Vol.2020-CSEC-91 No.8
Vol.2020-SPT-40 No.8
Vol.2020-EIP-90 No.8
2020/11/25

ブラウザーフィンガープリティングを用いた VM上のブラウザによるアクセス識別の試み

神 章洋[1,a)]　利光 能直[†1]　鈴木 嵩人[†1]　北條 大和[†1]　齋藤 孝道[1]

概要：本論文では，仮想マシン上のブラウザからアクセスを，物理 PC からのそれと区別すること（以降，VM ブラウザアクセス識別という）を，ブラウザーフィンガープリティング技術を用いて試みた．これまで，マルウェア（ネイティブアプリケーション）においては，当該マルウェアの稼働が，仮想マシン上での稼働であるか否かを検知する手法が様々提案されている．しかし，マルウェアで使う手法においては，ネイティブアプリケーションでのみ取得できる情報を利用しており，ブラウザでは取得できないものが多く，仮想マシン上のブラウザからアクセスなのか，物理 PC からのそれなのかをサーバサイドで識別することは一般に困難である．そこで，本論文では，ブラウザーフィンガープリティング技術を用いて，Web サーバ側で，VM ブラウザアクセス識別を試み，検知に使用できる特徴点と，その特徴点を使用した検知手法の提案と，その結果を示す．

キーワード：ブラウザフィンガープリンティング, 仮想マシン検知, JavaScript

An Attempt to Identify Access by Browser on VM Using Browser Fingerprinting

Abstract: In this paper, we attempted to distinguish access from a browser on a virtual machine from that from a bare-metal PC (henceforth called VM browser access discrimination) using browser fingerprinting. There have been various methods for detecting whether or not the malware (native application) is running on a virtual machine. However, many of the methods used in malware use information that can only be obtained by native applications and cannot be obtained by a browser, so it is generally difficult to distinguish on the server side whether the access is from a browser on a virtual machine or from a bare metal PC. In this paper, we attempt to discriminate VM browser access on the web server side by using browser fingerprinting technique, and propose a detection method using the feature points that can be used for detection and the results.

Keywords: Browser Fingerprinting, Virtual Machine Detection, JavaScript

1. はじめに

本論文では，Web サーバにブラウザからアクセスされた際に，仮想マシン上のブラウザからのアクセスか，物理 PC 上のブラウザからのアクセスかを識別すること（VM ブラウザアクセス識別）を目的としている．

マルウェアを動的解析する際，仮想マシン上で行うことが一般的である．そこでマルウェアは実行環境が仮想マシンの場合は挙動を変えるなどして，解析を妨害する．仮想マシンを判別する技術は，こうした背景からマルウェアに実装するため，様々な手法が考案されてきた．マルウェアが行う仮想マシン検知の手法はネイティブアプリケーションでのみ取得できる情報を利用しており，ブラウザでは取得できないものが多く，仮想マシン上ブラウザからのアクセスなのか，物理 PC 上ブラウザからのアクセスなのかをサーバサイドで識別することは一般に困難である．しかし，ブラウザフィンガープリンティング技術が VM ブラウザアクセス識別に利用できる可能性がある．

[1] 明治大学
Meiji University
[†1] 現在，明治大学大学院
Presently with Graduate School of Meiji University
[a)] ee177041@meiji.ac.jp

Vol.2020-CSEC-91 No.8
Vol.2020-SPT-40 No.8
Vol.2020-EIP-90 No.8
2020/11/25

ブラウザフィンガープリンティング（以降，フィンガープリンティングと呼ぶ）は，ブラウザから取得できる複数の情報を組み合わせ，端末のブラウザの識別を行う技術である．この技術はユーザの行動追跡やリスクベース認証などを目的として使用されている．

本論文では，仮想マシン上ブラウザと物理 PC 上ブラウザのフィンガープリントを調査し，フィンガープリンティングを利用した VM ブラウザアクセス識別の手法と識別結果についてまとめた．

2. 関連知識

2.1 ブラウザフィンガープリンティング

ブラウザフィンガープリンティングは端末のブラウザから採取した情報の組み合わせによって，端末上のブラウザを識別する技術である．また，ブラウザフィンガープリンティングでブラウザから取得する情報を特徴点と呼び，特徴点の値や組み合わせをブラウザフィンガープリントという．

Eckersley[1] はフィンガープリントを採取するサイトを構築し，フィンガープリントの分析をした結果，採取したフィンガープリントの 83.6%がユニークであり，Flash や Java が実行できる端末に限定した場合，94.2%がユニークであることを示した．

