Institut für Informatik
Goethe Universität Frankfurt am Main

Device Fingerprinting for Learning Analytics

Seminararbeit zu Computational Humanities - Learning Analytics
Prof. Dr. Hendrik Drachsler

March 19, 2018

Luis Glaser
Martikelnummer: 5312890
Informatik B.Sc.

Contents

Abstract

Abstract - Education is being reinforced by technology more and more from which students and teachers alike can profit. Even when students are part of or enrolled in one institution, the educational content they consume will most likely be distributed across the web. To improve the quality of a course, these resources should be taken into account as well, e.g. via suggestions to other students. In order to incorporate these sources without relying on traditional ways of identification like user accounts or cookies, browser fingerprinting can be a valuable alternative.

This work examines the recent work that has been done on web fingerprinting technology. Also, their effectiveness and limitations are discussed. Additionally, some of the available fingerprinting technology will be demonstrated and tested in a student-grade webservice. Finally, we envision a possible integration scenario for Learning Analytics.

1 Introduction

With the advent of online learning platforms like moodle a way to centralize the learning experience for students has come up. Students are able to perform tasks, upload their exercises or consume additional study material on web platforms that might be offered by their school or university. But not only material that is being dedicatedly distributed by an institution is also used for learning by students. Many students use external material. This might be used for deepening their understanding of the current topic or seeking additional aid for exercising. This again limits the ability of a tutor or teacher to monitor their students' activities.

Furthermore, Recommendation Systems play a central role in Technology Enhanced Learning Environments. As soon as the recommendation is based in some way on what peers or other students in general have used to study, another problem arises: There needs to be some way to collect data on who is viewing what content, maybe even enriched by quality or satisfaction measures [1]. This might be feasible when the content in question is held by one provider only, but becomes increasingly harder, when students are using external content, be it online video tutorials or *MOOCs*. Despite being material from external sources, they might provide additional value to a students curriculum.

In order to also incorporate these external materials into a learning platform or a recommendation system, their usage by a student needs to be tracked in someway. This might still be possible when using in-house technology, but will be made increasingly harder when students are using their private devices.

One approach from the content providers side would be, to track their visitors and then associate them with the student that is using a Technology Enhanced Learning Environment in question. This would hold multiple benefits:

- (1) No form of user account would be necessary, which could broaden the scope of used learning material beyond usual hubs users often have user accounts with.
- (2) Is ideally independent from a users device, [2, 3, 4] which makes it lightweight for a user and sustainable for a learning platform.

This opens the topic of this paper, accessing the usage of fingerprinting in the field of Learning Analytics. The main focus will be on recent research on fingerprinting while keeping the education focused usage scenario in mind.

The paper will be structured as follows. Section 2 will first give an explanation on what fingerprinting does and how we can use it. Section 2.2 will enumerate and discuss possible and combinable ways to fingerprint a given browser, while keeping resource efficiency in mind. We then propose criteria what such a fingerprinting tool should offer and present a tool that has been used to fingerprint browsers in Section 3. Section 4 will then discuss the results of the research undertaken and suggest a few ways of improvement. Section 5 will then conclude.

2 State of the Art

2.1 Browser Fingerprinting

In this paper the term *fingerprint*, if not specified further, will refer to a *browser fingerprint* analogously to the suggestion by Acar and colleagues: "A [browser] *fingerprint* is a set of system attributes that, for each [browser], take a combination of values that is, with high likelihood, unique, and can thus function as a [browser] identifier." [5]. It should thus be noted, that most of the features that will be discussed in section 2.2 will focus on *browser fingerprinting*. Features like the `User-Agent` String or most of the features extracted via JavaScript can differ from browser to browser. Fingerprinting across different browsers on one machine introduce another level of complexity. Then, multiple *browser fingerprints* need to be merged into one *device fingerprint* [3].

As already hinted, browser attributes can add different levels of value for fingerprinting. Some attributes may have only a limited number of states or be shared between many devices (e.g. boolean variables or coarse OS version). Other attributes contribute a lot to the quality of a fingerprint like the list of fonts [5, 6]. Thus, the likelihood of uniqueness does not only increase by the number of values we extract, but also by the quality of those values.

