University of Applied Sciences Upper Austria
Campus Hagenberg
Department Software Engineering

Bachelor Thesis

Web Browser Fingerprinting

Author:
Janine Denise Mayer
Hagenberg, 11 October 2018

Advisor:
FH-Prof. DI Dr. Werner C. Kurschl

Declaration of Autonomy

I hereby confirm that this thesis has been composed solely by myself and that I have used no other sources or references than those mentioned in the reference list. I did not submit this thesis, in whole or in part, to another university examination office.

Hagenberg, 11 October 2018

Janine Mayer

Abstract

//add when finished
//what information this BA holds

Content Table

1. Introduction

1.1. Motivation

//here why we would need it

The internet plays a bigger roll in our everyday life from year to year.
//basically that we use the internet with increasing frequency, most people can't imagine to live without it, some statistics, use on an everyday basis
You can search for information, listen to music or stay in touch with your friends. The possibilities are infinite.

//advantages -> connect to our data which floating around in the web
But with all these advantages there also come disadvantages.
A great example is the scandal around the social media platform Facebook and the political data firm Cambridge Analytica. In which the

Why is it used at all? -> commercial advantatge, how all started

//here how it could be abused
Cambridge Analytics as Example – show both sides, that there are good and bad ways to use it

1.2.Objectives

1.2.1. Analysis of web browser fingerprinting techniques

Analysis of them with an application scenario

1.2.2. Prototype

Due to the analysis there is one of the web browser fingerprinting methods chosen. To show the practical way how this method is implemented, there will be a documentation about the prototype

1.3. Overview

Following chapters discuss the stated content:

Chapter 2 discusses the basis of web browser fingerprinting, amongst others the different methods of fingerprinting. In order to compare them later on to analyse and eventually choose one of the methods to implement the prototype

Chapter 3 on the other hand will list requirements which have to be fulfilled in order to implement a proper prototype

Chapter 4 explains the implementation and design of the prototype

Chapter 5 concludes the previous chapter based on testcases and a proper evaluation

Chapter 6 finally concludes the bachelor thesis, summarizes the results and states possible prospects

//add some links for library?

2. Foundation

2.1. Fingerprint

When thinking of fingerprinting the average person thinks of our biometric fingers. Only few know that we can be tracked over the internet using a specific fingerprint.

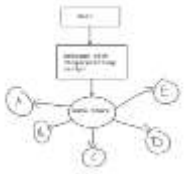
Browser fingerprinting, also known as device fingerprinting, is the systematic collection of information to identify and re-identify users[3,3][5]. This data tracking is possible due to the information provided by the browser[2,1][1,1].

Tracking over the internet is usually associated with the use of cookies. But other than cookies this form of tracking has no need to install any form of tracking data to collect information. [5]This cookieless monster is what makes users uneasy. Any interested third-party can expoit this weakness without the knowledge of the user.

How it is possible – browsers send information

Basics about browsers

Shared Data Store



Upon accessing the website, the user's fingerprint is determined and then pushed to a central data store. When the user accesses any of the websites A-E, they will be recognized, even though they do not have any cookies associated with those websites. [2, 8] - EXAMPLES

2.2.Critical configurations and plug-ins

Identify by certain configurations:
via configuration settings or other observable characteristics.
[3, 3]

2.2.1. User agent

User agent, accept headers are automatically sent to websited when a connection is initiated The user-agent string e.g. is a http request header that typically identifies the browser, renderer, version and os. For some populations, the user-agent string and IP address will commonly uniquely identify a particular user's browser. [3, 6]
What about user agent (without string)

Accept header

2.2.2. JavaScript

JS gives access to many browser-populated features like the plugins installed on the user's device

2.2.3. Flash plugin

While js does not directly permit browsers to get the full list of fonts, if Adobe flash is supported then it returns all fonts with one simple calls [2,5]

If the flash plugin is installed, its rich programming interface (API) provides access to many system-specific attributes: exact version of the operating system, list of fonts, screen resolution, timezone

2.2.4. HTML Canvas

Through the display of an HTML5 Canvas element, it is possible to collect small differences in the hardware on in the software configurations, thanks to slight differences in the image rendering between devices. The smallest pixel difference can be detected -> canvas fingerprinting
[5]

2.2.5. Others

2.3. Why is it needed?

One significant application of bs is the deanonymization of Tor users. Tor, by default comes with js enabled. Most of the modern web is nonfunctional with js disabled, but Tor websites are expected to work without js. Users who forget to turn off js may find themselves vulnerable to analytics and tracking methods.

