Vol.2019-IOT-44 No.25
2019/3/8

ブラウザの実装の差異を利用したプラットフォーム推定

Platforms Estimation based on the Difference of Implementations for Browsers

眞門裕伸† 廣瀬幸† 猪俣敦夫†

概要：Web サイトにアクセスしてきた端末を識別する手法として，Web Browser Fingerprinting がある．この手法は端末から採取可能な特徴点を組み合わせることで端末の識別を行う．特徴点として利用される情報の 1 つに User Agent 文字列があるが，これは利用者が変更可能であり，変更された場合は正しいプラットフォーム情報を得ることができない．そこで本論文では，ブラウザごとの JavaScript 実装の差異を利用し，User Agent 文字列を使用せずにプラットフォームの推定と Web Browser Fingerprinting による端末の識別を行うことを試みる．実験の結果，User Agent 文字列を使用した場合との識別精度の差はごく僅かなものとなった．

キーワード：Web Browser Fingerprinting，ブラウザ，JavaScript

1. はじめに

Web サイトにアクセスした利用者の端末を識別する手法として，Web Browser Fingerprinting がある．Web Browser Fingerprinting では，端末から採取可能な情報のうち，利用者の端末の識別に利用可能なものを特徴点として組み合わせることで，端末の識別を行う．Web Browser Fingerprinting はターゲティング広告の配信やリスクベース認証を目的として，Web 行動追跡に利用されている．

Web Browser Fingerprinting における特徴点の 1 つに, User Agent 文字列（以降，User Agent とする）がある．しかし，利用者はブラウザの設定や拡張機能などを使用して User Agent を書き換えることができる．利用者によって User Agent が書き換えられた場合，プラットフォームの正しい判定が困難となる．また，磯ら[1]により，User Agent は短期間のうちに変化することが明らかとなっている．User Agent の書き換え，または短期間での変化があった場合，Web Browser Fingerprinting による識別にも影響を与える．さらに，Safari において User Agent の固定が検討され，開発版に導入されていた[2]. 理由の 1 つとして, Fingerprinting 対策が挙げられている．後のリリースで固定はされなくなった[3]ものの，各ブラウザが今後 User Agent を固定する可能性も考えられる．

User Agent によるプラットフォーム推定は, Exploit Kit による攻撃対象の特定にも使用されている[4]．しかし前述のように User Agent は変更・固定が可能であるため，User Agent を使用しない手法でプラットフォームの推定が行われる可能性があると考えた．

そこで本論文では，プラットフォームの推定と Web Browser Fingerprinting を User Agent を使用せずに行うことを試みた．User Agent の代わりとなる特徴点として，ブラウザの種類やバージョンによる実装の差異を利用し，実験を行なった．

2. Web Browser Fingerprinting

前章で述べたように，特徴点の組み合わせによって利用者の端末を識別する手法を Web Browser Fingerprinting（以降，Fingerprinting とする）という．また，Fingerprinting に使用する特徴点とその組み合わせを Browser Fingerprint（以降，Fingerprint とする）という．Fingerprint は JavaScript や CSS などを用いて採取する．

2.1 関連研究

Eckersley[5]は，Flash または Java が有効なブラウザは，Fingerprinting を用いることで，採取したサンプルのうち 94.2%が識別可能であることを示した．このときに使用された Fingerprint は，User Agent，HTTP ACCEPT ヘッダ，クッキーの使用可否，画面解像度，タイムゾーン，プラグインリスト，フォントリスト，Super Cookie のテストを含むものであった．これらの特徴点のうち，User Agent はプラグインリストとフォントリストに次いで 3 番目に識別性が高かった．本論文では User Agent に着目し，User Agent を使用せずにプラットフォームの識別と Fingerprinting を行う．

野田ら[6]は，JavaScript の Math オブジェクトの演算結果がプラットフォームによって異なることに着目し，特徴点として利用した．実験により，Math.cos の演算結果を User Agent の代わりに使用した場合，Fingerprinting による識別精度には大差がないことが示された．しかし，Math.cos の演算結果を利用してバージョンの識別が可能なブラウザは限られていた．本論文では，ブラウザの種類やバージョン

† 東京電機大学 〒120-8551 東京都足立区千住旭町 5. Tokyo Denki University, 5, Senju-Asahi-cho, Adachi-ku , Tokyo 120- 8551, Japan.

