Computer Security Symposium 2016
11 - 13 October 2016

Webにおけるイベントトラッキング対策手法の提案と実装

細谷 竜平[†1] 宮田 大地[†2] 石川 貴之[†2] 角田 裕太[†2]
高橋 和司[†2] 安田 昂樹[†2] 八代 哲[†1] 齋藤 孝道[†1]

概要：Google Analytics や Ptengine などのアクセス解析ツールには，クリックやキー入力のような Web ページにおける閲覧者の行動を収集するイベントトラッキングという機能がある．この機能によって，Web サイトの管理者は Web ページでの閲覧者の行動を細かく分析することができる．しかし，このような情報収集は，プライバシーの観点において問題であると捉える閲覧者もいる．そこで，本論文では，ブラウザの拡張機能を用いて Web ページにおける JavaScript の特徴とリクエストの挙動からイベントトラッキングを検出し，防ぐ手法を提案する．

キーワード：イベントトラッキング，Web トラッキング，アクセス解析ツール，プライバシー

Proposal and Implementation of Countermeasure against Event Tracking on Web

Ryohei HOSOYA[†1] Daichi MIYATA[†2] Takayuki ISHIKAWA[†2]
Yuta TSUNODA[†2] Kazushi TAKAHASHI[†2] Koki YASUDA[†2]
Satoshi YASHIRO[†1] Takamichi SAITO[†1]

Abstract: Web analytics tools, e.g. Google Analytics , Ptengine have a function of Event Tracking, that track user's behavior like click or keystroke on Web page. It enables the administrator of Web site to analyze the user's behavior in detail. However, some users of the Web site feel displeasure at Event Tracking in terms of the privacy. Therefore, in this paper, we propose the countermeasure against Event Tracking by analyzing JavaScript and HTTP request on Web page.

Keywords: Event Tracking, Web Tracking, Web analytics tool, Privacy

1. はじめに

Web サイトの閲覧者のアクセス回数やページ遷移を分析する，所謂アクセス解析ツールというものがある．最近の調査[1]によると，企業におけるアクセス解析ツールの導入率は 33.2%を占めている事が分かった．

多くのアクセス解析ツールには，イベントトラッキングという機能がある．これは，クリック，キー入力，及びスクロールのような Web ページにおける閲覧者の行動を収集する機能である．イベントトラッキングによって，Web サイトの管理者は Web ページでの閲覧者の行動を細かく分析し，Web サイトのデザインの改良に役立てることができる．

しかし，イベントトラッキングによる入力情報の収集に対して，大半の Web サイトの閲覧者がプライバシーへの懸念を抱いているという現状がある．IPA の"2015 年度 情報セキュリティの脅威に対する意識調査[2]"によると，イベントトラッキングのようなブラウザへの入力情報が収集される行為に対する意識について，気になる（"たいへん気になる"と"少し気になる"）と回答した人の割合は，6 割以上であった．

本論文では，ブラウザの拡張機能を用いてイベントトラッキングを検出し，その実行を防ぐ手法の提案，実装，及び評価を行う．イベントトラッキングの検出には，Web ページ内のクリック，キー入力，及びスクロールのイベントに対して反応する JavaScript のメソッドと，イベント発生後に行われる外部ドメインへのページ遷移を伴わないリクエストの有無から判断している．

2. イベントトラッキング

本節では，一般的に利用されるイベントトラッキングの仕組みを説明する（図 1）．イベントトラッキングとは，Web サイト閲覧者によるクリック，キー入力，及びスクロールのような行動を収集する機能である．図 1 の（4）の矢印は端末による JavaScript の実行を表し，（4）以外の矢印は端末間で送信される HTTP リクエストを表している．

（1） 閲覧者 A がアクセス解析ツール Y を導入している Web サイト X にアクセスする．
（2） Web サイト X が閲覧者 A に Web ページを提供する．
（3） 閲覧者 A がアクセス解析ツールのサーバ Y から，イベントトラッキングを実行する JavaScript をダウンロ

†1 明治大学
Meiji University
†2 明治大学大学院
Graduate School of Meiji University

ードする．

(4) イベントトラッキングを実行する JavaScript が Web サイトの閲覧者 A のブラウザ上で実行される．

(5) Web サイト X の閲覧者 A がそのページ上でクリック，キー入力，及びスクロールなどの行動をした際に，その情報がアクセス解析ツールのサーバに送信される．送信される情報は，例えば，どの位置でクリックしたのか，どの場所でスクロールを止めたのか，といった情報である．