Mowery ら [2] は，Canvas API を用いて端末内のブラウザ上に複雑な文字や図形を描写した結果が GPU，ブラウザのバージョン，及び OS の組み合わせによって異なることを示した．また，この調査の中で採取した 294 個のサンプルにおいて，116 個がユニークであり，この結果をハッシュ化して保存することで，フィンガープリンティング に応用できることを示した．そのときのエントロピーは 5.73 ビットであり，98.2%の確率でブラウザを正しく識別できるとしている．また WebGL API を用いて GPU やモデル名が採取可能であることを示した．

田邉ら [3] はフィンガープリンティングにおいて，特徴点の最良の組み合わせを分析した．その結果，識別精度を向上させる特徴点と，反対に低下させる特徴点を明らかにした．

2.2 仮想マシン検知

仮想マシン検知は主にマルウェアが実装する技術の一つで，当該マルウェアの稼働が仮想マシン上での稼働であるかを検知する手法である．これは，マルウェアを仮想マシン上で動作させて挙動を解析する動的解析を回避するために実装される．実行環境の様々な情報を収集し，仮想マシンの特徴を見つけることで仮想マシンの検知を行う．

岩本らの研究 [4] では，文献調査とマルウェアの静的解析を行い，マルウェアに実装される仮想マシン検知の手法について調査している．この論文では 33 種類の検知手法が紹介されている．また，解析したマルウェアの仮想マシン検知機能の実装状況をまとめている．この論文に書かれている仮想マシン検知の手法をいくつか紹介する．

- プロセスを列挙して仮想マシン特有のプロセスを発見した場合に仮想マシンと判断する
- CreateFile または GetFileAttributes, PathFileExists を呼び出して特定のファイルが存在するか確認することで仮想マシンを検知する
- MAC アドレスのプレフィックスが VMware や Virtual Box に固有の値のときに仮想マシンと判断する

この研究で紹介されている手法はネイティブアプリケーションが利用できる情報を用いている．

Chen らの研究 [5] では，16,246 の一般的なマルウェアと 1,037 の標的型攻撃に利用されるマルウェアを調査し，ウイルス対策による検出率，デバッグ防止および仮想マシン検知技術の実装状況の違いについてまとめている．その結果，標的型攻撃に利用されるマルウェアは一般的なマルウェアよりデバッグ防止および仮想マシン対策技術を使用していないと結論付けた．

2.3 VM ブラウザアクセス識別

本論文では，仮想マシン上のブラウザからのアクセスを，物理 PC 上のブラウザからのアクセスと区別することを「VM ブラウザアクセス識別」と呼ぶ．Web サーバにブラウザからのアクセスがあったとき，そのブラウザが物理 PC 上で動作しているか，仮想マシン上で動作しているかをブラウザフィンガープリンティングを用いて判別することが目的である．概要を図 1 に示す．

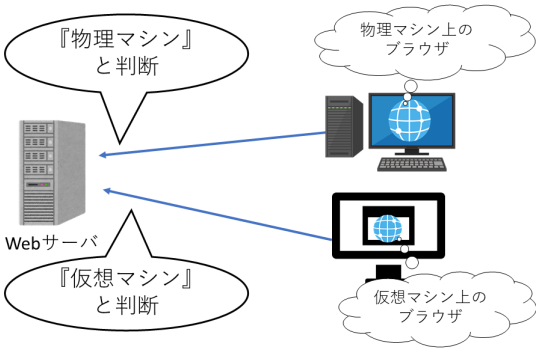

図 1 VM ブラウザアクセス識別の概要

3. 実験データ

3.1 フィンガープリント採取サイト

先行研究 [6] でフィンガープリントを採取する Web サイト（以降，フィンガープリント採取サイトと呼ぶ）が制作された．本論文ではフィンガープリント採取サイトを利用してブラウザのフィンガープリントを採取した．先頭ページに利用者がコメントを入力する項目があり，フィンガー

プリントとコメントの両方がデータベースに保存される．なお，フィンガープリント採取サイトは利用者の同意の上でフィンガープリントを採取している．

3.2 実験に使用したサンプル

物理 PC 上のブラウザと仮想マシン上のブラウザからフィンガープリント採取サイトにアクセスすることでサンプルの採取を行った．実験に使用したブラウザではフィンガープリンティングの対策技術やトラッキングの対策技術は適用していない．また，Tor Browser のようなフィンガープリンティング対策機能を持つブラウザも本論文では対象外である．実験に使用した仮想化ソフトは VirtualBox[7] と VMware Workstation Player[8] である．サンプルの収集は研究室内で協力者を募った．その際，協力者にはどちらの環境からのアクセスかをフィンガープリント採取サイトのコメント入力欄に入力してもらい，それを識別の際の正解データとした．以降，本論文用に集めたデータを「サンプルデータ」と呼ぶ．

