Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Exposé

Implementation of browser fingerprinting recognition and integration into PrivacyScore

submitted by

Tronje Krabbe
born 11. August 1994 in Hamburg
student number 6435002

submitted August 8, 2017

Betreuer: Dipl.-Inf. Heinz Mustermann
Erstgutachter: Prof. Dr.-Ing. Hannes Federrath
Zweitgutachter: N.N.

Contents

1 Introduction

When browsing the web, users can be tracked through the use of cookies, which store information on the user's computer that can also be accessed by a remote server. If the user wishes not to be trackable, they can modify or delete any cookie as they see fit. This interaction gives operators of websites the ability to track users only if they wish to be tracked. There are, however, other methods of uniquely identifying a user and tracking them during their use of the internet, which is not as easily mitigated as a cookie [2]. One of these methods is called 'browser fingerprinting', and it will be the focus of this thesis.

Tracking a user does not simply mean recognizing whether a visitor to one's website is a new or an existing user. Identifying information can be passed on or sold to partners, advertisers, or even governments, in order to construct a rich browsing history of a user.

Browser fingerprinting, also known as device fingerprinting, and in the following often simply referred to as 'fingerprinting', works by analyzing a web browser's configuration and settings; mostly installed fonts and HTML5 canvas behavior [4], as well as language settings, time zone settings, installed add-ons, and more. Flash is also sometimes used, though its end-of-life lies in the foreseeable future, reportedly in the year 2020[1]. These attributes are readily available to be collected through JavaScript functions. Simply recording all function calls made by the JavaScript front end of a website can reveal whether fingerprinting is likely to be taking place or not [5][7].

The goal of this thesis' work is to implement automated fingerprinting detection, and integrate this into PrivacyScore [2].

2 Related Work

There are some recent projects on the topic of browser fingerprinting that can be of use for this thesis. The following is a rundown of the most relevant and useful works.

Acar et al. provide some useful metrics to differentiate between fingerprinting and legitimate use of fingerprint-building resources [1]. They have implemented a fingerprinting-detection framework called 'FPDetective', that can be used as a guide and something to test against.

Englehardt et al. have developed a framework called 'OpenWPM' which can be used to analyse the way a website handles the users' privacy. They have used this to analyse one

1. https://arstechnica.com/information-technology/2017/07/with-html5-webgl-javascript-ascendant-adobe-to-cease-flash-dev-at-end-of-2020/
2. https://privacyscore.org

million websites' tracking behavior [4]. Their work can be used to implement customized fingerprinting detection, and their dataset can be tested against. They further provide some good insights into fingerprinting in general.

FaizKhademi et al. have created 'FPGuard', a browser extension designed to alert the user about active fingerprinting, and help prevent it [6], which can help with evaluation of our implementation.

Am I Unique? [2] is a website that can generate a browser fingerprint for demonstration purposes. It can be of great use in evaluating a fingerprinting detection algorithm, since it is an obvious and safe assumption that it employs fingerprinting.

The Panopticlick project by the Electronic Frontier Foundation also creates a browser fingerprint upon request [7], and can be used to confirm at least a minimal working state of the software, just like *Am I Unique?*.

3 Leading Question and Goals

The thesis presents the assertion that techniques to identify and track users across different websites without their knowledge or their ability to easily intervene, violates their privacy; by creating a fingerprint of a user's browser, with sufficiently sophisticated techniques, one can uniquely identify one user among hundreds of thousands [2], re-instate any cookies the user may have deleted, or simply track and analyze their use of any number of web services for a multitude of malicious reasons [3]. Methods exist to mitigate the effectiveness of browser fingerprinting, such as the one presented in [8]; these are, however, usually much more complicated and specialized for the average user of a web-browser. The author will therefore attempt to implement a technique to recognize when a website is deploying browser fingerprinting, and along the way explore the techniques used to do so, with the ultimate goal of integrating this implementation into `https://privacyscore.org`, a web-service to test and rank websites according to the extent to which they respect their users' privacy; this creates a different kind of defense against the privacy violating methods that form the topic of this thesis: informing users about websites they use and their treatment of sensitive information.

4 Methods and Approach

4.1 Metric

If a website includes a browser fingerprinting library, or makes many calls to JavaScript functions that return data which is useful in building a fingerprint, one can conclude: this website is very likely generating, saving, and possibly distributing a browser's fingerprint; simply put: it performs browser fingerprinting. However, some modern websites likely use

at least some of the attributes that can be used to construct a fingerprint for reasons other than identification. Analyzing installed fonts has an obvious, legitimate use: correctly displaying text. Thus, it is important to work out a metric or rating system of some sort, which can (somewhat) reliably represent the likelihood that fingerprinting is, in fact, taking place. Acar et al. assert that the more fonts are being requested, the more likely a website is to be practicing fingerprinting [1]. This is based on findings of Eckersley, of the Panopticlick project [3]. Acar et al. further state that they classify a JavaScript file as a fingerprinter "when it loads more than 30 fonts, enumerates plugins or mimeTypes, detects screen and navigator properties, and sends the collected data back to a remote server" [1]. Englehardt and Narayanan present a metric to detect Canvas-based fingerprinting by specifying certain attributes such a Canvas, and the code using it, should have [4]. The above considerations form a good basis upon which to build a reliable metric, which will have to be evaluated further during the implementation process.