(6) アクセス解析ツールのサーバ Y が，行動に関する情報をまとめた Web ページを，Web サイト X の管理者に提供する．

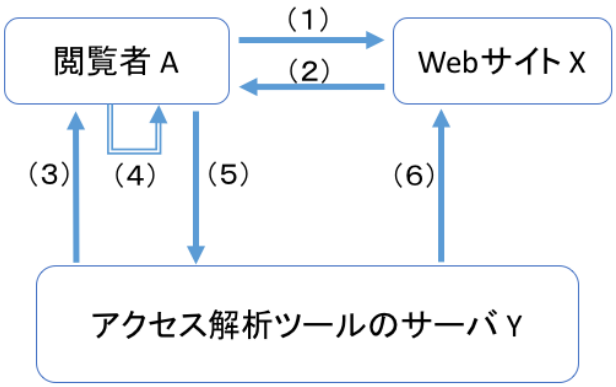

図 1 イベントトラッキングの仕組み

3. 既存の対策技術

本節では，イベントトラッキングに対して，現在，対策技術ととらえることができる Ghostery と NoScript を紹介する．

3.1 Ghostery

Ghostery[3]は，Web ページにおいて使用されているアクセス解析ツールを調査し，閲覧者に警告するブラウザ拡張機能である．このブラウザ拡張機能により，Google Analytics や Ptengine などの既存のアクセス解析ツールによるイベントトラッキングの実行を防ぐことができる．

しかしながら，Ghostery は，既存のアクセス解析ツールの実行を防ぐことが可能であるが，未知のアクセス解析ツールの実行を防ぐことができない．Ghostery は，現在閲覧している Web ページに使用されているアクセス解析ツールや広告ツールをブラウザ上に明示し，Web サイトの閲覧者がその実行の可否を指定するという機能を持つ．この機能は，Web ページにおいて JavaScript をダウンロードする際に参照する URL と Ghostery が持つアクセス解析ツールの URL のリストを照らし合わせることで，使用されているアクセス解析ツールや広告ツールを判断している．

3.2 NoScript

NoScript[4]は，ブラウザにおける JavaScript の実行を禁止する Firefox のアドオンである．これにより，JavaScript で動作するイベントトラッキングの実行を禁止することができる．

しかし，Alexa[5]による Web サイトアクセス数ランキング TOP100,000 のうち，JavaScript の利用率は 92.0%と非常に高く，JavaScript の実行を禁止することは現実的な対策とはいえない．

4. 提案手法

4.1 概要

提案手法は，閲覧中の Web ページ内の JavaScript のコードを検査し，ブラウザから送信される HTTP リクエストを解析することで，イベントトラッキングを検出する．本研究では Google Chrome[6]の拡張機能により提案手法を実装した．提案手法の概略図を図 2 に示す．提案手法を実装した拡張機能（以下，提案システムという）は閲覧者がブラウザで Web ページを表示した際，以下のように動作する．

1. ブラウザのタブに表示されている Web ページの JavaScript コード部を取得する
2. 取得した JavaScript コード部において閲覧者のクリック，キー入力，及びスクロールに反応するイベントハンドラとイベントリスナーを検出する
3. 閲覧者による入力が行われた後，ページ遷移を伴わないリクエストを検出する
4. ページ遷移を伴わないリクエストの送信先が閲覧ページとは異なる外部ドメインであるか判定する．外部ドメインへの送信であると判定したらイベントトラッキングが実行されていると判断する

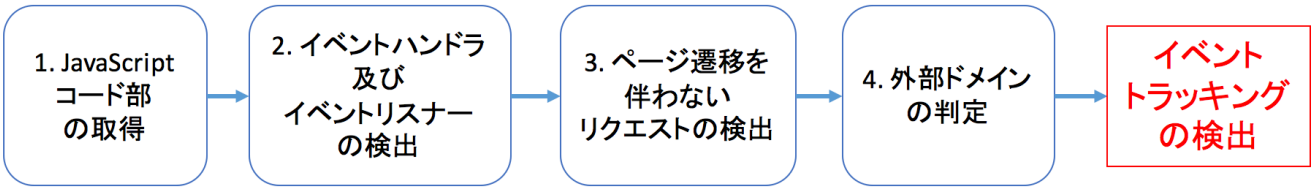

図 2 提案手法の概略図

提案システムが検査対象の Web ページに対してイベントトラッキングが実行されていると判断した場合，イベント発生後に行われるページ遷移を伴わないリクエストを遮断することで，イベントトラッキングを防ぐ．