今回，本論文用に集めたフィンガープリントのサンプルデータ数は 274 である．表 1 に環境ごとのデータ数をまとめた．

4. 実験について

4.1 実験概要

本論文では，ブラウザフィンガープリンティングを用いて VM ブラウザアクセス識別が可能か実験した．実験では以下の点について検証した．

- 物理 PC 上ブラウザと仮想マシン上のブラウザで差が出る特徴点の調査
- 個々の特徴点のみで判別した場合の精度評価
- 複数の特徴点を使用して判別した場合の精度評価

4.2 実験の詳細

本論文で行う 2 つの実験の詳細を説明する．

4.2.1 実験 1 の詳細

まず，採取したフィンガープリントを調査し，特徴点の中から物理 PC 上ブラウザと仮想マシン上ブラウザで差が出る特徴点の候補を見つけた．次に，それらの特徴点を使用し，特定の条件に当てはまったときに仮想マシン上ブラウザからのアクセスと判断するルール（以降，識別ルールと呼ぶ）を作成した．識別ルールの詳細については 5.1 で説明する．作成した識別ルールで表 1 のデータに対し識別を行い，個々の識別ルールごとに精度評価を行うことで，識別に有効な特徴点を特定した．識別精度の評価には，Precision, Recall, Accuracy, F1 値を使用した．

以下に，今回の VM ブラウザアクセス識別に合わせてそれぞれの評価について説明する．

- Precision：仮想マシン上ブラウザからのアクセスと予測したデータのうち，実際に仮想マシン上ブラウザからのアクセスであるデータの割合
- Recall：実際に仮想マシン上ブラウザからのアクセスであるデータのうち，仮想マシン上ブラウザからのアクセスであると予測されたものの割合
- Accuracy：仮想マシン上ブラウザからのアクセスと物理 PC 上ブラウザからのアクセスと予測したデータのうち，実際に仮想マシン上ブラウザからのアクセスと物理 PC 上ブラウザからのアクセスの割合
- F1 値：Precision と Recall の調和平均

以下に，Precision, Recall, Accuracy, F1 値を求める式を示す．

$$Precision = \frac{TP}{TP+FP}$$
$$Recall = \frac{TP}{TP+FN}$$
$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$$
$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

上記の識別精度の算出には表 2 に基づき，TP, TN, FP, FN を算出した．

表 2 TP,TN,FP,FN の分類

		判別予想	
		仮想マシン	物理 PC
実際の識別	仮想マシン	TP	FN
	物理 PC	FP	TN

4.2.2 実験 2 の詳細

実験 2 ではまず，実験 1 で調査した識別ルールを全て使用して表 1 のサンプルデータに対し識別を行い，精度評価を行った．識別精度の評価は実験 1 と同じ方法を使用した．

実験 2 では，仮想マシン上ブラウザからのアクセスの特徴にどのくらい当てはまるかを表す値（以降，スコアと呼ぶ）を使用した．スコアは初期値が 0 で，実験 1 で作成した識別ルールに当てはまるたびに加算される．スコアが特定の値を超えたデータを仮想マシン上ブラウザからのアクセスと判断した．以降，仮想マシン上ブラウザからのアクセスと判断するスコアの境目を閾値と呼ぶ．この方法ではスコアが高いほど仮想マシン上ブラウザからアクセスした際のデータである可能性が高いと言える．

Listing1 に，仮想マシン環境の特徴に当てはまる際のスコアの加算をするコードの例を紹介する．これは，画面解像度が 800×600 のときに仮想マシン上ブラウザからのアクセスと判断する識別ルールのコードを一部抜粋したものである．ここではスコアを表す変数を”vmscore”としている．

Vol.2020-CSEC-91 No.8
Vol.2020-SPT-40 No.8
Vol.2020-EIP-90 No.8
2020/11/25

表 1　サンプルデータ数の詳細

環境	物理 PC				仮想マシン					
仮想化ソフト名					Virtual Box			VMware		
OS / ブラウザ	Windows	Mac OS	Linux	不明	Windows	Mac OS	Linux	Windows	Linux	不明
Chrome	45	23	1	0	0	0	6	0	2	0
Firefox	22	7	7	0	5	1	41	0	22	0
Edge	43	0	0	0	1	0	0	2	0	0
IE	18	0	0	0	2	0	0	3	0	0
Safari	0	10	0	0	1	0	0	0	0	0
Others	3	0	0	8	0	0	0	0	0	1
合計	187				87					

```python
#仮想マシン環境の特徴に当てはまるか確認
#(以下は画面解像度を確認する際のコード)
if scList[0] == 800 and scList[1] == 600:
    #当てはまった場合にスコアを加算
    vmscore += 2
```

