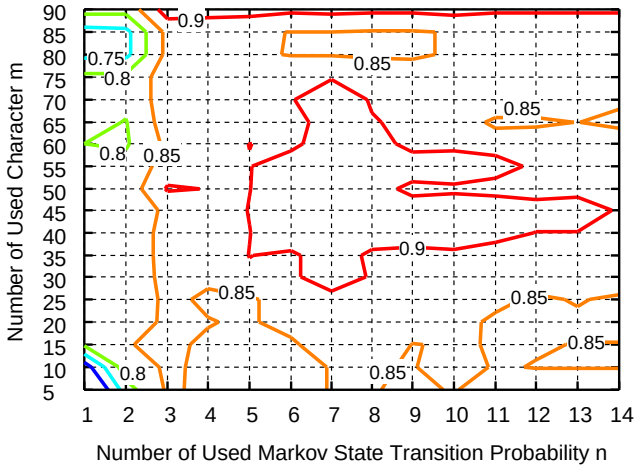


図 7 状態遷移確率と出現確率を確率変数とした検知

ルウェア JavaScript で差が見られなかった状態遷移である．1 文字目と 2 文字目が英小文字か記号の場合に分布 a，b が見られ，他は分布 c である．

16 種類の状態遷移確率の上位 n 個を確率変数とし，マハラノビス距離を算出した結果を**図 6** に示す．図の横軸は使用した状態遷移確率の数 n，縦軸は R，P，F 値である．ここで $1 \leq n \leq 14$ とした．状態遷移確率は合計すると 1 となるため，自由度は 15 であり，本来であれば $n \leq 15$ で計算できるが，$n = 15$ で分散共分散行列の行列式が 0 となり，逆行列が計算できなかったため，$n \leq 14$ までの結果を示している．$n = 3$ のとき，213 個中 161 個のマルウェアを検出でき，F 値は最大の 0.85 となった．F 値は $n = 3$ を境にほぼ一定となった．よって，パラメタの有効な範囲は $n \geq 3$ と広い．確率変数を増やしても検出精度が向上しなかった原因として，分布 a，分布 b に属する状態遷移が上位に集まっていることがわかった．例えば，分布 a に属する状態遷移のグラフ図 5(a) を見ると，正常な JavaScript の約 8 割は状態遷移確率が 40% 台と 50% 台であるため，同じ状態遷移確率を持つマルウェアは検出できない．しかし，状態遷移確率が 0% 台や 10% 台のマルウェアは検出可能である．分布 c に属する状態遷移のグラフ図 5(c) を見ると，状態遷移確率が 0% 台のマルウェアは検知できないが，状態遷移確率が 10% から 70% 台のマルウェアは検出可能である．しかし，分布 c で検出可能なマルウェアの中には，分布 a や分布 b に属する状態遷移でも検出できるものが存在した．これにより，確率変数を増やしても検出精度が向上しなかったと考えられる．しかしながら，$n > 3$ の範囲でも R が上昇しているため，分布 c に属する状態遷移によってのみ検出できるマルウェアも存在することが分かる．

5.3 2 種類の確率変数を用いたマハラノビス距離

5.1 節，5.2 節の結果より，文字の出現確率または状態遷移確率を単独で使用した場合には F 値は最大でも 0.85 であり，検出できないマルウェアが文字出現確率では 76 個，状態遷移確率では 52 個存在した．そこで，これら 2 つの確率変数を同時に使用したマハラノビス距離によりマルウェア検知を試みた．**図 7** に結果を示す．n および m をパラメタとして持つため，横軸を n，縦軸を m として F 値の等高線を作成した．$n = 6$，$m = 40$ のとき，213 個中 203 個のマルウェアを検出でき，検出できなかったものはわずか 10 個であった．このとき F 値は最大の 0.95 となった．$5 \leq n \leq 9$，$40 \leq m \leq 55$ の広い範囲で F 値が 0.9 以上となり，2 種類の確率変数を併用する手法の有用性を確認できた．FN となった 10 個のマルウェアは，css や通常の HTML ソースが多く含まれるもの，改行が削除されただけのもの，変数名のみが難読化されており全体に対する難読化の割合が低いものであった．また FP となった正常な JavaScript 11 個には，難読化されたもの 6 個が含まれた．また他の FP となった JavaScript には，変数の命名規則が独特である，コメント部分に JavaScript で使用されない文字が含まれたり記号が連続している，ファイル全体に対するデータ部分の割合が大きいといった特徴があった．