4.2 イベントハンドラの検出

提案システムは，検査対象の Web ページを読み込み，クリック，キーボード，及びスクロールのイベントに対して発火するイベントハンドラを検出する．

提案システムでは，以下の図 3 のように Web ページ上のイベントハンドラに対して関数が実装されているかを調べることで，そのイベントハンドラを検出する．

```
if(document.onclick) {
    //クリックに対するイベントハンドラを検出
}
```

図 3　提案システムにおけるイベントハンドラの検出方法

提案システムは，上記の図 3 のようなイベントハンドラを検出する処理をクリック，キー入力，及びスクロールの全てに対して行う．その際，表 1 に示した 5 種類のイベントハンドラを検出する．

表 1 提案システムにおいて検出するイベントハンドラの種類

閲覧者の行動	イベントハンドラ
クリック	onclick
ダブルクリック	ondblclick
キー入力	onkeydown
	onkeypress
スクロール	onscroll

4.3 イベントリスナーの検出

イベントリスナーは，イベントハンドラと違い，検出に使用できるプロパティや検出するための API が用意されていない．よって，提案システムでは Web ページ上の全ての JavaScript コードを取得し，文字列を静的に解析することで検出する手法をとる．例えば，提案システムがクリックに対するイベントリスナーを検出する場合は以下の図 4 のように実装する．図 4 では，”addEventListener(‘click’, …)”などのイベントリスナーのコードに対応するような正規表現を用意し，それとマッチする JavaScript の文字列を検索している．

```
//イベントリスナーの文字列を表す正規表現の登録
var reg_click =
/(addEventListener).+[(click)(dblclick)]|[(clic
k)(dblclick)].+(addEventListener)/;
...
//上記で定義した正規表現と Web ページ上の JavaScript
の文字列がマッチしているか
if (reg_click.test(dataArray[num])) {
     //クリックに対するイベントリスナーの検出
}
```

図 4　提案システムにおけるイベントリスナーの検出方法

提案システムは，図 4 のようなイベントリスナーを検出する処理をクリック，キー入力，及びスクロールの 3 つに対して行う．その際，提案システムは以下の表 2 に列挙した 5 種類のイベントリスナーを検出する．

表 2 提案システムにおいて検出するイベントリスナーの種類

閲覧者の行動	イベントリスナー
クリック	addEventListener("click",…
ダブルクリック	addEventListener("dblclick",…
キー入力	addEventListener("keydown",…
	addEventListener("keypress",…
スクロール	addEventListener("scroll",…

4.4 ページ遷移を伴わないリクエストの検出

ページ遷移を伴わないリクエストの検出方法として，Google Chrome の拡張機能で用意されている webRequestAPI[7]を使用する．これは，ブラウザから送られるリクエストを監視し，特定のリクエストに対して HTTP ヘッダの書き換えやイベント実行などの処理を行う API である．この API を使用してページ遷移を伴わないリクエストを検出する方法を図 5 に示す．図 5 では，ブラウザからリクエストが送られる前に発火する onBeforeRequest.addListener というイベントリスナーを使用しており，このイベントリスナーの第 2 引数（types: 以下）にページ遷移を伴わない種類のリクエストを設定している．これにより，図 5 のイベントリスナーはページ遷移を伴わないリクエストにのみ発火する．

```
chrome.webRequest.onBeforeRequest.addListener(
     function (details) {/*省略*/},
     types:["xmlhttprequest","image","sub_fram
e","stylesheet","script", "other"],
...
```

図 5　提案システムにおけるページ遷移を伴わないリクエストの検出方法

提案システムにおいて検出するページ遷移を伴わないリクエストの種類を以下の表 3 に示す.