Some of these areas from which browser attributes can be extracted are rather new, like the `WebGL API` and all features of the `HTML5 canvas` element. Others are on the way to be phased out like `Flash`, `Silverlight` or in-browser `Java`, as Laperdrix and colleagues have already seen in their dataset [6].

Research about web fingerprinting does not only focus on how values to differentiate are generated, but also how other identifiers can be persistently saved. Acar and colleagues have pointed out that content providers use cookie syncing to further keep alive a cookie, even after users have cleared their cookies and cache [4]. This technique might be valuable when applying it to Learning Analytics since it could be used to preserve a students history within restricted environments like computer labs and others.

The possibilities and limits of fingerprinting have found recent attention in research. Their work has been focused on three main areas. (1) Discovery of new attributes that can be used to fingerprint [7, 8, 9], (2) studies that try to test the effectiveness of browser fingerprinting [6, 10, 11] and (3) find ways to detect websites that employ someway of fingerprinting [11, 5, 4]. For the purpose of this research, we will mainly address studies from (1) and (2). (3) will be useful to us for looking at how the industry already uses this technology and what we can learn from that when transferring to Learning Analytics.

2.2 Fingerprintable Attributes

In this section we will discuss different fingerprinting attributes, how we can obtain them and their quality if they have been tested already.

It must be noted, that the enumeration in Table A is not exhaustive, it only contains attributes that have either been mentioned in multiple papers or that have been empirically studied in at least one. Eckersley also mentions fingerprinting irregularities within the TCP/IP stack, clock skew and others [10]. The available fingerprinting techniques presented below are not exclusive to each other but are most valuable when combined in order to increase the quality of a browsers fingerprint.

2.2.1 HTTP-Header

The most straightforward source of fingerprintable attributes is the `HTTP-Header`. It's being sent with every request and response message and can thus be obtained fairly easily. A variety of them is context-dependent, like the `referer` field, which thus are not useful for fingerprinting. The permanent ones have already been exploited by a few studies [6, 10] and proven to contribute significantly to the diversity of fingerprints.[1]

[1]Shannon's entropy for the AmIUnique dataset was 0.570, 0.531 for the Panopticlick dataset [6].

The `HTTP-Header` becomes even more important on mobile devices, since applications and even mobile carriers are customizing the `User-Agent` Attribute of the `HTTP-Header` which increases the likelihood of discriminating information[6]. Some browser extensions offer spoofing of the `User-Agent` String. This, counterintuitively, makes the browser more distinguishable because unless many other browser share this spoofed string, the group they share the string with is smaller then a basic wide spread `User-Agent` string [10].

2.2.2 Fonts

Fingerprinting characteristics in browsers based on their displayed font can be carried out in two ways. Either by (1) fingerprinting the set of available fonts of a browser and secondly by fingerprinting the (2) rendering process of single symbols of a font

(1) The variety of fonts installed on a device can be very revealing as found by Laperdrix and colleagues [6]. These can either be revealed directly via browser plug-ins like Flash or Java. When collecting the list of fonts from these sources they are unordered thus adding additional entropy by the order they are returned in [11]. Modern privacy oriented browsers already block these types of direct enumeration of fonts [9]. These can be bypassed by rendering a string of a given font, measuring its dimensions and comparing them to a fall-back font named `"Non-Existent-Font"` or something similar. If the dimensions differ, the font is available in the browser. Furthermore, Font Fingerprinting is even more revealing on mobile devices since many manufacturers on Android offer their own designs, thus enabling fingerprinting down to device level [5].

The more browser centered approach targets the rendering of font directly. (2) This renders single glyphs from Unicode and compares the hashes. Unlike the previous variant fonts aren't targeted directly but called by general `CSS`-names like `sans-serif` or `cursive`. This discloses both browser specific design choices and device particularities of the OS font rendering engine. [9]

Of course, all these processes are performed off the screen and is thus transparent to the user. Fifield and Egelman have found the latter technique to be more discriminating then the `User-Agent`. Their font metric measured 1.5 times the entropy of the `User-Agent` string [9] .

