Fingerprinting mobile devices: A short analysis
Alejandro Gómez-Boix, Pierre Laperdrix, Benoit Baudry

To cite this version:
Alejandro Gómez-Boix, Pierre Laperdrix, Benoit Baudry. Fingerprinting mobile devices: A short analysis. CIEL 2017 - 6ème Conférence en Ingénierie du Logiciel, Jun 2017, Montpellier, France. hal-01611101

HAL Id: hal-01611101
https://inria.hal.science/hal-01611101
Submitted on 5 Oct 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Fingerprinting mobile devices: A short analysis

Alejandro Gómez-Boix\textsuperscript{13}, Pierre Laperdrix\textsuperscript{23}, and Benoit Baudry\textsuperscript{3}

\textsuperscript{1} University of Rennes 1, Rennes, France
\texttt{alejandro.gomez-boix@inria.fr}

\textsuperscript{2} INSA-Rennes, Rennes, France
\texttt{pierre.laperdrix@insa-rennes.fr}

\textsuperscript{3} INRIA, Rennes, France
\texttt{benoit.baudry@inria.fr}

Abstract

The variety of possibilities for customizing web browser, combined with the capacity of a remote server to query information without requesting any permission, has created a new kind of privacy issue called browser fingerprinting. Previous studies demonstrated the effectiveness of fingerprinting to uniquely identify browsers running on laptops and desktops. Meanwhile, the majority of browsing activities have switched from laptops to mobile devices. Yet, there is currently a lack of systematic analysis of browser fingerprinting on mobile devices. In this paper, we describe browser fingerprinting technologies and summarize previous observations on laptops. Then, we present preliminary observations on mobile devices, highlighting differences with previous results, and challenges for future work.

Keywords: Browser fingerprinting, mobile devices, privacy

1 Introduction

The current web is the result of an open community that aims to define new technologies, such as HTML5, CSS3 and WebGL, guaranteeing every browser support them. To do that, web browsers have evolved along with web technologies, and have become a main tool for everyone entering in the virtual world of Internet. The design and architecture of current web browsers allows companies and open source communities to develop a wide number of software components, allowing users to personalize web browsers.

However, the variety of possibilities for customizing web browsers makes diversity one of the sources of privacy problems. This, combined with the information obtained without requesting any permission, arises in a new kind of privacy issue called browser fingerprinting. The term was introduced by Eckersley \cite{Eckersley2010} in 2010. Eckersley showed that when values for a set of components are collected, the combination of values obtained is mostly unique.

The increasing number of APIs make browsers richer and more powerful in execution platforms. Yet, they also open access to a growing variety of attributes that characterize the browser and their execution environment customizing browsers. Consequently, browser fingerprints become more and more precise and can be used to uniquely identify devices. Browser fingerprinting can hence be considered as a privacy side channel of some engineering decisions for web browser customization.

Most of the current Internet traffic comes from mobile devices, but there is a lack of systematic analysis of browser fingerprinting on these devices. Previous studies have focused on mobile and desktop fingerprints comparison, but to our knowledge there are not many studies focused exclusively on the analysis of fingerprinting on mobile devices.

Our work provides an extension to the analysis realized by Laperdrix et al. \cite{Laperdrix2014}. Our analysis takes into account data related only with mobile devices and in a more recent period of
time. The paper is organized as follows. Section 2 describes current fingerprinting techniques. Section 3 performs a short analysis on updated data of mobile devices and section 4 concludes this paper.

2 Fingerprinting techniques

Browser fingerprinting consists in collecting data regarding the configuration of a user’s browser and system when this user visits a website [5]. This identification technique is said to be “stateless” as it does not store any information on the user’s device. Nikiforakis and colleagues [9] demonstrated that web browsers can be identified in a consistent and measurable way, allowing scripts to discover them regardless of attempts to hide from scripts.

Eckersley demonstrated that browser fingerprinting is a powerful technique for user tracking. Through the Panopticlick website¹, fingerprints were collected from a sample of 470,161 browsers, and 83.6% of the browsers had an instantly unique fingerprint.

Later new browser fingerprinting techniques were developed, exploiting the latest JavaScript APIs. In 2011, using an interpreter based on timing and performance patterns, and acting only in the user agent header, Mowery et al. [6] detected browser version, operating system and microarchitecture, even when the user-agent header is modified or hidden. In the same year, the time zone, font set and the screen resolution were introduced to fingerprinting [2].

Using HTML5 for rendering browser font and WebGL scenes was introduced the Canvas element to fingerprinting [7]. This system fingerprint is not limited to browser properties only, through JavaScript by rendering an image, different hardware or software result in very small differences in the images. In 2014, Acar et al. [1] performed the first study of real-world canvas fingerprinting practices.