表 3 提案システムにおけるページ遷移を伴わないリクエストの判定基準としたリクエスト

リクエストの種類	解説
xmlhttprequest	XMLHttpRequest（XHR）によるリクエスト
image	画像を読み込むリクエスト
sub_frame	フレーム内でのリクエスト
stylesheet	CSS を読み込むリクエスト
script	JavaScript を読み込むリクエスト
other	ブラウザではなく閲覧者による挙動や, 他のプロセスなどから発生するリクエスト

4.5 外部ドメインの判定

4.4 節の図 5 で示したイベントリスナーでページ遷移を伴わないリクエストを検出した場合，続いて外部ドメインか否かの判定を行う．外部ドメインか否かの判定は，閲覧者が見ている Web ページの URL と，4.4 節で紹介した Google Chrome の webRequestAPI で検出したリクエストの宛先 URL からそれぞれ抽出したドメインの比較により行う．ドメイン抽出の範囲は以下の 2 通りがあり，提案システムの利用者はいずれかを選択する．

（1） Fully Qualified Domain Name（FQDN）の抽出
（2） FQDN からサブドメインを省いたドメイン抽出

（1）の方法は，URL から FQDN の抽出を行う．FQDN の抽出方法を以下の図 6 に示す．図 6 では，URL の文字列から正規表現を用いてスキーム名（[httpsfile]+:¥/{2,3}），ポート番号（:?[0-9]*?）を取り除いた形を抽出している．

```
//URL の文字列(str)から FQDN を抽出
formkeyEventDomain = str.match
(/^[httpsfile]+:¥/{2,3}([0-9a-z¥.¥-:]+):?[0-
9]*?/i)[1];
```

図 6 URL の文字列から FQDN を抽出する方法

例えば，Web サイト閲覧者が明治大学情報セキュリティ研究室のホームページ（”https://www.saitolab.org”）を閲覧している際に，イベントトラッキングのスクリプトが Google Analytics のサーバ（”https://www.google-analytics.com”）へ入力情報を送信していた場合，それらの URL から，（1）の方法でドメイン抽出し，比較すると，以下の図 7 のように異なるドメインと判定される．

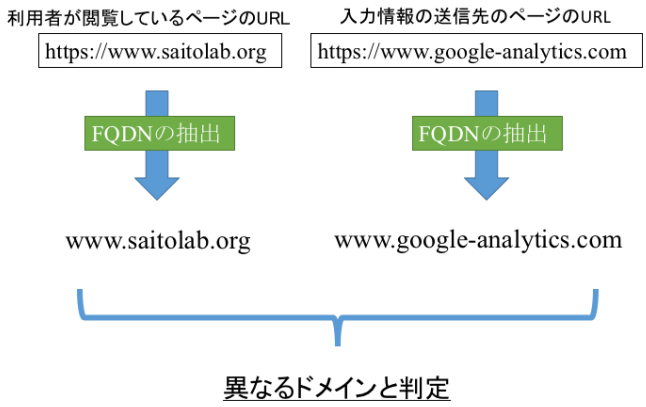

図 7 閲覧ページの URL と Google Analytics の URL から FQDN を抽出した場合の比較

（2）の方法は，JavaScript のライブラリの 1 つである tldjs[8]を用いて行う．これは，Mozilla[9]の提供するドメインのリスト[10]を利用して柔軟なドメイン操作を提供するライブラリである．図 7 の例に対して（2）の方法でドメイン抽出を行った場合，以下の図 8 のようになる．

