Obfuscation for and against device fingerprinting
Position Paper for Symposium on Obfuscation
New York University, February 15, 2014

Gunes Acar

KU Leuven, Dept. of Electrical Engineering (ESAT), COSIC, iMinds, Leuven, Belgium.

1 Introduction

Web-based device fingerprinting is the process of collecting information through the browser that is sufficient to perform device identification. Fonts, screen dimensions, language, time-zone, plugins and user agent string are examples of properties that, if combined, may serve as globally unique identifier. Indeed, a 2010 study by Peter Eckersley (EFF) based on data collected from about half a million browsers demonstrated that web-based device fingerprinting may well be a feasible way to uniquely identify browsers [1]. This was in line with what Mayer found in 2009, which, to the best of our knowledge, was the first study that discussed web-based device fingerprinting [2].

2 Fingerprinting in the "real" world

Until very recently, we had little to no information about the real world uses of device fingerprinting. Two studies in 2013 showed that web-based device fingerprinting is used by many websites, including some of the most popular ones, and that there are many companies that provide fingerprinting services using novel methods that were not considered in the two seminal studies mentioned above.

Analyzing the techniques and adoption of three fingerprinting companies, Nikiforakis et al. discovered the use of JavaScript based font probing by fingerprinters when Flash or Java plugins are not available [3]. Moreover, the same study found that companies use hidden Flash objects to circumvent proxy servers that privacy conscious people may use to conceal their real IP addresses.

In a later study, we identified 16 previously unknown fingerprinting scripts or Flash objects by developing a a crawler based framework called FPDetective [4]. By crawling the homepages of the top million websites in Alexa database with FPDetective[1], we found that 145 of the top ten thousand websites use Flash-based fingerprinting, and more than 400 of the top one million websites use JavaScript-based fingerprinting which would work for Flash-less devices (e.g. some smartphones or tablets) or users who disable Flash. The same study discovered novel distribution methods for fingerprinting scripts such as those embedded in ad banners or third party widgets and those loaded with asynchronous JavaScript calls. These scripts then remove themselves from the page's DOM after collecting and sending the fingerprint, probably to evade detection.

In the rest of this paper, we will first discuss the problems associated with device fingerprinting using the concepts laid down by Brunton and Nissenbaum [5] and the results of our

[1] Source code for FPDetective is freely available at `https://github.com/fpdetective/fpdetective`

work on fingerprinting [4]. We will then comment on the potentials of obfuscation against device fingerprinting using the Tor Browser and user agent switcher extensions as examples. Finally, we will end the paper with conclusions and open questions.

3 Problem

We think that *power and knowledge asymmetries*, as described by Brunton and Nissenbaum [5], are very useful concepts for grasping the problems associated with device fingerprinting. Brunton and Nissenbaum argue that users, whose personal digital data are collected and analyzed, are subject to two kinds of asymmetries: (1) *knowledge asymmetry*, meaning users have no or very limited knowledge about what happens to their data; and (2) *power asymmetry*, meaning users have no power, or control over the process of data collection and analysis.

3.1 Knowledge asymmetries

As most other tracking mechanisms, device fingerprinting runs behind the scenes, without users' awareness or consent. Furthermore, fingerprinting may leave no trace behind, and even if it does so, such as by storing cookies, it may not be possible to tell them apart from cookies left by traditional trackers. Its "stateless" nature makes it very hard to detect even for the technically savvy users. And, not surprisingly, most of the fingerprinting code we found in our study [4], was at least partially obfuscated, raising another barrier to understand what it collects from users' devices.

Passive and server-side fingerprinting techniques such as those based on clock skew[2] or network packet irregularities[3] pose an even greater challenge for detection, as they don't require running code on the user's device.

What we know about what happens to data once it leaves the users' device is again, very limited. Publicly available information about fingerprinting "solutions" shows that companies use massive device "reputation" databases[4] where users' actions across different sites are aggregated along with the device fingerprints and reputation scores. Currently, none of the companies that were found to use fingerprinting in our study offer an option for users to access the data stored about them [4].

3.2 Power asymmetries

Another imbalance facilitated by device fingerprinting is lack of user control over the fingerprinting process. Unlike cookies, there is no option in browsers to "block or disable fingerprinting". The so-called private browsing modes of browsers are not designed to protect against fingerprinting, and disabling cookies may even make the situation worse, as this would be a very rare configuration that could be used to enhance identification. Disabling JavaScript, Flash or similar plugins may prevent the collection of some information, but will again make the fingerprinting easier.