について，より詳細な識別を試みる．

3. 提案手法

User Agent はプラットフォームの判定に使用され，Fingerprinting における特徴点の 1 つにもなっている．しかし，User Agent は実際のプラットフォーム情報とは異なる内容に変更が可能である．変更が行われた場合，User Agent を使用したプラットフォームの判定や Fingerprinting の結果に影響を与える．そこで本論文では，ブラウザの実装の差異を利用し，User Agent を使用せずにプラットフォームの推定を行う手法を提案する．

ブラウザに実装されている JavaScript は ECMAScript[7]として標準化されている．しかし JavaScript リファレンス[8]によると，全ての機能が各ブラウザに実装されているわけではない．また，既に実装されている機能に関しても，実装のタイミングや実装自体がブラウザごとに異なる場合がある．さらに，ブラウザによっては独自の機能が実装されている場合もある．これらの機能のブラウザにおける実装有無の判定は，これまでも Web サイトにおいて正しい表示を行う目的で用いられてきた[9].

本論文では，ブラウザの種類やバージョンによって発生する，機能の実装の有無や実行結果の違いを，実装の差異として定義する．また，複数の機能について実装の差異を判定し，その結果を提案手法による特徴点とする．

この特徴点を利用し，User Agent を使用せずにブラウザの種類とバージョンを推定できると考えた. さらに Math オブジェクトの演算結果と組み合わせ，より詳細なプラットフォーム推定が可能になると考えられる．また，この特徴点を User Agent の代わりに使用し，User Agent の変更の影響を受けずに Fingerprinting を行うことができると考えた．

4. 実験

提案手法の検証のため，以下の 2 つの実験を行った．

- 実験 1
  提案手法による特徴点から，ブラウザの種類とバージョンを推定する．
- 実験 2
  提案手法による特徴点を使用して Fingerprinting による識別を行い，User Agent や Math オブジェクトをそれぞれ使用した場合と精度を比較する．

本章ではこれらの実験について説明する．

4.1 対象とするブラウザ

本論文の実験では，主要なブラウザとして以下のものを対象とした．

- Google Chrome
- Mozilla Firefox
- Microsoft Edge
- Internet Explorer
- Safari

各ブラウザのバージョンは，Chrome[10]や Firefox[11]のようにバージョンアップの頻度が高いものは 2018 年 11 月 30 日時点での最新版（Chrome 70，Firefox 63）から 10 バージョン程度前，それ以外のブラウザに関しては 5 バージョン程度前のバージョンまでを対象とした．

4.2 提案手法に使用する機能の選定

提案手法による特徴点に使用する機能の選定を行った．

4.2.1 選定手法

選定にあたり，MDN が提供している JavaScript リファレンス[8]と，MDN が管理するリポジトリ[12]を主に参照した．このリポジトリには，JavaScript を含む Web 技術についてブラウザごとの互換性がまとめられている．

対象とするブラウザとバージョンについて，より多くのブラウザにおいてバージョンごとに差異がある機能を MDN のリポジトリから抽出し，選定を行った．これはより少ない機能数でより多くのブラウザ，バージョンを識別するためである．

4.2.2 選定した機能

選定の結果，実験に使用した機能を表 1 にまとめた．

表 1　提案手法に使用した機能

No.	機能
1	Symbol.prototype.description
2	Array.prototype.flat
3	Array.prototype.values
4	ResizeObserver
5	Intl.PluralRules
6	Intl.getCanonicalLocales
7	Intl.DateTimeFormat, hourCycle
8	Intl.Collator.prototype.compare
9	WebAssembly.compileStreaming
10	WebAssembly.Global
11	SharedArrayBuffer
12	String.prototype.padEnd
13	Array.prototype.toSource
14	Object.fromEntries
15	Function.prototype.toString
16	Promise.prototype.finally
17	PushManager
18	MutationObserver