- Fingerprinting technique -> track users in a cross-browser fashion, with a fingerprint unique to their device. Many advanced fp techniques rely on details about the device and ignore the variable information that comes with the browser or the js engine. This fp, unique to the device, can be used to identify Tor users in additional context.
- If all devices were uniquely identifiable over the internet, the entire notion of privacy on the internet would be null and void
- Fingerprinting could also provide a cookie-free means of blacklisting certain users

[2, 9-10]

Like all tracking technology, it is a double-edge sword.
e.g. constructive: to combat fraud or credential hijacking
e.g. destructive: track and collect information about their habits without the users knowing about it, attackers know which software modules are installed on a specific device

Commercial:
Nikiforakis et al. determined that there are two methods for commercial fingerprinting service delivery. The first involves first-party websites who are not involved in the fingerprinting. Instead, the fingerprinting code was delivered via advertising syndicators. In this method, the fingerprint is sent back to the fingerprinting service provider and the first-party site may not even be aware its users are being fingerprinted. In the second method, the first party site contains the fingerprinting script. It fingerprints a visitor and then the fingerprint is submitted to the website in a hidden input element when the user submits credentials. The fingerprint is encrypted so it can only be decrypted by the commercial fingerprinting service provider; therefore it must be submitted to the service provider to match and return information based on the match. This allows commercial fingerprinting companies to hide their implementation details form clients.
The study also showed some interesting facts about the categories of websited that employ fingerprinting. Out of the top 10k websites web-crawled in their study, 15% were pornography sites and 12,5% were personals/dating websites. Their explanation for the findings is that the pornography sites are using fingerprinting as a way to detect shared or stolen credentials and the dating sites make use of a fingerprint to ensure that multiple profiles are not created for social-engineering purposes.
[1, 4]

Commercial -> what do they want/ the intentions behind using fingerprinting

- To lift up the CTR (click through rate) [2, 2]

2.4. What kind of threat does it pose?

Security/privacy -> refer to DSGVO?
Have there been any incidences involving fingerprinting yet?

/*
Even in times of free speech and equality, people still don't feel safe expressing their opinion. They fear being surveillance or discriminated against.
*/

Can be used as a security measure (e.g. as means of authenticating the user).
It can be used to identify a user, correlate a user's browsing activity within and across sessions, track users without transparency or control.
[3, 3]

- Identify a user: concerns about surveillance, personal physical safety, concerns about discrimination against them based on what they read or write when using the web. [3, 4]
- An online party can correlate multiple visits (on same or different sites) to develop a profile or history of the user. This may occur without the user's knowledge or consent and tools such as clearing cookies do not prevent further correlation. Also allows tracking across origins – different sites may be able to combine information about a single user even where a cookie policy would block accessing of cookies between origins. [3, 4]
- Tracking without transparency or user control: bf allows for collection of data about user activity without clear indications that such collection is happening. Allows for tracking of activity without clear or effective user controls. (typically cannot be cleared or re-set) /*like cookies*/ [3, 4-5]

Privacy and security
Constructive and destructive uses depends on the interpretation of the user.
e.g.. to ensure that the users trying to access their services are who they claim to be -> good
else – used by advertisers to track a user and display customized ads based on browsing habits
[1, 4]

The modern web exposes a number of details about the user's device, either directly or discoverable through testing. This is largely a result of a greater need to comply with different device types, support for different features, or simply leaking data by including too much information in protocol requests.
[2,4]

High risk:

In 2010 Peter Eckersley, in association with the Electronic Frontier Foundation(EFF), conducted a study of browser fingerprinting techniques. –

- Group of people, expected to have high awareness of privacy concerns.
- Out of 470,161 browsers – 93,6% instantaneously unique finerprint
- Group -> privacy-conscious users, informed of the dangers of browser fingerprinting
- Simple algorithm for detecting changes in fingerprints.
    - Check to see if there is any fingerprint which, aside from one non-matching field, is otherwise the same.
    - Not make guess if there were multiple potential matches
    - Made 65,56% guesses – 99,1% right

[2, 6-7]

2.5.How to avoid it

You cant avoid it. Only try to reduce it

Mitigations, not solutions. Users of the Web cannot confidently rely on sites being completely unable to correlate traffic, especially when executing client-side code.
Best practices:

- Avoid unnecessary or severe increases to fingerprinting surface
- Mark features that contribute to fingerprintability
- Specify orderings and non-functional differences
- Design APIs to access only the entropy necessary
- Enable graceful degradation for privacy-conscious users or implementers
- Avoid unnecessary new local state mechanisms
- Highlight any local state mechanisms to enable simultaneous clearing
- Limit permanent or persistent state

[3, 5]

- Decreasing fingerprinting surface
- Increasing anonymity set
- Detectable fingerprinting
- Clearable local state

[3, 7]

CONTINUE AT [3, 8]

The stateless nature of browser fingerprinting makes it hard to detect and even harder to opt-out. However, there are several common methods of preventing browser fingerprinting discussed in the literature. These include:

- Having all browser vendors agree on a single set of API calls to expose to the web applications as well as internal implementation specifics;
- Having blocking tools which are maintained by doing regular web-crawls to detect tracking and incorporate blocking mechanisms into the tool;
- The introduction of a universal font list that a browser is limited to choose from for rendering;
- Reporting unified and uncommunicative attributes
- Blocking or disabling js;
- Reducing the verbosity of the User-Agent string and the plug-in version;
- Having flash provide less system information and report only a standard set of fonts

- Canvas fingerprinting

Is a difficult technique to automatically detect and prevent without false positives. A solution would be to turn to crowd-sourcing (sugg. By Acal et al.). A browser tool can default to blocking all pixel data extraction attempts, but slowly over time, take into account user feedback and improve thetool to identify valid fingerprinting attempts. Other suggested methods – having the browser add random pixel noise whenever pixels are extracted or having the browsers render scenes in a generic software renderer. These suggestions com e with a significant performance penalty and therefore, unacceptable for general use. The easiest method appears to require user approval if a script requests pixel data. Modern browsers already use this approach, for example with the html5 geolocation api. However, this introduces another permission dialogue which requires user interaction.

- Accelerometer fingerprinting

  Bojinov et al – 2 methods for the prevention of AF. The first proposal states that the feasibility of fingerprinting can be practically eliminated at the hardware level if the sensor in similar devices are calibrated to the same specification during the manufacturing process. This would remove any variance in the sensor reading, and thus, eliminate its use for fingerprinting. The second proposal is a software based solution. A random value can be added to the sensor output at the os level. This value can remain constant during continuous use and then change during periods of inactivity.

- Extensions and plug-ins

  The use of some common extensions and plug-ins were also mentioned in the literature. The notable ones which seemed to aid in reducing fingerprintability were AdBlock Plus, Ghostery and NoScript. Nikiforakis et al. examined several highly rated extensions for firefox and chrome, which allowed changing the user-agent to appear as if sessions were from different com-puters. They noted in their study that these were inadequate and contained discrepancies which led to the browser being more distinguishable. These discrepancies allow fingerprinting scripts to increase the accuracy of the fingerprint since they are able to tell the presence of a specific extension. It should be noted that these results seem to be different from what was concluded by Yen et al. in their study. They suggested that changing the User-Agent string to one that corresponds to a populare browser version was a simple method to become less distinguishable.

[1, 4]

	Browser	Canvas	JavaScript Engine	Cross-Browser	Accelerometer
Disable JavaScript	x	✓	✓	✓	✓
Disable Flash	✓	x	x	x	x
Disable Java	✓	x	x	x	x
Unified Attributes	✓	x	x	x	x
Universal/Standard Font List	✓	x	x	✓	x
Blocking Tools/Extensions	✓	✓	x	✓	x
Less Verbose User-Agent	✓	x	x	x	x
Less Verbose Plugin Version Info	✓	x	x	x	x
Calibration During Manufacturing	x	x	x	x	✓
Identically Configured Device	✓	✓	✓	✓	x

Fig. 2. Comparison of web browser fingerprinting prevention techniques

[1,5]

2.6. Web browser fingerprinting methods

2.6.1. Cookie-like fingerprinting

Users, user agents and devices may also be re-identified by a site that first sets and later retrieves state stored by a user agent or device. This cookie-like fingerprinting allows re-identification of a user or inference about a user in the same way the http cookies allow state management for the stateless http protocol.
Can also circumvent user attempts to limir or clear cookies stored by the user agent, as demonstrated by the “evercookie” implementation. Where state is maintained across user agents (as in the case of common plugins with local storage), across devices (certain browser syncing mechanisms) or across software upgrades, it can allow re-identification where active/passive fingerprinting might not.
[3, 6]