2.2.3 Canvas

[8] Canvas fingerprinting is only little more complex than font fingerprinting but is able to add further diversity to a fingerprint. Introduced with `HTML5`, the `canvas`-Object enables tighter interaction with a device's underling hardware. Intended for enabling more sophisticated Web Applications, this can also be used to extract quirks not only from browser implementations but also from a device's hardware [8]. The fingerprinting works by drawing forms or text on this canvas object and then collecting the hashed return values from `toDataURL(type)`. Since only the hashed value ends up in the fingerprint, this can also be performed rather easily. Mowery and Shacham have also suggested drawing 3D forms onto a `canvas` to add more complexity, but Laperdrix and colleagues found these to be non consistent [6].[2] However, it's consistent and viable for drawing 2D forms, fonts or extracting values like the GPU vendor or make.

2.3 Further possibilities for fingerprinting

This section will point out additional elements that can be fingerprinted but have either not been studied extensively yet or not in the connection to fingerprintability. However they need to be pointed out, in case the intended use case in Learning Analytics especially profits from one or more of them.

Another way to find unique features of a browser is not within the realm of values but more in the realm of attributes themselves. Browser functionality today is relatively standardized, but the implementations still offer some unique features which can be fingerprinted as well, for example the `screen.mozBrightness` in Mozilla Firefox or `navigator.msDoNotTrack` in Internet Explorer [11].

[2]resizing the browser window and redoing the test returned other values.

Also, the viability of fingerprinting characteristics are different between mobile and desktop devices. As there are no plug-ins or `Flash` on mobile devices, the bulk part of diverseness must come from other values as the User-Agent.[3][6].

3 Method

After having reviewed recent research on fingerprinting, we turn to test the technique. Unfortunately due to the limited scope of this seminar paper we will focus on a small scale example fingerprinting of the aforementioned `HTTP-Header` values.

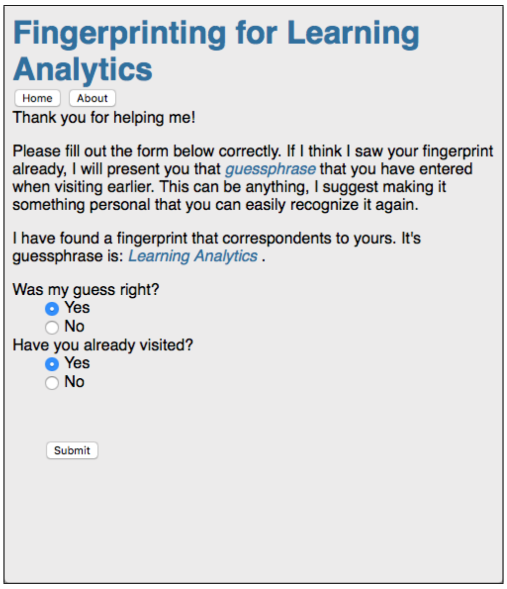
3.1 User recognition

To test the described fingerprinting method, we create a small online fingerprinting service. In a live scenario, we will depend on recognizing revisiting users to leverage this service properly. When seeing User U, represented by his browser, having n fingerprints $f_1, f_2, \ldots, f_n$ in our database, several things can occur:

- (1) (*True Positive*): U has visited already and we *correctly* connect him to Fingerprint f_U.
- (2) (*False Positive*): U has not visited yet and we *falsely* connect him to Fingerprint f_V or create a new one.
- (3) (*False Negative*): U has visited already but we *falsely* do not connect her with the correct f_U, instead creating a new one.
- (4) (*True Negative*): U has not visited yet and we *correctly* connect her to a new Fingerprint f_U.

[3]Phone Carriers include their name in addition phone model and other information in the `User-Agent`.

In order to get a rough estimate, how well we are able to hit cases 1 and 4 we will need some way to tell if we recognized users correctly. In a real life scenario this might be hard, but in our test case we can use an identifying phrase which we will call a *guessphrase*. New users will be asked to provide such a identifying *guessphrase* (e.g. name of first cat) which we will save together with the fingerprint. Revisiting users will be presented their *guessphrase* and should tell, which of the above cases apply to them. This will of course be limited to a device and browser combination, since we are aiming for *browser fingerprints* as explained in section 2.