19	Window.setImmediate
20	window.atob
21	window.getSelection
22	window.orientationå

4.3 Fingerprint の採取

Fingerprint 採取専用のサイトを用意し，アクセスのあった端末のブラウザから Fingerprint の採取を行った．アクセス時に，使用しているプラットフォームの情報を入力させた．また，採取時に生成した UID をブラウザの Cookie に保存し，採取した Fingerprint や入力されたブラウザの種類，バージョンの情報と紐づけて Fingerprint 採取サイトのデータベースにも保存した．この UID を用いることで，実験協力者の端末のブラウザを一意に識別できる．

4.3.1 Fingerprint の採取フロー

Fingerprint 採取サイトにおける採取フローを図 1 に示す．

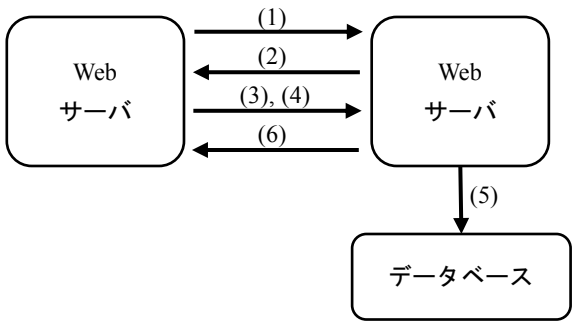

図 1　Fingerprint の採取フロー

(1) 実験協力者は Fingerprint 採取用ページのアドレスにアクセスする．
(2) Web サーバは Fingerprint 採取用のページを送信する．このページには，表 2 に示す特徴点の採取と送信を行う処理が含まれている．Web ブラウザは受信したページを表示する．
(3) 実験協力者は，使用しているプラットフォームの情報を入力し，送信ボタンを押す．
(4) 実験協力者が送信ボタンを押すと，Web ブラウザは表 2 に示す特徴点を採取する．採取された特徴点は，入力されたプラットフォーム情報，端末のブラウザ識別用の UID と共に，Web サーバへ送信される．このとき，ブラウザの Cookie に UID が保存されていない場合は UID を生成し，Cookie に保存する．この UID を用いることで，実験協力者の端末のブラウザを一意に識別できる．
(5) Web サーバは(3)で送信された UID とプラットフォーム情報，特徴点をデータベースに保存する．
(6) Web サーバは Fingerprint の採取が完了したことを示すページを送信し，Web ブラウザは受信したページを表示する．

4.4 実験に使用した特徴点

実験に使用した 22 個の特徴点を表 2 にまとめた．本節では，本論文の提案手法に関連する特徴点を中心に説明する．

表 2　採取した特徴点

No.	特徴点
1	ブラウザの実装の差異
2	Math オブジェクト
3	User Agent
4	ブラウザの言語
5	色深度
6	ピクセル比
7	論理プロセッサ数
8	画面解像度
9	ウィンドウの有効領域
10	タイムゾーン
11	Session Storage の有無
12	Local Storage の有無
13	IndexedDB の有無
14	CPU 情報
15	プラットフォーム
16	DoNotTrack の設定
17	プラグインリスト
18	Canvas fingerprinting
19	WebGL fingerprinting
20	AdBlock の有無
21	タッチスクリーンの有無
22	フォントリスト

4.4.1 ブラウザの実装の差異（提案手法）

表 1 に示した各機能について，実装の有無や実行結果を調べるメソッドを作成した．特徴点として，このメソッドの実行結果を使用した．

4.4.2 Math オブジェクト

野田ら[6]により，Math オブジェクト[13]の Math.cos がプラットフォーム推定に適していることが示された．そこで，Math.cos の演算結果を特徴点として採取した．引数には，10 のべき乗（1, 10, 100, …, 1e300）を使用した．

4.4.3 その他の特徴点

User Agent を含む 20 個の特徴点は，Fingerprintjs2[14]を使用して採取した．

4.5 実験に使用したサンプル

実験協力者に Fingerprint 採取サイトへのアクセスを依頼し，採取したデータをサンプルとして使用した．採取期間

は 2018 年 11 月 30 日から同年 12 月 17 日の 18 日間である．

採取したサンプルは 235 個で，複数回のアクセスがあった端末は 37 台であった．

4.6 実験 1

この実験では，ブラウザの機能の実装が異なることを利用して，採取したサンプルを分類し，ブラウザの種類とバージョンを推定する．さらに，提案手法と同様に User Agent を使用しない手法として，Math オブジェクトの演算結果と組み合わせた推定も行う．