2.6.2. Active fingerprinting

Techniques where a site runs js or other code on the local client to observe additional characteristics about the browser. E.g. accessing the window size, enumerating fonts or plug-ins, evaluating performance characteristics, or rendering graphical patterns. Active fingerprinting takes place in a way that is potentially detectable on the client. [3, 6]

2.6.3. Passive fingerprinting

Based on characteristics observable in the contents of web requests, without the use of any code executed on the client. Include cookies, the set of http request headers and the ip address and other network-level information. [3, 6]

2.7. Techniques of fingerprinting

2.7.1. Browser specific fingerprinting

Large scale browser fingerprinting study was conducted by P. Eckersley in the Panopticlick project
Study has shown 94,2% of the user with flash or java could be distringuided (without cookies)
Browser-dependent features (flash/java) to retrieve information on installed fonts, plug-ins, browser platform, screen resolution
Study showed: only 15-20 bits of identifying info necessary to uniquely identify a particular browser – most identifying metrics – plug-ins, fonts, user agent, http accept, screen resolution – plug-ins due to version numbers

Weakness: unable to distinguish between instances of identically configured devices, unstable/change quite easily, changed easily caused by upgrades of plug-in/installing a new font or simply external monitor(with other screen resolution)
Adapted through a simple heuristics
[1, 2]

2.7.2. Canvas fingerprinting

Presented by Mowery and Shacham in 2012
Rendering text and WebGL scenes on to an area of the screen using HTML5ycanvas> element programmatically and then reading the pixel data back to generate a fingerprint
Works: a 2D graphics context is obtained and text or an image is drawn to the canvas. The toDataUrl(type) method is called on the canvas object and a Base64 encoding of a png image containing the contents of the canvas are obtained. A hash of the Base64 encoded pixel data is created to that the entire image is not needed to be uploaded to a website. The has is also used as the fingerprint. This technique uses WebFonts to guarantee that the correct font will always be used when drawing to the canvas. The results also showed that at least the operating system, browser version, graphics card, installed fonts, sub-pixel hinting and anti-aliasing all play a part in the final fingerprint.
It is estimated that 10-bits entropy are possible over the whole population of the web using this technique
2014 study by Acar et al. -> canvas fingerprinting the most common form of fingerprinting (ax. 5,5% of top 100k websites)
Weakness: fingerprint can change across browsers, same as browser fingerprinting
It is orthogonal to other fingerprints, it is transparent to the user and it is readily obtainable
Any website that is running javascript on a visitor's browser can fingerprint its rendering behaviour with no requirement for special access to system resources.
As other forms of browser fingerprinting, canvas fingerprinting is unable to distinguish between users who use the exact same hardware and software.
[1,2]

2.7.3. JavaScript Engine fingerprinting

Relies on JS conformance tests such as the Sputnik test suite to determine a specific browser and major version number. A method demonstrated by Mulazzani et al. executed a set of conformance test cases from the Sputnik test suite in a user's browser. They then compare the failed tests to a set of test cases that are known to fail in each version of a browser. This allow a match to be made to identify a specific browser and the major version. This can be considered more robust than relying on the User-Agent string to determine the browser and version number for use in fingerprinting. While the user-Agent string can be modified to an arbitrary value by the user, the JavaScript engine fingerprint will be unique for each browser and cannot be ported to a different browser. Other properties of this technique are that it can be used to detect modified User-Agent strings, can be used on mobile devices, and can be used to reliably identify the browser of Tor Browser Bundle users.
[1,2]

2.7.4. Cross-Browser fingerprinting

The cross-browser fingerprinting uses browser-independent features to generate the fingerprint. A simple cross-browser fingerprinting algorithm can rely solely on JavaScript and use font detection as a technique. Since a JS engine is available and enabled by default on all modern browsers, it differs from techniques which rely on browser dependent plug-ins Adobe Flash or java for obtaining a font list.
Weakness: common as already stated (p. Eckersley) -> inability to distinguish between identical configurations, as is the case with multiple users browse the web in a computer lab.
[1, 2-3]