Listing 1: 識別方法の例 (Python)

スコアに加算する値について，実験 1 の識別ルールごとの精度評価で Precision が 0.9 を超えた識別ルールに当てはまる場合はスコアに 2 加算し，それ以外の識別ルールに当てはまる場合はスコアに 1 加算するようにした．Precision の値が高い識別ルールは物理 PC 上ブラウザではあまり見られない仮想マシン上ブラウザの特徴を利用していると考えられるため，Precision を基準にスコアに加算する値を決定した．閾値は 1 から，スコアがなり得る最大の値（以降，最大スコアと呼ぶ）の間で設定した．それぞれの閾値で精度評価を行うことで，最も精度評価が良くなる閾値を求めた．

例として，最大スコアが 14 の場合は Listing2 のように閾値を変動させる．

```python
#閾値を 1 から 14 の間で設定
for i in range(1, 15):
    #vm_pred に識別結果を保存
    vm_pred = []
    for index, row in df.iterrows():
        #row['vmscore'] に各データのスコアが入っている
        #スコアが閾値以上なら 1，それ以外は 0
        #1:仮想マシンと判断
        #0:物理 PC と判断
        vm_pred.append(1 if row['vmscore'] >= i else
        ↪  0)
    #以下に識別結果を表示させるコードが続く
```

Listing 2: 閾値変更の例 (Python)

次に，作成した識別ルールから，精度評価の結果が良くない特徴点や，調査の結果，識別にあまり関係がないと考えられる特徴点を取り除き，同じように精度評価を行った．これにより，除外した識別ルールが無くても識別が可能か，精度はどう変化するかを確認した．

5. 実験 1 の結果

5.1 物理 PC 上ブラウザと仮想マシン上ブラウザで差が出る特徴点

採取したデータを調査し，取得している特徴点の中から仮想マシン上ブラウザと物理 PC 上ブラウザで差が出る特徴点の候補をいくつか見つけた．以下でそれらの特徴点について説明する．

5.1.1 画面解像度

JavaScript の Screen オブジェクトを使用することでディスプレイの画面解像度を取得することができる．これは横幅×縦幅で表される． VirtualBox と VMware Workstaion Player で仮想マシンを起動した際のデフォルトの画面解像度は 800 × 600 である．また，仮想マシンは自在に画面サイズを変更できるため，中途半端な値になる場合がある．

5.1.2 論理プロセッサ数

JavaScript の window.navigator.hardwareConcurrency メソッドを利用することでコンピュータの論理プロセッサ数を取得することができる．デフォルトでは，VirtualBox では 1，VMware Workstaion Player では 2 で設定される．

5.1.3 メディアデバイス情報

JavaScript の MediaDevices.enumeratedDevices メソッドで，システム上で利用できるカメラ，オーディオ，マイクの情報を取得できる．VirtualBox や VMware Workstaion Player ではデフォルト設定でカメラが認識されていない．

5.1.4 WebGL 情報

WebGL はブラウザ内で 3D，2D グラフィックスをレンダリングするための JavaScript API である．WebGLRenderingContext.getExtension メソッドの拡張機能である WEBGL_debug_renderer_info を使用するとグラフィックスドライバのベンダ情報，レンダラ情報を取得できる．このベンダ情報が，仮想マシンでは'VMware, Inc' になる場合が多い．また，ベンダ情報が'VMware, Inc' 以外のもの

Vol.2020-CSEC-91 No.8
Vol.2020-SPT-40 No.8
Vol.2020-EIP-90 No.8
2020/11/25

では，レンダラ情報が'Google SwiftShader' か'Microsoft Basic Render Driver' になるものがあった．

5.1.5 メモリサイズ

JavaScript の navigator.deviceMemory を利用することでデバイスのメモリサイズを取得できる．仮想マシン環境ではホストマシンのメモリの一部を割り当てるためメモリが小さく設定される傾向がある．