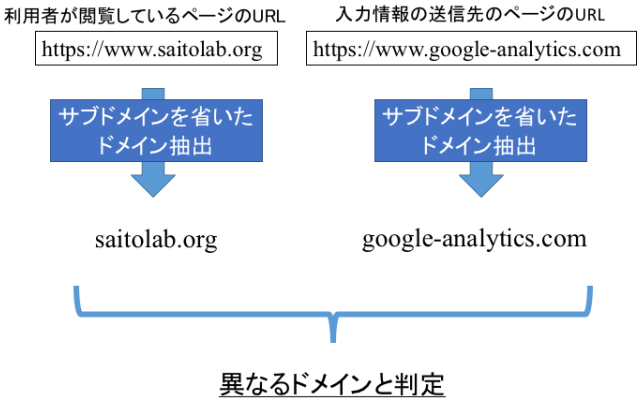

図 8　閲覧ページの URL と Google Analytics の URL から tldjs を用いて抽出したドメインの比較

5. 評価

提案手法の評価をするために，提案システムを用いて行った以下の 2 つの調査について示す．

- 代表的なアクセス解析ツールによるイベントトラッキングの検出数の調査
- Alexa Top100 ドメインにおけるイベントトラッキングに対する検出数の調査

上記のいずれの調査においても，Google Chrome のバージョン 51.0.2704.103 を使用して行った．

5.1 代表的なアクセス解析ツールに対する提案手法による検出数の調査

代表的なアクセス解析ツールによるイベントトラッキングに対して提案システムが検出するか調査を行った．

調査対象は，trust radius[11]にてレビューされたアクセス解析ツールのうち，評価のスコアが上位を占める 14 個のアクセス解析ツールとした．

調査手順としては，各アクセス解析ツールを使用している Web サイト[12]において，Google Chrome のデベロッパーツールを使用して，閲覧者による行動の後に送信されるページ遷移を伴わないリクエストのクエリを確認することで，アクセス解析ツールがイベントトラッキングを実装しているかを確認した．その上で，イベントトラッキングを実装している Web サイトに対して，提案手法を適用した Google Chrome でアクセスし，クリック，キー入力，及びスクロールを行い，提案システムがイベントトラッキングを検出するかどうかを調査した．

調査の結果を以下の表 4 に示す．提案システムによりイベントトラッキングを検出したアクセス解析ツールを“〇”で表し，検出していない場合，“×”で表している．また，イベントトラッキングが実装されてないアクセス解析ツールは“-”と表している．

表 4 アクセス解析ツールごとの提案手法による検出の有無

アクセス解析ツール	(1) FQDN によるドメイン比較	(2) FQDN からサブドメインを省いた形によるドメイン比較
GoSquared[13]	〇	〇
Chartbeat[14]	〇	〇
Google Analytics[15]	〇	〇
AT Internet[16]	〇	〇
Google Analytics Premium[17]	〇	〇
Woopra[18]	〇	〇
Adobe Analytics[19]	〇	〇
Piwik[20]	〇	〇
Mixpanel[21]	〇	〇
Kissmetrics[22]	-	-
Digital Analytics Enterprise[23]	-	-
IBM Digital Analytics[24]	〇	〇
StatCounter[25]	-	-
Webtrends Analytics[26]	-	-

提案手法は代表的なアクセス解析ツールによるイベントトラッキングを全て検出できることが分かった．表 4 より，アクセス解析ツール 14 個中，提案システムでイベントトラッキングを検出したのは 10 個であった．表 4 において，”〇”と表示しているアクセス解析ツールを使用している Web サイトの挙動を確認したところ，閲覧者による行動がある度にアクセス解析ツールのサーバへページ遷移を伴わないリクエストが送信されていることが確認できた．

5.2 Alexa Top 100 ドメインに対する提案手法による検出数の調査

この調査では，提案手法が正しくイベントトラッキングを検出したか調査した．調査方法は，Alexa による世界の Web サイトアクセス数ランキング TOP100（2016 年 7 月時点）の Web サイトに対して，提案システムを適用した Google Chrome でアクセスし，クリック，キー入力，及びスクロールを行い，提案システムがイベントトラッキングを検出するかどうかを調査した．その後，それらの検出がイベントトラッキングによるものであるかページ遷移を伴わないリクエストの送信先ドメインから判断した．

提案システムによってイベントトラッキングを検出した

Web サイトの数を以下の表 5 に示す．ただし，それらの Web サイトにおいて，ページ遷移を伴わないリクエストの送信先のドメイン名がアクセス解析ツールのドメイン名である場合，正常に検出したとみなす．ページ遷移を伴わないリクエストの送信先のドメイン名がアクセス解析ツールのドメイン名ではない場合，誤検出したとみなす．また，イベントトラッキングを正常に検出し，かつ誤検出もした Web サイトは両方の検出数にカウントした．

表 5 Alexa Top100 ドメインにおける提案システムの検出数

ドメイン抽出法	正常に検出した数（100 サイト中）	誤検出した数（100 サイト中）
(1) FQDN によるドメイン比較	24 サイト	26 サイト
(2) FQDN からサブドメインを省いた形によるドメイン比較	22 サイト	7 サイト

表 5 より，ドメインの抽出法によって誤検出した数が大きく変わることが分かった．(1) と (2) における誤検出の原因は，全て検索フォームでの入力補完機能であった．(1) のドメイン抽出法で誤検出したサイトの大半は，ワード予測の問い合わせ先のドメインが閲覧ページのドメインと比べてサブドメインのみが異なる場合であった．例えば，Amazon[27]のトップページ（“https://www.amazon.com.jp”）における検索フォームに文字を入力すると，入力補完機能による問い合わせを行うために，”https://completion.amazon.co.jp”にページ遷移を伴わないリクエストを送信している．この瞬間のリクエストに対して，上記の（1）の方法でドメイン比較すると，以下の図 9 のようになる．