Fingerprinting for Learning Analytics

Thank you for helping me!

Please fill out the form below correctly. If I think I saw your fingerprint already, I will present you that *guessphrase* that you have entered when visiting earlier. This can be anything, I suggest making it something personal that you can easily recognize it again.

I have found a fingerprint that correspondents to yours. It's guessphrase is: *Learning Analytics* .

Was my guess right?
[✓ Yes] No

Have you already visited?
[✓ Yes] No

Figure 1: Screenshot from `laprint`

3.2 Fingerprint creation

We will limit our fingerprint to the basic `HTML` focused features. These should provide enough information to differentiate the small set of users. This will also facilitate the analysis, because we will only need to compare strings of features. Comparing different renderings in `WebGL` would introduce significant overhead in the sample implementation, whilst the main goal is to test recall and precision of the technique.

In Figure 2 you can find all `HTTP-header` attributes we have collected from visiting users, together with values from one example visitor.

This recognition test, to again emphasize, aims to create a *browser fingerprint*, not a *device fingerprint*. In order to form the latter, additional feature extracted from the underlying hardware would be needed, which goes beyond the extent of this work. Cao and colleagues have already pointed out possibilities for doing so [3].

To test recall and precision for the fingerprints described in subsection 3.1,we created a web service called *Fingerprinting for Learning Analytics* which we used to fingerprint incoming users.[4] It works roughly as follows:

`HTTP-header` attribute	example
`user-agent`	Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:58.0) Gecko/20100101 Firefox/58.0
`accept-language`	de,en-US;q=0.7,en;q=0.3
`accept-encoding`	gzip, deflate
`accept`	text/html, application/xhtml+xml, application/xml; q=0.9, */*; q=0.8

Figure 2: Collected `HTTP-header` values.

- (1) Ask an incoming user, if she/he agrees on saving fingerprinting features of her/his browser.
- (2) Create a fingerprint and associating it with the user's *guessphrase.*
- (2a) In case the fingerprint was recognized, the *guessphrase* was presented and the user was asked whether it's correct.
- (3) Save the result for later analysis.

This is implemented rather straightforward. If the user agrees, we collect the `HTTP-Header` Attributes seen in Figure 2 from the `request` object that is being sent with every `HTTP`-request. The 4-tuple (`User-Agent`, `Accept`, `Accept-Language`, `Accept-Encoding`) constructs the fingerprint f_U. We query our database for f_U to see if the User is known. If we have seen f_U in connection with multiple *guessphrases*, we chose one of them at random and present them to the user. We could also have thrown an error, but in a real life scenario we wouldn't want to interrupt the users experience. Another solution would have been to then fall back to a default user.

[4]Inspired by the work of amiunique, but more lightweight.

3.3 xAPI integration

```
{
  "id":"12345678",
  "actor":{
    "openid":"http://www.example.org/video1/fp?acceptEncoding=gzip
      ,deflate\&UserAgent=Mozilla/5.0(Macintosh;IntelMacOSX10.11;.."
  },
  "verb":{
    "id":"http://adlnet.gov/expapi/verbs/watched",
    "display":{
      "en-US":"watched"
    }
  },
  "object":{
    "id:":"http://example.adlnet.gov/xapi/example/activity"
  }
}
```

Figure 3: Sample xAPI Statement with Fingerprint (modified from documentation)

Another subject to touch on is the integration with other Educational Technology. To properly use fingerprinting for tracking educational experiences, we need to have these fingerprints conform to a standard. A standardized exchange format is `xAPI`[5]. Since a fingerprint in general will refer to a user that visits a website, we will propose ways to construct an `actor` object, that can then be combined with other properties to form a statement. The triple of (`actor`, `verb`, `object`) is minimal. Note that it will be impossible to come up with a fully standard compliant way to implement a fingerprint as an `actor` object, since these are required to only identify one agent or a group of agents, which cannot be guaranteed by fingerprinting.