Because browser vendors frequently add new features, new versions are released with shorter time intervals. Mulazzani et al. [8] used tests that cover the ECMAScript standard to find functional differences between browsers. Their method was tested in browsers belonging to mobile and desktop devices.

Recent studies focused on providing deeper analyses and effective means to uniquely identify users through browser fingerprinting. Laperdrix et al. [5] offered a richer fingerprint with 17 attributes that uses the most recent web technologies and reveals not only browser configuration, but also the hardware and software environment it is running in. In Table 1, column ‘Attribute’ shows some attributes, the following attributes are not contained in the table: Cookies enabled, Use of local/session storage and Use of an ad blocker, because all of them have only two possible values: yes or no, and Do Not Track attribute has seven possible values. The Source column indicates the origin of each attribute (HTTP, JavaScript or Flash). Through the AmIUnique² website, 118,000 fingerprints were collected and analyzed. This study performed an analysis about browser fingerprinting on mobile devices (81% of the mobile fingerprints collected are unique) and assessed the evolution of fingerprints against technical changes, such as the disappearance of Flash, the end of browser plugins, among others.

In 2016 Wu et al. [10] introduced a fingerprinting method for Android device identification and presented 38 implicit identifiers that cover the features of physical layer, application layer, and user layer in Android system, without requesting any permission.

An extensive and very detailed measurement of online tracking was performed in [4]. The study was conducted based on a crawl of the top 1 million websites. In this study, the authors detected fingerprinting scripts utilizing AudioContext and related interfaces.

¹https://panopticlick.eff.org/
²https://amiunique.org/
### 3 Analysis on Mobile devices

Given the growth of mobile devices to browse the web, in conjunction with the different behavior of fingerprints on mobile devices and desktop machines, it is necessary to study both groups of fingerprints separately. We will focus our analysis on mobile device fingerprints collected on AmIUnique.org.

Discriminating attributes for mobile fingerprints are very different from those for desktop fingerprints. First, because web browsers on mobile devices were designed to take full advantage of HTML5 functionalities. For example, Adobe removed the Flash player from the Google Play store in August 2012 as part of a change of focus of the company. Additionally, plugins are not presented on mobile devices. Nevertheless, mobile fingerprints are recognizable [5].

To date AmIUnique.org counts more than 365,000 fingerprints, including desktop and mobile devices. Given the changing nature and speed with which technologies are updated, we offer an extension of the study realized by Laperdrix et al. [5] by performing an analysis of mobile fingerprints collected by AmIUnique.org from July 1st, 2016 to March 12th, 2017. In this period 17,370 fingerprints belonging to mobile devices were collected.

Using all attributes proposed in [5], we succeeded in uniquely identifying 81% of mobile fingerprints, this number was the same value obtained by [5], besides the 9.98% of fingerprints are contained in groups with two fingerprints per group. The insight here is that browser fingerprinting on mobile devices is still effective.

We encountered 55 operating systems in our dataset, belonging to five families: Android (15 versions), BlackBerry (five versions), FireFox OS, iOS (29 versions) and Windows Phone (five

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Source</th>
<th>Distinct values</th>
<th>Unique values</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>User agent</td>
<td>HTTP header</td>
<td>5737</td>
<td>3992</td>
<td>Mozilla/5.0 (Linux; Android 5.0.1; SHIELD Tablet Build/LRX22C) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.124 Safari/537.36</td>
</tr>
<tr>
<td>Accept</td>
<td>HTTP header</td>
<td>41</td>
<td>16</td>
<td>text/html,application/xhtml+xml,application/xml;q=0.9,<em>/</em>;q=0.8</td>
</tr>
<tr>
<td>Content encoding</td>
<td>HTTP header</td>
<td>15</td>
<td>1</td>
<td>gzip, deflate, sdch, br</td>
</tr>
<tr>
<td>Content language</td>
<td>HTTP header</td>
<td>984</td>
<td>539</td>
<td>en-GB,en;q=0.8</td>
</tr>
<tr>
<td>List of plugins</td>
<td>JavaScript</td>
<td>235</td>
<td>153</td>
<td>Plugin 0: Chrome PDF Viewer; Portable Document Format; internal-pdf-viewer. Plugin 1: Native Client; ; internal-nacl-plugin.</td>
</tr>
<tr>
<td>Timezone</td>
<td>JavaScript</td>
<td>40</td>
<td>3</td>
<td>-60</td>
</tr>
<tr>
<td>Screen resolution and color depth</td>
<td>JavaScript</td>
<td>458</td>
<td>212</td>
<td>960x600x32</td>
</tr>
<tr>
<td>List of fonts</td>
<td>JavaScript</td>
<td>117</td>
<td>97</td>
<td>Android, cooky_Clockopia_Cool, jazz_Courier, New_Droid, Arabic, Naskh_Droid, Sans_Armenian_Droid, Sans_Ethiopic_Droid</td>
</tr>
<tr>
<td>List of HTTP headers</td>
<td>HTTP headers</td>
<td>313</td>
<td>140</td>
<td>Upgrade-Insecure-Requests, Referer, Connection, Accept, X-Real-IP, Accept-Language, armv7l</td>
</tr>
<tr>
<td>Platform</td>
<td>JavaScript</td>
<td>41</td>
<td>12</td>
<td>Cwm fjordbank glyphs vext quiz, Cwm fjordbank glyphs vext quiz, Cwm fjordbank glyphs vext quiz, Cwm fjordbank glyphs vext quiz, Cwm fjordbank glyphs vext quiz,</td>
</tr>
<tr>
<td>Canvas</td>
<td>JavaScript</td>
<td>1323</td>
<td>759</td>
<td>NVIDIA Corporation</td>
</tr>
<tr>
<td>WebGL Vendor</td>
<td>JavaScript</td>
<td>22</td>
<td>7</td>
<td>NVIDIA Corporation</td>
</tr>
<tr>
<td>WebGL Renderer</td>
<td>JavaScript</td>
<td>124</td>
<td>42</td>
<td>NVIDIA Tegra</td>
</tr>
</tbody>
</table>