[2] See, for example, "Time-Differential Linking" described at `http://www.the41.com/solutions/complex-device-recognition`

[3] For example, "ThreatMetrix goes beyond simple Browser Fingerprinting to include Operating System Fingerprinting, Protocol Fingerprinting and Packet Fingerprinting in order to provide a complete view of whether a device should be trusted or not [...]" `http://www.threatmetrix.com/how-device-fingerprinting-works/`

[4] See, for instance, `https://www.iovation.com/blog/iovation-surpasses-one-billion-tracked-devices`

Unfortunately browser vendors are far from providing a solution against device fingerprinting[5] and there is no working browser extension that can be used to defend against fingerprinting.

In a further attempt to avoid tracking, users may turn to using proxies to uncover real IP addresses. However, companies limit the effectiveness of such methods by employing practices such as "proxy piercing" that allows them to detect the geolocation of the user nevertheless. Although, most of the companies seem to legitimize this invasive technique as a protection against fraudsters, others see no problem in apllying the method for marketing purposes, disrespecting users' choices[6]. Actually, the analogy between discourses over surveillance and device fingerprinting is notable: for the former, it's the terrorists that makes surveillance necessary and legitimate, taking away citizens' right to a private life; and for the latter, it is fraudsters that must be stopped, which makes it legitimate and necessary to fingerprint each and every visitor without their knowledge or consent.

Finally, opting out from being tracked by fingerprinters maybe a good example to what Brunton and Nissenbaum refer to as "onerous at best and ridiculous at worst" for attempts to opt-out from data collection [5]. The only fingerprinting company that offers opt-out frankly admits that opting out from tracking does not necessarily prevent their customers (e.g. websites that use their fingerprinting software) to recognize your device[7]. Paradoxically, it's again fingerprinting that is used to recognize users who opted out and make sure that their preference persists[8]. In fact, this is in line with our findings about the effect of DoNotTrack header, or lack thereof. Simply put, in our study [4], we haven't found a single website that stopped fingerprinting when we visit them with the DoNotTrack header signal turned on. Yet, it is possible that companies may treat the data in a different manner on the server side when they receive a DoNotTrack signal. [9]

4 Calling upon obfuscation as a protection strategy?

In the next subsections, we will discuss the uses of obfuscation for protecting users against web-based device fingerprinting through two examples. We believe that the non-trivial challenge posed by fingerprinting may serve as an instructive case study for obfuscation based protection.

4.1 Obfuscation based fingerprinting countermeasures in Tor Browser

To the best of our knowledge, the Tor Browser is the only piece of software that adopts a well informed and carefully designed approach based on obfuscation to thwart fingerprinting.

[5] A partial exception was a recent effort from Mozilla to limit plugin enumeration: "Bug 757726 - disallow enumeration of navigator.plugins" `https://bugzilla.mozilla.org/show_bug.cgi?id=757726`

[6] See, for example, patent entitled "Proxy Piercing for Identifying Address of and Marketing to a User Device Involved in an Online Transaction" `http://www.google.com/patents/US20110218856`

[7] "Note that we do have some customers who use our services on their own web sites, *including for things like fraud prevention*, and the opt out you do here will not necessarily prevent them from using our technology to recognize your device when you visit their sites." (emphasis added) `http://bluecava.com/opt-out/`

[8] `http://www.informationweek.com/security/risk-management/browser-fingerprinting-9-facts/d/d-id/1112056?page_number=2`

[9] Actually this is what is claimed by a fingerprinting company, in response to a news article based on our findings (see the linked article in the previous footnote). Due to the aforementioned opaqueness problem, we have no means to verify or dispute this claim, or comment about which interpretation of DoNotTrack they may refer to.

The design document of the Tor Browser describes the goals and implementation details of countermeasures. These include justifications for the deployed mechanisms based on academic research and known fingerprinting attacks[10].

The main goal of countermeasures in the Tor Browser is to achieve *Cross-Origin Fingerprinting Unlinkability*, that is, to prevent linking of users' activities across different websites by collecting and comparing her fingerprint. The high level approach followed by the Tor Browser is one of *cooperative obfuscation* [5], realized by reporting uniform values to potential fingerprinters for all Tor users. In order to achieve this, the Tor Browser spoofs, among others, properties such as system language, browser version, timezone and operating system. It also sends fixed values in *User Agent Accept-Language* and *Accept-Charset* HTTP headers.