The first way to construct the `actor` object which would be most intuitive, is to simply store all fingerprint attributes. This would result in a user object that is similar to what the fingerprinting entity has seen, adding no further complication. Unfortunately, the xAPI specification forbids an `actor` object to be customized and only allows it to have one out of four Inverse Functional Identifiers from which we can already exclude `mbox`, `mbox_sha1sum` and `account`. If these could be obtain or created, fingerprinting would possibly be obsolete.

The only way to construct a valid IFI would be to create an OpenID. The URI would begin with the site that the fingerprinted user has visited and then store all attributes in a form of query attributes.[6] We would then

[5]Specification available on GitHub: xAPI-Spec.

[6]e.g.: `http://www.example.org/video1/fp?acceptEncoding=gzip,deflate&UserAgent=Mozilla/5.0(Macintosh;IntelMacOSX10.11;`[...]

have encapsulated the fingerprint in the `actor` object to build a statement with, as seen in Figure 3.

4 Results

We ran the site from March 14. until March 19, mostly visited by friends and colleagues, see Figure 4 for general metrics. We excluded 2 of the total 34 collected fingerprints because they did not have a *guessphrase* associated. Thus we cannot decide whether they stem from the same browser or different ones. We see on first glance that roughly $\frac{3}{4}$ of incoming devices where in an anonymity set of size 1 only by collecting the `HTTP-Header`. It must be noted that it's likely that this fraction rapidly drops for larger N, as Laperdrix and colleagues found it to only directly identify $\frac{3}{4}$ of visiting devices [6]. Similar to their findings, the `User-Agent` attribute adds the most diversity to the fingerprint, which can be seen in Figure 5.

Total Fingerprints	34
Valid Fingerprints	32
Unique Fingerprints	26

Figure 4: Descriptives

`HTTP-header` attribute	unique values
`user-agent`	24
`accept`	5
`accept encoding`	1
`accept language`	10

Figure 5: Fingerprints

The fingerprinting service we ran also asked visiting users for giving a *guessphrase* that we associated with their fingerprint. This was inteded to make the service more engaging. Furthermore we encouraged them to revisit in order to find out, whether their fingerprint has changed. Finally this enabled us to have a look at precision and recall. Surprisingly despite only 75% of fingerprints being unique, both recall and precision were either 1 or rather close to 1, see Figure 6.

While being seemingly good news at first, it also seems rather unlikely, having only $\frac{3}{4}$ of unique fingerprints. Since one user personally reported that they already saw a *guessphrase* when visiting for the first time with a browser, we assume that users did not fill out the form completely if the *guessphrase* did not match. Unfortunately, we did not track this case when running the site.

In order to test the validity of our results, we ran a test in which we tried to guess the *guessphrase* of the fingerprints we had already collected. This gave us a value of 0.81 for precision, which seems more reasonable. Unfortunately we couldn't retrieve recall from the simulation since constructing negatives by splitting our dataset would have decreased the number of fingerprints too much to still make reasonable statements about them.

Metric	Value
Total	79
True Positive	46
False Positive	2
True Negative	31
False Negative	0
Precision	0.95
Recall	1.00

Figure 6: Precision & Recall

After our test had ended, multiple users reported via mail, that the script was completely unable to recognize their devices. After looking at the corresponding fingerprints, we saw that they indeed were unique. This occurred with two Samsung smartphones, which suggests the frontend part of our tool was faulty and did not change the display accordingly.

5 Conclusion

In this paper we assessed the recent research on fingerprinting and tested some of the techniques with the help a web service. As discriminating as the technique has been shown to be, there must be cases where it is not. Especially institutions that issue a large number of identically configured devices that are running the exact same OS configurations and possibly are updated centrally altogether, will create one or a small number of anonymity sets. Thus, fingerprinting likely will not replace previous ways of identifying users like cookies or log-ins. Despite these limitations, it can prove to be a valuable addition to other identification techniques. In our use case, fingerprinting could potentially be a initial identification technique that can personalize content for a new visiting user until he decides to *upgrade* to a standard user account. Also, when improving recommendation systems, the number of associated visitors of some educational material might be small or diverse enough for fingerprinting to be sufficient to uniquely identify the set of visitors.