サンプルの分類は，提案手法による特徴点，または提案手法による特徴点と Math オブジェクトによる特徴点を文字列として連結し，ハッシュ化したものを使用する．

4.6.1 使用したサンプル

対象とするブラウザのサンプルを中心に，採取した全てのサンプルを使用した．

採取したサンプルに含まれていたブラウザからのアクセスのうち，対象となるブラウザの種類とサンプル数を表 3 に示す．

表 3　実験 1 のサンプル

OS	ブラウザ	サンプル数
Windows, macOS, Linux	Chrome	103
Android	Chrome	23
Windows, macOS, Linux	Firefox	23
Windows	Edge	10
Windows	Internet Explorer	35
macOS	Safari	29
iOS	Safari	11

対象外のブラウザについては，Samsung Internet Browser からのアクセスが 1 件あった．

4.6.2 推定結果の検証

ブラウザの種類とバージョンの推定結果について，Fingerprint の採取時に入力された情報と，採取した User Agent を使用して検証を行った．

4.7 実験 2

この実験では，ブラウザごとに機能の実装が異なることを User Agent の代わりの特徴点として利用し，Fingerprinting による識別を行う．また，特徴点として User Agent を使用した場合や Math オブジェクトを使用した場合と，Fingerprinting の識別精度を比較する．

4.7.1 使用したサンプル

採取した 235 個のサンプルをそのまま使用した．サンプル中で複数回のアクセスがあった端末は 37 台である．

4.7.2 Fingerprint の組み合わせ

Fingerprinting による識別に使用する Fingerprint の組み合わせを以下に示す．括弧内の番号は，表 2 に示した特徴点を指す．

- User Agent を使用する場合（No. 3, 4〜22）
- Math オブジェクトを使用する場合（No. 2, 4〜22）
- 提案手法による特徴点を使用する場合（No. 1, 4〜22）

4.7.3 Fingerprint 精度の評価手法

本論文では，ROC（Receiver Operating Characteristic）曲線の AUC（Area Under the Curve）の値を Fingerprinting の識別精度とする．ここで識別とは，同じ端末のブラウザから採取したサンプルを同一と判定し，異なる端末のブラウザから採取したサンプルを異なると判定することである．

Fingerprinting の各結果からそれぞれの TPR（True Positive Rate）と FPR（False Positive Rate）を求め，AUC の値を算出することで，精度の比較が可能となる．

5. 実験結果

5.1.1 実験 1

各特徴点による分類結果を表 4 に示す．

表 4　各特徴点による分類結果

特徴点	グループ数
User Agent	50
Math オブジェクト	8
提案手法	20
提案手法と Math オブジェクト	23

提案手法により，サンプルを 20 個のグループに分類することができた．プラットフォームの推定結果を表 5 に示す．

表 5　提案手法による推定結果

No.	OS	ブラウザ	バージョン
1	Windows, macOS, Linux	Chrome	70, 71
2			69
3			65
4	Android	Chrome	70, 71
5			69
6	Windows, macOS, Linux	Firefox	63, 64
7			62
8			60, 61
9			59

10			52
11	Windows	Edge	18
12			17
13	Windows	Internet Explorer	11
14			10
15			9
16			8
17	macOS	Safari	12
18			11.1
19	iOS	Safari	12
(20)	Android	Samsung Internet Browser	-

主にブラウザの種類とバージョンによって分類されている．提案手法と Math オブジェクトによる特徴点を組み合わせた場合，サンプルは 23 個のグループに分類された．プラットフォームの推定結果を表 6 に示す．

表 6　提案手法と Math オブジェクトの組み合わせによる推定結果

No.	OS	ブラウザ	バージョン
1	Windows, macOS, Linux	Chrome	70, 71
2			69
3			65
4	Android	Chrome	70, 71
5			69
6	Windows	Firefox	63, 64
7	macOS		
8	Linux		
9	macOS	Firefox	62
10	Linux		
11	Windows, macOS, Linux	Firefox	60, 61
12			59
13			52
14	Windows	Edge	18
15			17
16	Windows	Internet Explorer	11
17			10
18			9
19			8
20	macOS	Safari	12
21			11.1
22	iOS	Safari	12
(23)	Android	Samsung Internet Browser	-