5.1.6 Web worker

先行研究 [9] にて，Web Workers で並列処理を行い，CPU の処理性能の差を利用して，利用者のデバイスの CPU コア数を推定する手法が提案された．本論文では，先行研究 [9] と同じ JavaScript のコードで 16 個のスレッドを動作させ，完了までの処理時間を計測した．各スレッドでは for 文で大量の繰り返し処理を行うことで CPU に負荷をかけた．

```
onmessage = function(e) {
    var i, j;
    for (i = 0; i < 50000; i++) {
        for (j = 0; j < 2500; j++)
            ;
    }
    postMessage('');
};
```

Listing 3: Web worker で実行させる処理 (JavaScript)

仮想マシン環境では物理 PC に比べて処理時間が長くなる傾向がある．

5.1.7 p0f による推定値

TCP のパケットから OS の種類などを推定するパッシブ OS フィンガープリンティングという手法がある [10]．仮想マシン環境の場合，p0f で推定できる OS と，ユーザエージェントで推定できる OS が異なる場合がある．例えば，Windows 上で Linux の仮想マシンを起動し，仮想マシンの Linux 上のブラウザからユーザエージェントとパッシブ OS フィンガープリンティングで OS の推定を行った場合，ユーザエージェントでは Linux と推定できるが，パッシブ OS フィンガープリンティングでは Windows と推定される．

5.1.8 デバイスピクセル比

JavaScript の window.devicePixelRatio を用いることでデバイスピクセル比を取得できる．これは，ブラウザで表示する画像の 1 ピクセルを，端末のディスプレイ上で何ピクセルとしてレンダリングを行うかを表した比率である．ブラウザで表示するピクセルを「css ピクセル」と言い，ディスプレイ上のピクセルを「デバイスピクセル」と言う．仮想マシン環境のデータでは css ピクセル:デバイスピクセルの比率が 1:1 になる傾向がある．

5.1.9 リフレッシュレート

1 秒間に画面に描画される回数を表す．例えば画面が 1 秒間に 100 回描画された場合は 100Hz と表す．JavaScript の Animation Timing API で定義される requestAnimationFrame は，ディスプレイが描画されるタイミングで実行されるため，このメソッドの処理時間を計測することでリフレッシュレートを求めることができる．一般的にリフレッシュレートは 60Hz 前後だが，仮想マシン環境だと小さくなったり，大きくなっているデータをいくつか確認した．

5.2 識別ルールの作成

特徴点の調査を基に実験 2 で使用する識別ルールを作成した．作成した識別ルールを表 3 にまとめる．それぞれの識別ルールに番号を付け，以降はその番号で識別ルールを示す（以降，ルール番号と呼ぶ）．「仮想マシンの識別基準」には，仮想マシン上ブラウザからのアクセスと判断する条件を記述してある．

表 3 仮想マシン環境の識別ルール

ルール番号	特徴点	仮想マシンと判断する条件
1	画面解像度	画面解像度の横幅が 800 で，縦幅が 600 の場合
2	画面解像度	画面解像度の横幅か縦幅が 10 で割り切れない値の場合
3	論理プロセッサ数	論理プロセッサ数が 2 以下の場合
4	メディアデバイス情報	カメラが認識されない場合
5	WebGL 情報	ベンダ情報が'VMware, Inc.' だった場合
6	WebGL 情報	レンダラ情報が'Google SwiftShader' か'Microsoft Basic Render Driver' だった場合
7	メモリサイズ	メモリサイズが 2GB 以下の場合
8	Web worker	ブラウザが Firefox：処理時間が 1000 ミリ秒を超えた場合 ブラウザが Firefox 以外：処理時間が 700 ミリ秒を超えた場合
9	p0f	ユーザエージェントと Passive OS Fingerprinting で OS の推定を行い，推定した OS の種類が違った場合
10	デバイスピクセル比	デバイスピクセル比が 1:1 だった場合
11	リフレッシュレート	リフレッシュレートが 59 未満, あるいは 62 より大きかった場合

5.3 識別ルールごとの識別結果

作成した表 3 の識別ルールをサンプルデータに対して個々に使用して識別を行った精度評価を表 4 に示す．表 3 と表 4 のルール番号は対応している．

表 4 ルールごとの識別結果

ルール番号	Precision	Recall	Accuracy	F1
1	0.9464	0.6091	0.8649	0.7412
2	0.3424	0.2873	0.5985	0.3124
3	0.9714	0.7816	0.9233	0.8662
4	0.8453	0.9425	0.9270	0.8913
5	0.9206	0.6666	0.8759	0.7733
6	0.7777	0.1609	0.7189	0.2666
7	1.0000	0.0574	0.7007	0.1086
8	0.6969	0.7931	0.8248	0.7419
9	1.0000	0.8965	0.9671	0.9454
10	0.6296	0.9770	0.8102	0.7657
11	0.6896	0.4597	0.7627	0.5517