6 References

[1] H. Drachsler, K. Verbert, O. C. Santos, and N. Manouselis, "Panorama of Recommender Systems to Support Learning," in *Recommender Systems Handbook* (F. Ricci, L. Rokach, and B. Shapira, eds.), pp. 421–451, Boston, MA: Springer US, 2015.

[2] T.-F. Yen, Y. Xie, F. Yu, R. P. Yu, and M. Abadi, "Host Fingerprinting and Tracking on the Web: Privacy and Security Implications.," in *NDSS*, p. 16, 2012.

[3] Y. Cao, S. Li, and E. Wijmans, "(Cross-)Browser Fingerprinting via OS and Hardware Level Features," Internet Society, 2017.

[4] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The Web Never Forgets: Persistent Tracking Mechanisms in the Wild," pp. 674–689, ACM Press, 2014.

[5] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel, "FPDetective: Dusting the web for fingerprinters," pp. 1129–1140, ACM Press, 2013.

[6] P. Laperdrix, W. Rudametkin, and B. Baudry, "Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints," in *Security and Privacy (SP), 2016 IEEE Symposium On*, pp. 878–894, IEEE, 2016.

[7] S. J. Murdoch, "Hot or not: Revealing hidden services by their clock skew," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pp. 27–36, ACM, 2006.

[8] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in HTML5," *Proceedings of W2SP*, pp. 1–12, 2012.

[9] D. Fifield and S. Egelman, "Fingerprinting web users through font metrics," in *International Conference on Financial Cryptography and Data Security*, pp. 107–124, Springer, 2015.

[10] P. Eckersley, “How unique is your web browser?,” in *International Symposium on Privacy Enhancing Technologies Symposium*, pp. 1–18, Springer, 2010.

[11] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, “Cookieless monster: Exploring the ecosystem of web-based device fingerprinting,” in *Security and Privacy (SP), 2013 IEEE Symposium On*, pp. 541–555, IEEE, 2013.

A Tables

Attribute-Name	Source	Remark	Example Value	Studies
`User-Agent`	`HTTP-Header`		Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:58.0) Gecko/20100101 Firefox/58.0	[6, 10]
`accept`	`HTTP-Header`		text/html, application/xhtml+xml, application/xml; q=0.9, */*; q=0.8	[6, 10]
`accept-encoding`	`HTTP-Header`		gzip, deflate	[6, 10]
`accept-language`	`HTTP-Header`		de,en-US;q=0.7,en;q=0.3	[6, 10]
List of HTTP-Headers	`HTTP-Header`		see above	[6]
List of Fonts	`Flash`, `Java`, `JavaScript`		Abyssinica SIL, Aharoni CLM, AR PL UMing CN, ... [6]	[6]
List of plug-ins	`JavaScript`		Plugin 1: Chrome PDF Viewer, Plugin 2: ... [6]	[6]
`Timezone`	`JavaScript`		-60 (UTC+1) [6]	[6]
Screen Resolution & Color Depth	`JavaScript`		1920x1200x24 [6]	[6, 5]
Cookies enabled	`JavaScript`		yes [6]	[6]
Use of local/session storage	`JavaScript`		yes [6]	[6]
`navigator.platform`	`JavaScript`	spoofed `User-Agent`	MacIntel	[6, 5]
`WebGL Renderer`	`JavaScript`		NVIDIA Corporation [6]	[6, 8]
`WebGL Vendor`	`JavaScript`		GeForce GTX 650 [6]	[6, 8]
Do Not Track Flag	`JavaScript`		yes	[6]
HTML5-canvas	`JavaScript`			[6, 8]
Usage of an AdBlocker	`JavaScript`		no [6]	[6]
IP Address	IP Layer	Mostly for respawning	`127.0.0.1`	[2]
Browser Vendor & Version	Feature availability		Firefox/58.0	[11]

Table 1: Recently studied browser attributes