

**Web Browser Fingerprinting**

Bachelorarbeit

zur Erlangung des akademischen Grades

Bachelor of Science in Engineering

Eingereicht von

**Janine Denise Mayer**

Begutachtet von FH-Prof. DI Dr. Werner C. Kurschl

Hagenberg, 25 October 2018

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Date Signature

Abstract

//add when finished

//what information this BA holds

Kurzfassung

Content Table

[Declaration 2](#_Toc528009644)

[Abstract 3](#_Toc528009645)

[Kurzfassung 4](#_Toc528009646)

[Content Table 5](#_Toc528009647)

[1. Introduction 6](#_Toc528009648)

[1.1. Motivation 6](#_Toc528009649)

[1.2. Objectives 8](#_Toc528009650)

[1.2.1. Analysis of web browser fingerprinting techniques 8](#_Toc528009651)

[1.2.2. Prototype 8](#_Toc528009652)

[1.3. Overview 9](#_Toc528009653)

[2. Foundation 10](#_Toc528009654)

[2.1. Fingerprint 10](#_Toc528009655)

[2.2. Why is it needed? 13](#_Toc528009656)

[2.3. What kind of threat does it pose? 15](#_Toc528009657)

[2.4. How to avoid it 16](#_Toc528009658)

[2.5. Critical configurations and plug-ins 18](#_Toc528009659)

[2.5.1. User agent 18](#_Toc528009660)

[2.5.2. JavaScript 18](#_Toc528009661)

[2.5.3. Adobe Flash 19](#_Toc528009662)

[2.5.4. HTML Canvas 20](#_Toc528009663)

[2.5.5. Others 20](#_Toc528009664)

[2.6. Web browser fingerprinting methods 22](#_Toc528009665)

[2.6.1. Cookie-like fingerprinting 22](#_Toc528009666)

[2.6.2. Active fingerprinting 22](#_Toc528009667)

[2.6.3. Passive fingerprinting 23](#_Toc528009668)

[2.7. Techniques of fingerprinting 24](#_Toc528009669)

[2.7.1. Browser specific fingerprinting 24](#_Toc528009670)

[2.7.2. Canvas fingerprinting 24](#_Toc528009671)

[2.7.3. JavaScript Engine fingerprinting 25](#_Toc528009672)

[2.7.4. Cross-Browser fingerprinting 25](#_Toc528009673)

[2.7.5. Accelerometer fingerprinting 26](#_Toc528009674)

[2.8. Analysis based on an application scenario 27](#_Toc528009675)

[2.8.1. Introduction 27](#_Toc528009676)

[2.8.2. Browser specific fingerprinting 27](#_Toc528009677)

[2.8.3. Canvas fingerprinting 28](#_Toc528009678)

[2.8.4. JavaScript Engine fingerprinting 29](#_Toc528009679)

[2.8.5. Cross-Browser fingerprinting 30](#_Toc528009680)

[2.8.6. Accelerometer fingerprinting 31](#_Toc528009681)

[2.9. Comparison 32](#_Toc528009682)

[3. Requirement (Prototype) 33](#_Toc528009683)

[3.1. Use cases 33](#_Toc528009684)

[3.2. Functional requirement 33](#_Toc528009685)

[3.3. Non functional requirement 34](#_Toc528009686)

[4. Design and Implementation 35](#_Toc528009687)

[4.1. Idea 35](#_Toc528009688)

[4.2. Architecture 36](#_Toc528009689)

[4.3. Programming 37](#_Toc528009690)

[4.4. Obtained Data 39](#_Toc528009691)

[4.5. Fingerprint 42](#_Toc528009692)

[4.5.1. Calculation 42](#_Toc528009693)

[4.5.2. Possibility to adjust 42](#_Toc528009694)

[4.6. Visualisation 43](#_Toc528009695)

[5. Evaluation 44](#_Toc528009696)

[5.1. Prototype testcases 44](#_Toc528009697)

[5.2. Evaluation of web browser fingerprinting methods 45](#_Toc528009698)

[5.3. Comparison to AmIUnique and Panopticlick 46](#_Toc528009699)

[6. Summary 47](#_Toc528009700)

[6.1. Result 47](#_Toc528009701)

[6.2. Prospects 47](#_Toc528009702)

[7. List of literature 48](#_Toc528009703)

[8. List of figures 49](#_Toc528009704)

1. Introduction
   1. Motivation

The internet plays a big role in our everyday life. It is used to search for information, stay in contact with other users and even used to shop for items. Most of the users do not waste another thought about who can see what they are doing. Nobody has time to read the endless general terms and conditions which have to be accepted to use most of the webservices nowadays. It is anchored in our mind that it is enough to switch to incognito mode in case we need some extra privacy, not thinking about all the parties which can still legally access our browser data (internet service provider, employer, …).