Math オブジェクトを組み合わせた場合，Firefox のバージョン 62 以降で OS が分類されていることが分かる．Math オブジェクトのみの場合は，8 個のグループに分類される結果となった．

5.1.2 実験 2

User Agent，Math オブジェクト，提案手法をそれぞれ利用した Fingerprinting の精度を図 2 に示す．提案手法のみ AUC の値が 0.01%低くなっているが，3 つの手法において精度の差は小さいといえる．

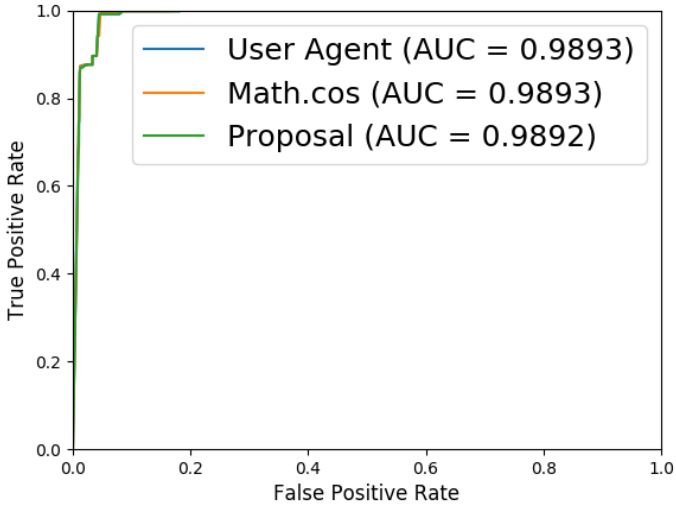

図 2　各特徴点を使用した Fingerprinting の AUC 値

6. 考察

6.1 実験 1

実験 1 では，採取したサンプルを，提案手法を用いることで 20 個のグループに分類することに成功した．Math オブジェクトの 8 個と比較して，より詳細なプラットフォーム推定が可能であるといえる．20 個のグループは主にブラウザの種類とバージョンによって分類され，OS については Android や iOS というモバイル OS に関してのみ判別が可能だった．表 5，表 6 中の Chrome 71 と Firefox 64 は Fingerprint の採取期間中にリリースされたが，どちらも 1 つ前のバージョンとの識別は出来なかった．これらの結果から，ブラウザの機能の差異を使用することで，既知のバージョンはある程度広い範囲で推定が可能であることが分かった．さらに，Math オブジェクトと組み合わせた場合は 23 個のグループに分類された．提案手法では主にブラウザの種類とバージョン，Math オブジェクトによる手法では特定の OS とブラウザの分類が可能であったため，これらを組み合わせることで，推定可能なプラットフォームの種類が増えた．

6.2 実験 2

提案手法による特徴点を使用して Fingerprinting を行い，User Agent や Math オブジェクトをそれぞれ使用した場合と比べてほとんど変わらぬ識別精度を保つことができた．ここで，実験 2 で使用したサンプルを各特徴点で分類した場合のグループ数は，表 4 に示した通り，特徴点ごとに 2 倍以上の差があった．しかし，図 2 の通り Fingerprinting の精度差はごく僅かであった．つまり，今回採取した期間や

サンプル数では有用であるといえるが，より長期間，より多くのサンプルでの識別精度も検証する必要がある．

6.3 課題

提案手法により推定可能なブラウザのバージョンは，特徴点として使用する機能に依存する．そのため，既知のバージョンについては広い範囲で識別可能だが，先に述べた Chrome 71 や Firefox 64 のように，新たにリリースされたバージョンについては識別できない可能性がある．このことから，プラットフォームの推定を長期間に渡って行う場合は，提案手法では新バージョンのリリースのたびに対応が必要になるといえる．また，本実験のサンプルでは Fingerprinting の精度を保つことができた．しかし，未知のバージョンが識別できない場合，より長期的な Fingerprinting においては精度の低下が起こる可能性がある．対策として，選定に使用したリポジトリにおいて，各ブラウザの新たなバージョンについて更新または追加が行われた機能を定期的に抽出し，提案手法の特徴点に追加するという方法が考えられる．また，他の対策として，ECMAScript において Candidate または Finished の段階[15]にある機能を提案手法の特徴点にあらかじめ追加しておくという方法が考えられる．以上の方法により，未知のバージョンのリリース時に新しいバージョンの識別に対応し，Fingerprinting 精度を保つことができる可能性がある．