6. 実験 2 の結果

6.1 全ての識別ルールを使用した結果

実験 1 の表 3 の識別ルールを全て使ってサンプルデータに対して識別を行った．4.2.2 で説明した通り，表 4 で，Precision が 0.9 を超えている識別ルールに当てはまる場合はスコアを 2 加算し，それ以外のルールに当てはまる場合はスコアを 1 加算した．表 6.1 に識別ルールごとに，スコアに加算される値をまとめる．この表のルール番号は表 3 のルール番号と対応している．

表 5 識別ルールごとのスコアに加算される値

ルール番号	スコアに加算される値
1	2
2	1
3	2
4	1
5	2
6	1
7	2
8	1
9	2
10	1
11	1

ルール番号 1,2 とルール番号 5,6 はそれぞれ同時に当てはまることは無いため，最高スコアは 14 である．閾値を 1 から 14 に設定して識別した結果を表 6 にまとめた．Accuracy と F1 値が最も高くなったのは閾値を 4 に設定したときで，Accuracy が 0.9781，F1 値が 0.9662 となった．

表 6 全識別ルールを用いた識別結果

閾値	Precision	Recall	Accuracy	F1
1	0.4123	1.0000	0.5474	0.5838
2	0.7190	1.0000	0.8759	0.8365
3	0.8969	1.0000	0.9635	0.9456
4	0.9450	0.9885	0.9781	0.9662
5	0.9438	0.9655	0.9708	0.9545
6	0.9651	0.9540	0.9744	0.9595
7	0.9871	0.8850	0.9598	0.9333
8	0.9864	0.8390	0.9452	0.9068
9	0.9838	0.7011	0.9014	0.8187
10	0.9818	0.6206	0.8759	0.7605
11	1.0000	0.5057	0.8430	0.6717
12	1.0000	0.1839	0.7408	0.3106
13	0.0000	0.0000	0.6824	0.0000
14	0.0000	0.0000	0.6824	0.0000

6.2 一部の識別ルールを取り除いた結果

表 3 の識別ルールのうち，いくつかを取り除いて識別を行った．

まず，ルール番号 9 の識別ルールを除外した．p0f は通信の間にファイアウォールなどがあるとそれらの情報を取得してしまう可能性があり，アクセスする環境によっては誤判定の原因になる可能性が高いためである．

次に，ルール番号 10 の識別ルールを除外した．除外した理由は OS に Linux を使用している物理 PC 上のブラウザからフィンガープリントを収集した結果，デバイスピクセル比が 1:1 になったことからこの特徴点の差は物理 PC と仮想マシンの違いではなく，使用している OS の違いによる可能性があるためである．

次に，ルール番号 2 の識別ルールとルール番号 11 の識別ルールを除外した．除外した理由は F1 値が低く，識別に有効ではないと考えられるからである．

ルール番号 6 と 7 の識別ルールに関しては，F1 値が低いが，これは低い原因がわかっているために残して識別した．F1 値が低い原因は，ルール番号 6 については 7.1.5 で，ルール番号 7 については 7.1.7 で述べる．

表 7 に，上記のルール番号 9，10，11 の 3 つの識別ルールを除外して識別を行った結果の精度評価をまとめた．Accuracy と F1 値が最も高いのは閾値を 2 に設定したときで，Accuracy が 0.9744，F1 値が 0.9608 になった．

7. 考察

7.1 実験 1 について

表 4 の識別結果について，識別ルールごとに考察する．

7.1.1 ルール番号 1 の識別ルール

この識別ルールについて，Precision が高いのは画面解像度が 800 × 600 の物理 PC 環境は現在ではほぼ使われていないからだと考えられる．サンプルデータでは，この識別

表 7　一部のルールを除外した識別結果

閾値	Precision	Recall	Accuracy	F1
1	0.6397	1.0000	0.8211	0.7802
2	0.9347	0.9885	0.9744	0.9608
3	0.9418	0.9310	0.9598	0.9364
4	0.9629	0.8965	0.9562	0.9285
5	0.9565	0.7586	0.9124	0.8461
6	0.9827	0.6551	0.8868	0.7862
7	0.9772	0.4942	0.8357	0.6564
8	0.9714	0.3908	0.8029	0.5573
9	1.0000	0.0114	0.6861	0.0227
10	0.0000	0.0000	0.6824	0.0000