//look up how people really view privacy on the web

But then again, from time to time, there are scandals like Facebooks data leak to Cambridge Analytica in March 2018. Gears start to grind and for just one second our consciousness is directed towards the big stream of information which flows by continuously in form of the internet. This big stream gives users the feeling of privacy, because who would be interested in one specific user? But scandals like the one Facebook caused disrupt this philosophy. There are actually companies and people out there who are interested in this specific data. But blocking these third party cookies should be enough to secure our privacy – right? Unfortunately, no. Because since short before 2010 there has been a new technique which is able to re-identify users and therefore collect data about users. This technique can not be shut out or avoided. This is what makes it so terrifying. Thus a user can not protect his browsing habits completely there are still methods which help to mitigate the effectiveness of this technique. This technique is called web browser fingerprinting and this bachelor theses discusses how it works and what users can do to reduce its effectiveness.

* 1. Objectives
     1. Analysis of web browser fingerprinting techniques

One of the objectives of this bachelor thesis is to take a closer look as the different web browser fingerprinting techniques, which are currently in use.

First some of the configurations and plugins which are used most are introduced to give a better explanation why the browser leaks information and what it is necessary for. Subsequently the different methods of fingerprinting are explained and based on them the different techniques.

Based on the foundation set in these subchapters the techniques will be shown based on an application scenario.

The most important points which are discussed regarding this objective are:

* What it is used for
* If it poses a threat
* How to avoid it
* Some of the concerned configurations and plug-ins
* Fingerprinting methods
* Fingerprinting techniques
* Analysis on the basis of an application scenario
  + 1. Prototype

Due to the outcome of the analysis of the different web browser fingerprinting techniques the best technique for the prototype will be chosen. The objective of the prototype is to show how web browser fingerprinting is implemented and on a different level how it works. Based on different configurations and plug-ins the prototype will create a hash which will help to re-identify users.

The most important points regarding this objective are:

* Read configurations and plug-ins
* Create a hash
* Re-identify users
  1. Overview

Following chapters discuss the stated content:

[Chapter 2](#basis) 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](#requirement) on the other hand will list requirements which have to be fulfilled in order to implement a proper prototype

[Chapter 4](#design) explains the implementation and design of the prototype

[Chapter 5](#evaluation) concludes the previous chapter based on testcases and a proper evaluation

[Chapter 6](#summary) finally concludes the bachelor thesis, summarizes the results and states possible prospects

Chapter 7 contains the literacy referenced in this thesis.

Chapter 8 states all figures used.

1. Foundation
   1. Fingerprinting

When speaking of fingerprint, the average person usually thinks of our biometric fingers. Not aware that online activities can be tracked using a socalled fingerprint.

Web browser fingerprinting, also known as device fingerprinting, is the systematic collection of information to identify and later re-identify users. ([3], W3C, p. 3)([5], AmIUnique) This data tracking is possible due to the information provided by the web browser. ([2], R. Havens, p.1), ([1], R. Upathilake, p.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. This cookieless monster is what makes users uneasy, as any interested third-party can exploit this weakness without the knowledge of the user. ([5], AmIUnique)

* Describe what it is
* That it is used + 1 example
* That it poses a thread

Tool for online tracking and deanonymization attacks [2,3]

Widespread use for both digital marketing and fraud protection – but its capacity for online tracking combined with its ease of implementation and high reliability make this a dangerous tool for more maliciously-minded tracking programs[2,3]

* First developed short before 2010
* Quickly gained traction in the tracking industry
* Capacity to identify individual devices allowed trackers an extremely reliable method of correlating data across websites with individual users

[2, 3]

* Analysing a multitude of device-specific features and combining those metrics to create a fingerprint “hash” for that device.
* Backend server – keeps track of the fingerprint and all actions associated with it
* Fingerprint – unlikely to change significantly (based on features that are either immutable, hard to disguise, or extremely unlikely to change)

[2,4]

* Simple and easy to implement – extremely effective [2,10]

Princeton study – 2014-2016 decrease in the number of webpages running fingerprint analytics scripts -> traced the cause to the public backlash surrounding their initial report in 2014 [2, 11]