On the other hand, the Tor Browser disables some browser APIs such as the Battery API and the Network Connection API and plugins such as the Flash plugin. Moreover, when asked to extract image data from canvas, the Tor Browser returns a blank (white) image to thwart canvas-based fingerprinting attacks [6]. The Browser also limits the number of fonts that a page can use, thus disabling any attempts to enumerate underlying system fonts for fingerprinting purposes.

The two latter approaches outlined in the paragraphs above can be loosely mapped to two types of disinformation described by Alexander and Smith [7] , that is, *constructive* and *destructive disinformation*. The browser properties that have to be disclosed (e.g. by the protocol) are replaced by uniform values for all Tor users, whereas functionalities that might allow an attacker to uncover real device or operating system properties are simply disabled. Moreover, when there is more than one way for the fingerprinter to query a property (e.g. getting screen size using JavaScript but also CSS media queries), the Tor Browser reports the same value.

4.2 User agent switcher extensions

In their aforementioned study on device fingerprinting, Nikiforakis et al. evaluated 12 browser extensions that are designed to spoof browser's user agent string and fool fingerprinters. They found that not only do none of the the extensions completely hides the true identity of the browser, but, since they are installed by a small number of users, they also lead to increased distinguishability of the users' fingerprints [3]. Nikiforakis et al. notably refers to this as an *iatrogenic* problem, referring to "harm resulting from medical treatment"[11]. Eckersley, too, observes this phenomenon which he refers to as *The Paradox of Fingerprintable Privacy Enhancing Technologies* [1].

The failures of these examples can possibly inform "obfuscation studies" with insights into common pitfalls when dealing with complex pieces of software, like modern browsers, where an attacker may find many side channels that can be used to uncover real values of obfuscated properties.

5 Conclusion and open questions

Device fingerprinting poses a greater power and knowledge asymmetry for users, by rendering existing countermeasures against tracking useless, being unexpected and almost completely

[10] `https://www.torproject.org/projects/torbrowser/design/#fingerprinting-linkability`

[11] `http://en.wikipedia.org/wiki/Iatrogenesis`

invisible. We definitely lack a basic, widely deployable solution against fingerprinting that may give users effective control and transparency over the fingerprinting process.

If obfuscation is the preferred method against fingerprinting, then a carefully designed obfuscation scheme that is sensitive to the domain specific intricacies, such as dependency between different properties or values that maximize anonymity sets, need to be considered. Unlike the Tor Browser, redacting information by disabling some browser functionalities may not be an option for a solution that strives to be universal. What makes this quest more challenging is that in addition to convincing attackers with carefully adjusted parameters, the selected obfuscation mechanisms also need to play well with the huge codebases of browsers and extremely complex web ecosystem so that it won't "break the web"; a problem which can even discourage browser manufacturers from thinking about a solution.

We look forward to discussing this problem with all the symposium participants in relation to tools and papers presented at the symposium. We believe that tackling such a complex problem would also inform obfuscation studies, bringing it one step closer to be the "science of obfuscation".

6 Acknowledgements

The author would like to thank Claudia Diaz, Seda Gurses and Marc Juarez Miro for their comments on the first draft of this paper.

References

1. P. Eckersley How Unique Is Your Browser?. In *Proceedings of the 10th Privacy Enhancing Technologies Symposium (PETS '10)*, pages 1–17, 2010.
2. J. R. Mayer. Any person... a pamphleteer: Internet Anonymity in the Age of Web 2.0. Senior Thesis, Stanford University, 2009.
3. N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, G. Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 541–555, 2013.
4. G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gurses, F. Piessens and B. Preneel. FPDetective: dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security (CCS '13)*, pages 1129–1140. ACM, 2013.
5. F. Brunton and H. Nissenbaum Vernacular resistance to data collection and analysis: A political theory of obfuscation. In *First Monday*, 16(5), doi:10.5210/fm.v16i5.3493, 2011.
6. K. Mowery and H. Shacham. Pixel perfect:Fingerprinting canvas in HTML5. In *Proceedings of W2SP 2012*, IEEE Computer Society, 2012.
7. J.M. Alexander and J.M. Smith. Disinformation: A taxonomy. University of Pennsylvania Department of Computer and Information Science, Technical Report number MS-CIS-10-13. Available online at `http://repository.upenn.edu/cis_reports/920/`, accessed 20 January 2014.