4.2 Scope

Finding or creating a single website which collects a fingerprint, and basing development of fingerprinting recognition on it, will not yield reliable or representative results. It is imperative to test the created recognition software against a multitude of fingerprinting libraries and users of these. It might be prudent to test the software against, say, the top 500 websites as ranked by Alexa[1], as there is a high likelihood these sites employ a multitude of techniques to track their users, but there is no way to be sure if some of these sites wouldn't generate false positives. The safest way could be to create websites which employ an array of fingerprinting libraries both preexisting and implemented by the author, and then comparing the employed JavaScript calls with those from popular websites.

4.3 False Positives

Filtering false positives is a non-trivial problem in this context. Requesting fonts via JavaScript might be a dead giveaway that fingerprinting code is, in fact, fingerprinting code. However, it might cause legitimate code, such as a multimedia player, to be mistaken for fingerprinting code, as well, even though it has a legitimate reason and perhaps a necessity to know a system's installed fonts. Meta-data can be used to filter these; a list can be compiled of popular JavaScript libraries with legitimate, yet fingerprinting-like behavior, and then extended through analysis of collected data (see Technology below).

5 Implementation

TODO: Something about PrivacyScore

1. `http://www.alexa.com/topsites`

5.1 Technology

5.1.1 OpenWPM

"OpenWPM is a web privacy measurement framework which makes it easy to collect data for privacy studies on a scale of thousands to millions of sites"[1]. The author is planning to build upon OpenWPM [4] to create a software that can detect browser fingerprinting. OpenWPM includes capabilities to record "all method calls (with arguments) and property accesses for APIs of potential fingerprinting interest", providing a good basis for data collection. Analyzing and interpreting this data in a meaningful way will need to be implemented by the author, and go a long way towards creating a reliable software to detect fingerprinting.

5.1.2 SQL

Since OpenWPM produces Sqlite databases containing the results of its tests, a database will be used. It may prove useful to import the resulting Sqlite databases into a more sophisticated system, such as PostgreSQL, perhaps merging them in a meaningful way, and analyzing data not from a single website, but many, as one.

6 Schedule

Further in-depth research will have to be conducted in order to obtain a firmer grasp on the topic. The work will then have to follow an implement-evaluate-repeat pattern, and thus happen iteratively. It can be structured like so: first, implement a fingerprinting-detection algorithm. Then, evaluate its effectiveness by comparing it to other methods and perhaps analysing some sample-websites that employ fingerprinting. Then the algorithm must be tweaked and improved, and then evaluated again. The time table will look something like this:

- 2-3 weeks: First implementation and tests in place
- 5-6 weeks: Evaluation and improvement
- 3-4 weeks: Writing the thesis paper

This time table leaves about a month of the 5 month period as a buffer, to allow for some deviation.

1. `https://github.com/citp/OpenWPM`

Bibliography

[1] Gunes Acar et al. "FPDetective: dusting the web for fingerprinters". In: *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*. Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung. ACM, 2013, pp. 1129–1140. ISBN: 978-1-4503-2477-9. DOI: 10.1145/2508859.2516674. URL: http://doi.acm.org/10.1145/2508859.2516674.

[2] *Am I Unique?* URL: https://amiunique.org/ (visited on July 4, 2017).

[3] Peter Eckersley. "How unique is your web browser?" In: *Privacy Enhancing Technologies*. Vol. 6205. Springer. 2010, pp. 1–18.

[4] Steven Englehardt and Arvind Narayanan. "Online Tracking: A 1-million-site Measurement and Analysis". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. Ed. by Edgar R. Weippl et al. ACM, 2016, pp. 1388–1401. ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978313. URL: http://doi.acm.org/10.1145/2976749.2978313.

[5] Amin FaizKhademi. "Browser Fingerprinting: Analysis, Detection, and Prevention at Runtime". PhD thesis. 2014.

[6] Amin FaizKhademi, Mohammad Zulkernine, and Komminist Weldemariam. "FPGuard: Detection and prevention of browser fingerprinting". In: *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer. 2015, pp. 293–308.

[7] Electronic Frontier Foundation. *Panopticlick*. URL: https://panopticlick.eff.org/ (visited on July 4, 2017).

[8] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. "Mitigating browser fingerprint tracking: multi-level reconfiguration and diversification". In: *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE Press. 2015, pp. 98–108.