* Websites that runs the fp-analytics script will be able to track users through a shared data store [2,8]

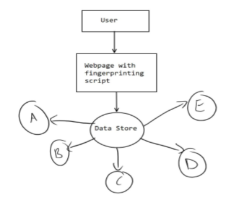
The use of a set of attributes obtained from the user’s browser, e.g. info about installed fonts or plug-ins, to uniquely identify a browsing environment without cookies or maintaining client side state

Browser fingerprinting can be used to deliver malware that is customized for a specific browser configuration, or even a particular user in a targeted host. This fingerprint can then be used to match against known execution environments in order to launch exploits specific to the host as well as to prevent execution inside an analysis system.[1,1]

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

* 1. Why is it needed?

Like many means of tracking, web browser fingerprinting has many kinds of utilization. Some more malicious than others. But no matter the cause, most people nowadays do not like the thought of being tracked.

The following paragraphs will list different categories in which web browser fingerprinting can have a

Constructive use

* Security authentication

Using web browser fingerprinting to correctly identify a device which is used to log into an account can be used to reidentify a user and help combat fraud or credential hijacking. [1,4], [13,2]

Example:

Whenever it is tried to access your google service account with a new device, google informs the owner of the account about the attempt.

Destructive use

* Hacking

Habits of users can be tracked without their knowledge and attackers can acquire specific knowledge about the user’s software setup [1,4]

Example:

With the help of web browser fingerprinting a hacker can acquire knowledge about the users operating system and adapt his hacking method accordingly.

Con- and destructive use

* Identifying criminals

As fingerprinting is the art of identifying and re-identifying users (see chapter 2.1) it is the perfect mean of tracking criminals. As far as they are criminals. While fingerprinting can help to track illegal business or fraud it can also threaten users which practice their right of free speech.

Example: positive use

When a user is harassed or stalked by another user the website can use web browser fingerprinting as a mean of blacklisting said user.

Example: negative use

Further it grants countries where for example the right of free speech is not ensured the means of blacklisting users who violate the countries policies.

* Commercial

Commercial Fingerprinting can be used to track people’s habits and preferences. On the one hand this can be used to show people items which match their preferences, while on the other hand it can be used to show them items which will match a certain price range.

Example: positive use

A website picks up the user’s habit to look for climbing gear and suggests similar products.

Example: negative example

A website registers that the user recently bought more expensive products and pends to showing items in this certain price range rather than the better product.

while the two most popular categories of sites are: “Pornography” (15%) and “Personals/Dating” (12.5%). For pornographic sites, a reasonable explanation is that fingerprinting is used to detect shared or stolen credentials of paying members, while for dating sites to ensure that attackers do not create multiple profiles for social-engineering purposes. Our findings show that fingerprinting is already part of some of the most popular sites of the Internet, and thus the hundreds of thousands of their visitors are fingerprinted on a daily basis.

[13, 6]

In a study conducted by Nikiforakis et al [13,6]

As just stated, web browser fingerprinting has a variety of uses. But which websites do use its functionality.

A study conducted by Nikiforakis et al [13], …

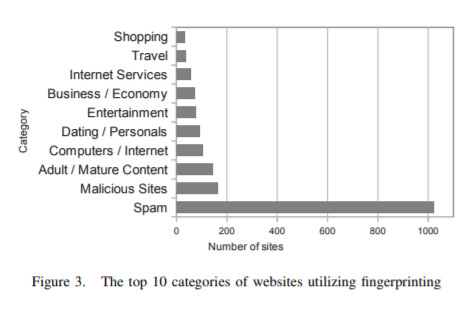
Top 10k websites webcrawled in their study

15% pornography websites (detect shared and stolen credentials)

12.5% personals/dating websites (ensure that multiple profiles are not created for social-engineering purposes)

([13], N. Nikiforakis)

ADD A GRAPHIC/STUDY WHICH AND HOW MANY WEBPAGES USE FINGERPRINTING!!!!

[13, 7]

* 1. 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.

\*/

Refer to destructive use in chapter 2.3

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]

* Deanonymization of Tor users

Tor enables JavaScript by default and if the user fails to turn it off, even Tor users can be tracked due to the information leaked by JavaScript. [2, 9f]

* 1. How to avoid it

Even nowadays people still appreciate privacy, so when there is the issue of web based device fingerprinting of course the question how to avoid it pops up. But that’s the clue. There is no way to avoid being fingerprinted a 100% except if you don’t use the internet at all. So when using the internet you have to accept the disadvantages which come with the advantages of fingerprinting as there are only ways to reduce your fingerprint

You can’t 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]

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.