ルールに当てはまった物理 PC 上ブラウザのデータは 187 件のうち 3 件のみだった．

一方，Recall が低いのは，サンプルデータには仮想マシンの画面解像度を変更したものが含まれていたためだと考えられる．この識別ルールに当てはまった仮想マシン上ブラウザは 87 件のうち 53 件だった．

7.1.2　ルール番号 2 の識別ルール

この識別ルールは Precision も Recall も低めである．今回の結果から画面解像度の横幅，縦幅の値が 10 で割り切れない値になることは識別にはあまり有効でないと判断する．

7.1.3　ルール番号 3 の識別ルール

この識別ルールについて，Precision が高いのは，現在の物理 PC では論理プロセッサ数が 2 以下の環境は少ないからだと考えられる．サンプルデータの物理 PC 上ブラウザのデータには 187 件中 2 件のみ，論理プロセッサ数が 2 以下だった．Recall が低いのは，仮想マシンの設定で論理プロセッサ数を変えることが簡単にできるため，今回使用したサンプルデータの中に論理プロセッサ数を多く設定していたものが含まれていたからだと考えられる．

7.1.4　ルール番号 4 の識別ルール

この識別ルールについて，Precision が低いのは，物理 PC でもカメラを搭載していないデバイスが存在するためだと考えられる．Recall が高いのは，VirtualBox や VMware Workstation Player のデフォルト設定ではカメラを認識しないようになっていることが影響していると考えられる．

7.1.5　ルール番号 5 の識別ルール

この識別ルールについて，Precision が高いのは，仮想マシン環境でのグラフィックドライバのデフォルト設定に物理 PC との差が出るからだと考えられる．

7.1.6　ルール番号 6 の識別ルール

これはルール番号 5 の識別ルールに当てはまらない場合に満たす可能性のある識別ルールである．サンプルデータにルール番号 5 の識別ルールを満たす仮想マシン環境のデータが多いため，Recall と F1 値が低くなっていると考えられる．

7.1.7　ルール番号 7 の識別ルール

この識別ルールについて，Precision が高い理由は，仮想マシン環境ではホストマシンのメモリの一部を割り当てるためメモリが小さく設定される傾向があるためだと考えられる．Recall と F1 値が小さい理由は，Firefox でメモリサイズが取得できないからと考えられる．表 1 より，サンプルデータの仮想マシン上ブラウザの 87 件のデータのうち，69 件のデータは Firefox からのアクセスである．

7.1.8　ルール番号 8 の識別ルール

この識別ルールは Precision も Recall も約 0.7 である．仮想マシン環境では物理 PC に比べて処理時間が長くなる傾向があるが，このルールはコンピュータの性能に依存するために Precision と Recall が約 0.7 になったと考えられる．

7.1.9　ルール番号 9 の識別ルール

この識別ルールは Precision も Recall も高い．これは，今回集めたデータについて，仮想マシンの OS とホスト OS が異なる環境のものが多かったからだと考えられる．しかし

7.1.10　ルール番号 10 の識別ルール

この識別ルールは Precision が低く，Recall が高い．しかし，物理 PC の Linux のデバイスピクセル比を調べたところ，css ピクセル:デバイスピクセルの比率が 1:1 になったため，デバイスピクセル比は仮想マシン環境ではなく OS によって異なると考えられる．Recall が高い理由は，表 1 より，採取した仮想マシン上ブラウザのデータの OS に Linux を使っていることが多いからだと考えられる．よって，この識別ルールは識別にはあまり有効でないと判断する．

7.1.11　ルール番号 11 の識別ルール

この識別ルールは Precision も Recall も低めである．今回の結果からリフレッシュレートの値が 60Hz から離れることは識別にはあまり有効でないと判断する．

7.2　実験 2 について

実験 2 の精度評価の結果について考察する．

7.2.1　全ての識別ルールを使用した識別について

表 6 の結果について考察する．まず，閾値を 1 から 12 まで変化させた場合，閾値が高くなるにつれて Precision は高くなり，Recall は低くなる傾向にあることが判明した．

閾値を 13 以上に設定すると Precision も Recall も 0 になった．これは，今回のデータではスコアが 13 以上になる仮想マシン環境がなかったためである．また，Accuracy が約 0.7 なのは，全てのデータを物理 PC と判断した際，表 1 から，物理 PC 上ブラウザと仮想マシン上ブラウザのデータ数に偏りがあるためだと考えられる．

7.2.2　一部のルールを除外した識別について

一部のルールを除外して識別を行ったところ表 7 の結果

になった．全てのルールを使用した結果である表 6 と比べると Accuracy と F1 値の値は少し下がったが，ほぼ同じくらいの精度評価であることがわかった．よって，ルール番号 2,9,10,11 の識別ルールを使用は識別にあまり影響しないと考えられる．