2.7.5. Accelerometer fingerprinting

Mobile browsers -> harder to fingerprint than desktop browsers, due to the fact that mobile browsers are more uniform and they do no have the plug-ins that are available on desktop systems. A recent study by Bojinov et al. showed a technique of generating a unique fingerprint by accessing a sensor that is commonly found on mobile devices. This technique allows a fingerprinting script to avoid using traditional hardware identifiers such as the IMEI and UDID. The study showed that the accelerometer on mobile devices was exposed without user Notification by iOS and Android browsers through JS. Imprecisions in accelerometer calibration during manufacturing result in device specific measured values. The method employed by him involved repeatedly querying the accelerometer and then estimating the calibration errors in each of the 3 dimensions to generate a unique fingerprint. This form of fp can be classified as a subcategory of cross-browser fingerprinting. This is due to the fact that the generated fingerprint does not vary across browsers. Another property of this technique is that it may be exercised without user cooperation. However, a factor in limiting its usefulness is that the device must be left facing up and down to gather readings to generate the fingerprint.
[1, 3]

2.8. Analysis based on an application scenario

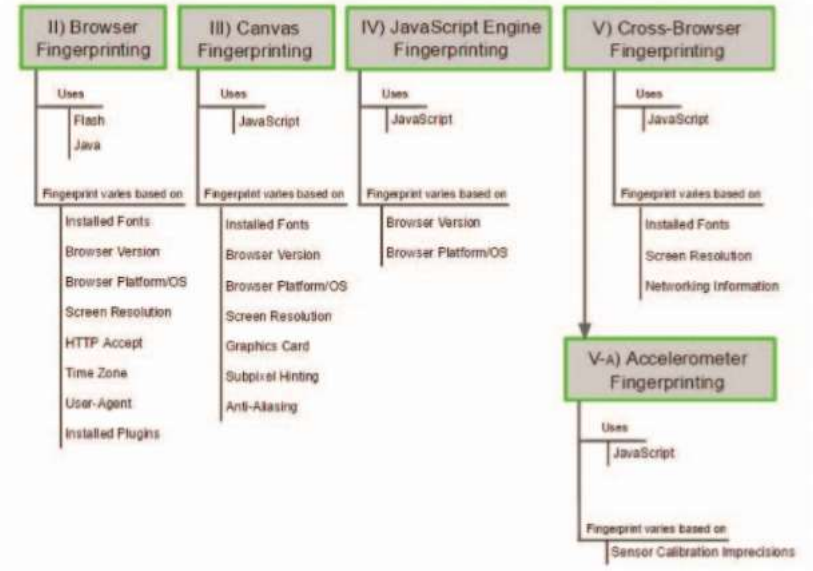


Fig. 1. Fingerprinting techniques and the attributes that affect the fingerprint.

2.8.1. Introduction

2.8.2. Browser specific fingerprinting

2.8.3. Canvas fingerprinting

2.8.4. JavaScript Engine fingerprinting

2.8.5. Cross-Browser fingerprinting

2.8.6. Accelerometer fingerprinting

2.9. Comparison

End Comparison with and Conclusion!
As seen in the descriptions, etc.

Neither of the techniques have the ability to distinguish between multiple users on a single device. [1, 5]

3. Requirement (Prototype)

3.1. Use cases

3.2. Functional requirement

3.3.Non functional requirement

4. Design and Implementation

4.1. Idea

4.2. Architecture

4.3. Programming

4.4. Obtained Data

4.5. Fingerprint

4.5.1. Calculation

4.5.2. Possibility to adjust

4.6. Visualisation

5. Evaluation

5.1. Prototype testcases

5.2.Evaluation of web browser fingerprinting methods

5.3.Comparison to AmIUnique

6. Summary

6.1. Result

What do I figure from all that
How important security is, how fraguile or better said non existent anonymity/privacy online

6.2. Prospects

How could I improve the prototype
e.g. using heuristics to distinguish better

7. List of literature

[1] Randika Upathilake, Yingkun Li, Ashraf Matrawy, A Classification of Web Browser Fingerprinting Techniques, IEEE, 2015, https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7266460

[2] Ryan Havens, Browser Fingerprinting: Attacks and Applications, 2016, http://www.cs.tufts.edu/comp/116/archive/fall2016/rhavens.pdf

[3] Mitigating Browser Fingerprinting in Web Specifications, W3C Editor's Draft 11 January 2018, https://w3c.github.io/fingerprinting-guidance/

[4] The Open Web Application Security Project, Testing for Web Application Fingerprinting, https://www.owasp.org/index.php/Testing_for_Web_Application_Fingerprint_(OWASP-IG-004)

[5] https://amiunique.org/faq

[6]

8. List of figures