Table 1: Browser measurements of AmIUnique.com fingerprints with an example.
versions). Fingerprints belonging to Android and iOS represent most of the data (56% Android and 41% iOS); note that BlackBerry, FireFox OS and Windows Phone represent only the 3% of data. Consequently, we consider that results obtained over the set of these operating systems will not affect results over the complete data. The Distinct values and Unique values columns in Table 1 give a global overview of the most discriminating attributes in a fingerprint. Finally, the last column displays a complete example of a browser fingerprint on mobile devices.

Figure 1 shows the percent of unique fingerprints for all five operating system families. Android and iOS have similar percents of unique fingerprints, 81% and 80%, respectively. The rest of operating system have higher values.

Table 1 reveals that we collected 3,992 unique values for the user-agent attribute, making it the most distinctive attribute. This is due to the many possible combinations between the operating system, its version and the browser of the device. In addition, we observed that the user-agent on mobile devices contains more information than on desktop machines. In these devices, the user-agent is used to communicate information regarding to the manufacturer or the Internet Service Provider. Wu et al. [10] demonstrated the presence of some manufacturers that are not real manufacturers, since the third-part ROM is modified. This makes the devices easier to be identified.

Canvas attribute is the second most distinctive attribute, with 759 unique values. Our canvas test includes emojis. Since every phone manufacturer provides their own sets of emojis, such a test can be a powerful technique to uncover information.

With the absence of the Flash plugin to provide the list of fonts and the poor presence of plugins, mobile devices would be less recognizable than desktop machines. Yet, our results tend to demonstrate that fingerprints on mobile devices are still unique, because of specificities of mobile software environments.

## 4 Conclusion and Future Work

The wide range of possibilities for customizing browsers makes diversity a threat for privacy and security in web browsers. Besides, differences in hardware and software between mobile and desktop devices pose a new challenge, making that fingerprinting techniques and solution to counter them require different analyzes and strategies. The literature reports on many ways to fingerprint browsers, as well as some solutions to limit browser fingerprint tracking and highlight some of their shortcomings. However, due to the increasing speed of technology development, efforts to counter fingerprinting are not enough.

We provide a short analysis of fingerprints collected from mobile devices and in addition to the work realized in [5]. We corroborate with updated data that fingerprinting on mobile
devices is still quite effective, with 81% of success.

We detected that despite the most popular operating systems, Android and iOS represent 97% of data collected, the number of existing versions joined to the combination of values for the rest of components makes fingerprinting techniques to success.

The establishment of some strategies to defend against fingerprint tracking entails reconfiguration of components that make up fingerprints. In this sense there are two different directions: to remove diversity making all fingerprints equals or to change all elements in a fingerprint making the fingerprint completely different in every moment. Both cases require radical changes in fingerprint components. Since there are some component whose values depend on the hardware and software environment, if one of both directions is followed purely some changes will not be easy or will be meaningless, unless a fake fingerprint is created.

In the future, we aim to provide effective algorithms and tools to mitigate tracking through browser fingerprinting. New solutions will be based on decentralized identification of similar devices that are prone to share the same fingerprint, combined with the automatic reconfiguration of devices in order to modify the fingerprint. We plan to model the parts of a device that can be reconfigured and automatically reason about this model in order to take autonomous reconfiguration decisions that increase the similarity among devices that are in the same cluster.

References