8. 今後の課題

現在の問題点と今後の研究課題について述べる．

8.1 対策のしやすさ

従来の，ネイティブアプリケーションが行う仮想マシン検知技術では仮想マシン上で動いているプロセスや MAC アドレスなどを使用して検知しているのに対し，本論文では，論理プロセッサ数や画面解像度など，仮想マシンに多い設定を利用することで物理 PC 上のブラウザと仮想マシン上のブラウザを識別している．今回の識別方法の場合，仮想マシンの設定を変えることで，従来の仮想マシン検知に比べて容易に回避することができる．そのため，他の識別に使える特徴点の調査をする必要がある．

8.2 実験データの課題

今回の実験では研究室内で協力者を募りデータを収集したため，限られた環境のデータが多く，データ数も少ない．また，OS やブラウザなどでデータに偏りがある．収集範囲を広げた場合，今回の実験と異なる結果が出る可能性がある．

8.3 モバイルエミュレータの検知

今回は仮想マシン上のブラウザからのアクセスと物理 PC からのアクセスとを区別するという実験を行ったが，仮想マシンは PC だけではなくモバイル端末を PC 上でエミュレートする場合にも用いられる．しかし，今回の実験と同様にブラウザ上から得られる特徴点を利用したルールベースの識別方法で本物のモバイル端末と PC 上でエミュレートされたモバイル端末を識別できるかは現状ではわかっていない．そのため，今後はこれらのデータも収集し識別可能かの調査を行う必要がある．

9. 研究倫理

我々は，Menlo report[12] の精神に則り，倫理的配慮をして実験を行った．実験を行う際，機微な情報は扱わず，倫理的な問題はないことを確認した．

10. まとめ

本論文では，ブラウザフィンガープリンティングを利用して VM ブラウザアクセス識別を試みた．

まず，物理 PC 上ブラウザと仮想マシン上ブラウザのフィンガープリントを調査することで，識別に利用できる特徴点について調査しまとめた．また，その特徴点を利用した識別ルールを作成した．

次に，識別ルールを使用して実際に識別を行い，その結果の精度評価をまとめた．今回利用したデータでは Accuracy と F1 値で約 0.95 の精度で識別が可能であった．

参考文献

[1] Peter Eckersley, ”How Unique Is Your Web Browser?”, 10th International Symposium on Privacy Enhancing Technologies (PETS’10), 2010.
[2] Keaton Mowery, Hovav Schacham, “Pixel Perfect: Fingerprinting Canvas in HTML5”, Web 2.0 Security and Privacy (W2SP), 2012
[3] 田邉一寿, 高橋和司, 安田昂樹, 種岡優幸, 細谷竜平, 小芝力太, 齋藤祐太, 齋藤孝道, ” Browser Finger- printing における特徴の組み合わせに関する考察 ”, コンピュータセキュリティシンポジウム 2017 論文集 CD-ROM pp.1090-1097,2017
[4] 岩本一樹, 高田一樹, 津田侑, 遠峰隆史, 井上大介, ”マルウェアに実装されている仮想マシン検知機能の調査分析”, コンピュータセキュリティシンポジウム 2017 (CSS’17), 2017
[5] Ping Chen, Christophe Huygens, Lieven Desmet, Wouter Joosen, ”Advanced or Not? A Comparative Study of the Use of Anti-debugging and Anti-VM Techniques in Generic and Targeted Malware”, 31st IFIP TC 11 International Conference on ICT Systems Security and Privacy Protection (SEC 2016), 2016
[6] 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, ”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 (CSS’14), 2014
[7] Oracle VM VirtualBox (https://www.virtualbox.org/)
[8] VMware Workstation Player (https://www.vmware.com/jp/products/workstation-player/workstation-player-evaluation.html)
[9] 桐生直輝, 磯侑斗, 金子洋平, 齋藤孝道, ” Web Workers を用いた演算処理性能の差による CPU コア数の推定”, コンピュータセキュリティシンポジウム 2014 (CSS’14), 2014
[10] ”Passive OS Fingerprinting: Details and Techniques”, http://www.ouah.org/incosfingerp.htm (2020.10.20)
[11] ”リモートの p0f (passive fingerprinting) の結果を参照してスパム対策を行なう ”, http://www.gcd.org/blog/2008/01/150/ (2020.10.20)
[12] Dittrich, D. and Kenneally, E. The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research. U.S. Department of Homeland Security, Aug 2012.