前節で述べたように，User Agent，Math オブジェクト，提案手法による分類数の差に対し，これらを使用した Fingerprinting の精度の差は小さかった．さらに，各精度は図2に示したように98.9%以上という高い値を示している．また，2 章で挙げた研究も含め，Fingerprinting に関する既存の研究でも 90%以上の高い精度が出ているものが多く，Fingerprinting 技術は既に高いレベルに達していると考える．しかし，これらの Fingerprinting は現在一般的なマシン上で動くブラウザに対して行われたものである．そこで今後はクラウド上の仮想ブラウザに対して既存の Fingerprinting 技術が有効であるかどうかの検証が必要であると考える．

7. まとめ

本論文では，ブラウザによる機能の実装の差異を利用することで，User Agent を用いることなく，プラットフォームの推定が行えることを示した．また，ブラウザによる機能の実装の差異を User Agent の代わりに特徴点として使用することで，精度をほとんど低下させることなく Fingerprinting が行えることが分かった．しかし，本論文の提案手法は使用する機能に依存し，ブラウザの未知のバージョンに対応できない可能性がある．その結果，より長期間に渡って Fingerprinting を行なった場合は精度が低下してしまうことが考えられる．

今後は，より長期間の Fingerprinting における精度の検証が必要である．

参考文献

[1] 磯侑斗，桐生直輝，塚本耕司，高須航，山田智隆，武居直樹，齋藤孝道，2014，”Web Browser Fingerprint を採取する Web サイトの構築と採集データの分析”，コンピュータセキュリティシンポジウム 2014 論文集 p.378-p.385
[2] “Release Notes for Safari Technology Preview 46 | WebKit”, https://webkit.org/blog/8042/release-notes-for-safari-technology-preview-46/
[3] “Safari 11.1”, https://developer.apple.com/library/archive/releasenotes/General/WhatsNewInSafari/Articles/Safari_11_1.html
[4] “RIG エクスプロイトキット 解析レポート - NTT Security”, https://www.nttsecurity.com/docs/librariesprovider3/default-document-library/jp-rigek-analysis-report
[5] P. Eckersley, “How Unique is Your Web Browser?”, *Proc. Privacy Enhancing Technologies Symposium (2010)*, LNCS vol. 6205
[6] 野田隆文，安田昂樹，高橋和司，種岡優幸，田邉一寿，細谷竜平，齋藤祐太，小芝力太，齋藤孝道，2018，”JavaScript の Math オブジェクトによるプラットフォーム推定”，2018 年暗号と情報セキュリティシンポジウム（SCIS2018）
[7] “Standard ECMA-262”, http://www.ecma-international.org/publications/standards/Ecma-262.htm
[8] “JavaScript リファレンス | MDN”, https://developer.mozilla.org/ja/docs/Web/JavaScript/Reference
[9] “ユーザーエージェントの検出の回避 (User Agent を用いたブラウザーの判定 | MDN)”, https://developer.mozilla.org/ja/docs/Web/HTTP/Browser_detection_using_the_user_agent#Avoiding_user_agent_detection (参照 2018-12-10)
[10] “Chrome Releases: Stable updates”, https://chromereleases.googleblog.com/search/label/Stable%20updates
[11] “Mozilla Firefox Web Browser — Mozilla Firefox Release Notes — Mozilla”, https://www.mozilla.org/en-US/firefox/releases/
[12] “mdn/browser-compat-data: This repository contains compatibility data for Web technologies as displayed on MDN”, https://github.com/mdn/browser-compat-data
[13] “Math | MDN”, https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math
[14] “Valve/fingerprintjs2: Modern & flexible browser fingerprinting library”, https://github.com/Valve/fingerprintjs2
[15] “The TC39 Process”, https://tc39.github.io/process-document/