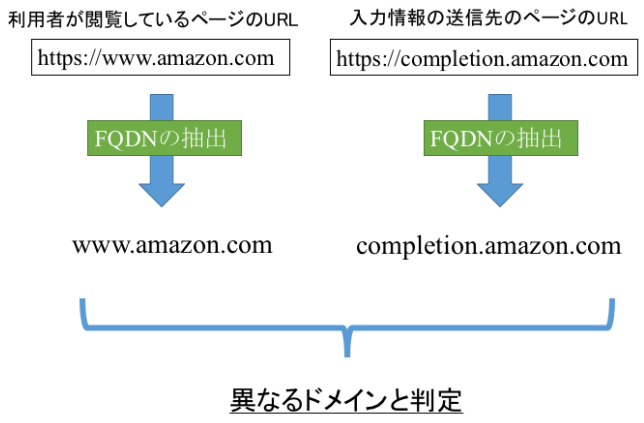

図 9　Amazon の Web ページの URL と検索フォームでの入力情報の送信先 URL から抽出した FQDN の比較

図 9 では，本来同じ Amazon の Web サーバに送信しているリクエストでも，サブドメインが異なる場合があるので，ドメインの比較で異なるものと判断されてしまう．その結果，このワード予測の機能もイベントトラッキングとみなされてしまった．それに対し，(2) では，閲覧ページとは全く異なるドメインに対して入力補完機能の問い合わせを行っていた場合のみ誤検出していた．しかし，このような入力補完を行う Web サイトは少数であった．

6. 考察

6.1 同一ドメインでの通信について

5.4 節の評価の結果から，FQDN の比較による外部ドメインの判定では，誤検出が多いことが分かった．しかし，必要最低限の相手にしか入力情報を与えたくない Web サイトの閲覧者とっては，あらゆる外部ドメインへのリクエストを遮断する方法が適しているので，FQDN の比較による方法も有用であると考えた．

6.2 既存の対策技術との比較

3.2 節で取り上げた Ghostery は，未知のアクセス解析ツールには対応できないという欠点がある．その一方で，提案手法は，URL ではなく実際の JavaScript のコードからイベントトラッキングを検出しているので，未知のアクセス解析ツールにも対応できる可能性がある．

3.3 節で取り上げた NoScript は，JavaScript を使用している Web サイトのコンテンツや一部の機能が制限されるか，利用ができなくなるという欠点がある．その一方で，提案手法は，イベントトラッキングを実行する JavaScript のみを防ぐので，Web サイトの閲覧者は提案手法に影響されずに Web サイトを閲覧することができる．

6.3 提案手法のイベントトラッキング対策以外の用途

提案手法は，JavaScript を用いて閲覧者の入力内容を窃取する攻撃に対しても有効な対策になると考えられる．

Web ページの改ざんや XSS（クロスサイトスクリプティング）によって，悪意のある JavaScript を閲覧者のブラウザ上で実行させる攻撃が報告されている[28]．これらの攻撃を利用して，クリックやキー入力に対して発火するイベントハンドラやイベントリスナーを Web サイトの閲覧者のブラウザ上で実行させる．こうすることで，攻撃者は閲覧者によるクリックやキー入力の情報を窃取することができ，その結果，閲覧者の ID やパスワードの文字列を不正に取得することが可能であることが実証されている[29]．

このような閲覧者の入力情報を窃取する攻撃は，イベントハンドラやイベントリスナーを利用して，Web サイトの閲覧者の入力を記録し，その入力情報をページ遷移のないリクエストで攻撃者のサーバへ送信する．ゆえに，この攻撃は，イベントトラッキングと同様の仕組みで動作するので，提案手法でも検出することが可能である．

7. 課題

提案手法における課題として，Web ページの JavaScript コードが難読化されていた場合，4.3 節で示した検出方法では，イベントリスナーで追加されたイベントの検出ができず，結果イベントトラッキングを検出することができなくなることが挙げられる．