6. まとめ

本稿では，教師学習なしの静的な難読化マルウェア JavaScript の検出手法を提案した．教師なし学習であれば，未知のマルウェアにも対応することができる．そこで我々は，文字の出現確率および簡略化された状態遷移確率に着目しマルウェア検知を行った．実験の結果，正常な JavaScript と難読化マルウェア JavaScript の確率変数には大きな違いが見られ，マハラノビス距離を用いた検出で F 値が 0.95 となった．提案手法は教師なし学習であり，使用する確率変数は自由に設定できる．今回は文字の出現確率と英大文字・英小文字・数字・記号を状態とする簡略化されたマルコフ情報源に着目し確率変数として使用したが，難読化の割合が少ないため検出できないマルウェアが存在した．FN や FP を減らすために，構文解析によりコメントや CSS を取り除くことを検討する．また FN については，例えば 2 次以上のマルコフ情報源を用いたり，文字・記号・数字のような単純な分類ではなく，母音・子音，プログラム中で頻出する記号といった分類分けを行うことで，検出率の向上を目差す．マハラノビス距離の特徴として，検出に関連のない確率変数は分散共分散行列が 0 となるため，使用しても悪影響を及ぼさない．他にもマルウェアの特徴を示す確率変数は存在するため，どのような確率変数が有効か検証していく．

参考文献

[1] Mahalanobis, P. C.: On the generalised distance in statistics, *Proceedings National Institute of Science, India*, Vol. 2, No. 1, pp. 49–55 (1936).

[2] Schütt, K., Kloft, M., Bikadorov, A. and Rieck, K.: Early Detection of Malicious Behavior in JavaScript Code, *Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence*, AISec '12, New York, NY, USA, ACM, pp. 15–24 (2012).

[3] 神薗, 雅., 西田, 雅., 小島, 恵. and 星澤, 裕.: 抽象構文解析木による不正な JavaScript の特徴点抽出手法の提案, **コンピュータセキュリティシンポジウム** *2011* **論文集**, Vol. 2011, No. 3, pp. 474–479 (2011).

[4] Curtsinger, C., Livshits, B., Zorn, B. and Seifert, C.: Zozzle: Low-overhead Mostly Static JavaScript Malware Detection, Technical Report MSR-TR-2010-156 (2010).

[5] Curtsinger, C., Livshits, B., Zorn, B. and Seifert, C.: ZOZZLE: Fast and Precise In-browser JavaScript Malware Detection, *Proceedings of the 20th USENIX Conference on Security*, SEC'11, Berkeley, CA, USA, USENIX Association, pp. 3–3 (online), available from ⟨http://dl.acm.org/citation.cfm?id=2028067.2028070⟩ (2011).

[6] Likarish, P., Jung, E. and Jo, I.: Obfuscated malicious javascript detection using classification techniques, *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*, pp. 47 –54 (2009).

[7] 西田, 雅., 星澤, 裕., 笠間, 貴., 衛藤, 将., 井上, 大. and 中尾, 康.: 文字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, Technical Report 21 (2014).

[8] 松本, 悦., 満永, 拓., 近藤, 伸. and 力宗, 幸.: 難読化された JavaScript 攻撃コードの WAF による検出手法 (セキュリティ, 情報通信マネジメント, ライフログ活用技術, オフィス情報システム, 一般), **電子情報通信学会技術研究報告**. *ICM,* **情報通信マネジメント**, Vol. 110, No. 374, pp. 99–104 (online), available from ⟨http://ci.nii.ac.jp/naid/110008675902/⟩ (2011).

[9] Rieck, K., Krueger, T. and Dewald, A.: Cujo: Efficient Detection and Prevention of Drive-by-download Attacks, *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, New York, NY, USA, ACM, pp. 31–39 (online), DOI: 10.1145/1920261.1920267 (2010).

[10] 酒井, 裕. and 佐々木, 良.: 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**. *CSEC, [***コンピュータセキュリティ***]*, Vol. 2013, No. 29, pp. 1–6 (online), available from ⟨http://ci.nii.ac.jp/naid/110009551652/⟩ (2013).

[11] 秋山, 満., 神薗, 雅., 松木, 隆. and 畑田, 充.: マルウェア対策のための研究用データセット〜MWS Datasets 2014〜, Technical Report 19, 日本電信電話株式会社, NTT セキュアプラットフォーム研究所, 株式会社セキュアブレイン, 先端技術研究所／独立行政法人情報通信研究機構, 株式会社 FFRI, エヌ・ティ・ティ・コミュニケーションズ株式会社 (2014).