JSFuck[30]を用いて，クリックに対するイベントリスナーの JavaScript のコードを難読化した場合の例を図 10 に示す．図 10 では，”document.addEventListener(‘click’, function(){…});”という JavaScript の文字列を，6 種類の記号からなる文字列に置き換えている．

```
document.addEventListener('click', function(){…});

        ⬇ 難読化

[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[]
)[!+[]+!+[]]+(!![]+[])[+[]]   //以下略 (計63402文字)
```

図 10　イベントリスナーの JavaScript のコードを難読化した例

提案手法は，4.3 節で示した通り，文字列の比較によるイベントリスナーの検出をする．よって，イベントトラッキングを実行する JavaScript コードが難読化されていた場合，提案手法はイベントトラッキングを検出することができない．

8. まとめ

本論文では，ブラウザの拡張機能を用いて Web ページにおける閲覧者のクリック，キー入力およびスクロールに反応するイベントハンドラ，イベントリスナーの検出，および外部ドメインへのページ遷移を伴わないリクエストの検出をすることによって，イベントトラッキングを検出し，防ぐ手法を提案した．また，実装した拡張機能を評価したところ，提案手法は代表的なアクセス解析ツールによるイベントトラッキングを全て検出できることが分かった．

参考文献

[1] “キーマンズネット マーケティング支援ツールの導入状況（2015 年）”. http://www.keyman.or.jp/at/30008476/, (参照 2016/08/04)

[2] ”IPA 2015 年度情報セキュリティの脅威に対する意識調査”. https://www.ipa.go.jp/files/000050002.pdf, (参照 2016-07-19).

[3] “Ghostery”. https://www.ghostery.com, (参照 2016-07-19).

[4] “NoScript”. https://noscript.net, (参照 2016-07-19).

[5] “Alexa”. http://www.alexa.com/(参照 2016-07-19).

[6] “Google Chrome”. https://www.google.co.jp/chrome/browser/desktop,(参照 2016/07/21)

[7] “Chrome.webrequest”. https://developer.chrome.com/extensions/webRequest, (参照 2016/07/21)

[8] “tldjs”. https://www.npmjs.com/package/tldjs, (参照 2016/07/21)

[9] “Mozilla”. https://www.mozilla.jp,(参照 2016/08/01)

[10] “Public Suffix List”. https://publicsuffix.org/list/,(参照 2016/08/01)

[11] “trustradius”. https://www.trustradius.com/web-analytics,(参照 2016/07/28)

[12] “Wappalyzer”. https://wappalyzer.com, (参照 2016/07/29)

[13] “GoSquared”. https://www.gosquared.com,(参照 2016/07/29)

[14] “Chartbeat”. https://chartbeat.com, (参照 2016/07/29)

[15] “Google Analytics”. https://www.google.com/intl/ja_jp/analytics, (参照 2016/08/04)

[16] “AT Internet”. http://www.atinternet.com, (参照 2016/07/29)

[17] “Google Analytics Premium” https://www.google.co.jp/intl/ja_ALL/analytics/premium/index.html, (参照 2016/07/29)

[18] “Woopra”. https://www.woopra.com, (参照 2016/07/29)

[19] “Adobe Analytics”. http://www.adobe.com/jp/marketing-cloud/web-analytics.html, (参照 2016/07/29)

[20] “Piwik”. https://piwik.org, (参照 2016/07/29)

[21] “Mixpanel”. https://mixpanel.com, (参照 2016/07/29)

[22] “Kissmetrics”. https://www.kissmetrics.com, (参照 2016/07/29)

[23] “Digital Analytics Enterprise”. https://www.comscore.com/jpn/Products/Enterprise-Analytics/Digital-Analytics-Enterprise, (参照 2016/07/29)

[24] “IBM Digital Analytics”. http://www-03.ibm.com/software/products/ja/digital-analytics, (参照 2016/07/29)

[25] “StatCounter”. https://statcounter.com, (参照 2016/07/29)

[26] “Webtrends Analytics”. https://www.webtrends.com, (参照 2016/07/29)

[27] “Amazon”. https://www.amazon.co.jp,(参照 2016/07/26)

[28] “ソフトウェア等の脆弱性関連情報に関する届出状況(IPA)”. https://www.ipa.go.jp/security/vuln/report/vuln2012q4.html,(参照 2016/07/27)

[29] Wade Alcorn , Christian Frichot , Michele Orru. ブラウザハック. 翔泳社, 2016, 167p.

[30] “JSFuck”. http://www.jsfuck.com, (参照 2016/07/23)