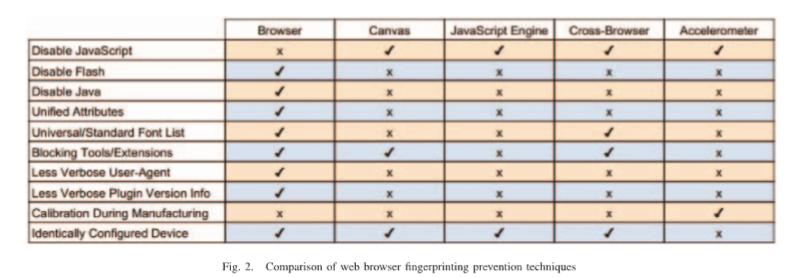
Accelerometer fingerprinting

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 computers. 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]

[1,5]

* 1. Critical configurations and plug-ins

As mentioned in the previous paragraph (chapter 2.1.) the scripts which use fingerprinting can read certain data via configuration settings or other observable characteristics from the web browser which enables them to create a specific fingerprint. ([3], W3C, p. 3)

The following paragraphs will introduce some of these certain information sources disclosed to third parties by the web browser and why they are used at all.

* + 1. User agent

A user agent is a request header field in the hypertext transfer protocol (http) which is used for the communication between browser and website. [9] It is automatically sent with each page call ([3], p. 6), except if it is specifically configured not to. [9] ([8], R. Fieldings, §5.5.3)

The user agent contains the name and version of the used browser. In R. Fieldings work about http, this combination is also called product identifier. The product identifiers are listed in the order or their importance, whereas each of them can be followed by one or multiple comments. [9] ([8], R. Fieldings, §5.5.3) This field value is often called user agent string. ([11], Hoffman)

Example 1:

Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.102 Safari/537.36 Vivaldi/2.0.1309.37

This string tells the receiving server, that the latest version (2.0.1309.37) of the Vivalid browser is used. In the comments (contained in the brackets) it gives away that the computer runs Windows 10 (Windows NT 10.0) on a 64-bit version (WOW64).

Example 2:

Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134

As well as with the other example, this browser (Edge with the version number 17.17134) also gives away the operating system and its version (same as before).

As shown in the examples above, not only the used browser name and its version are passed on but also information about the operating system and sometimes hardware. ([11], Hoffman) [9] The passed-on information is used to display different content in different browsers, on different operating systems, or to gather statics ([11], Hoffman) ([10], Arntz) [9]

If wanted, changes on the user agent string are easily made. ([10], Arntz)

The problem with this configuration is, that the longer the user-agent field values become, the higher becomes the risk of being identified through for example fingerprinting. This is the case why the user should limit the addition to the user-agent string which can be requested by third parties. ([8], R. Fieldings, $5.5.3)

* + 1. JavaScript

Js – the language which never works – for the developer

JS gives access to many browser-populated features like the plugins installed on the user’s device [5]

Identifying web browsers based on the underlying javascript engine

[12, M.M., 2013)

When JavaScript is disabled, websites won’t be able to detect the list of active plugins and fonts you use, and they also won’t be able to install certain cookies on your browser.

The disadvantage of disabling JavaScript is that websites won’t always function properly, because it’s also used to make websites run smoothly on your device. This will impact your browsing experience.[pixelprivacy]

Browser fingerprinting, while a powerful tool, is often part of a larger suite of tracking techniques. Fingerprinting operates through the js engine as part of an analytics program. Any website that runs the fingerprinting analytics script will be able to track users through a shared data store.

[2, 8]

* + 1. Adobe Flash

Adobe Flash is a browser plug-in which provides different ways of delivering rich media content which could traditionally not be displayed using HTML. [13,3]

* Criticized for poor performance, lack of stability
* Newer technologies (HTML5) can potentially deliver what used to be only possible through flash – still available on the vast majority of desktops [13,3]

* Returns all fonts with one simple call [2,5]
* Its rich programming interface (API) provides access to many system-specific attributes (version of OS, list of fonts, screen resolution, timezone) [AmIUnique]
* Can be disabled without a negative impact on UEX – only impacts browsing experience on old websites [prixelprivacy]
* Vulnerability, called “local storage objects” -> enables “zombie cookie” or “supercookie”. Almost impossible to remove. Populate numerous different storage areas with tracking information. Even if cookies deleted from browser -> still come back. Other storage locations repopulate cookie as long as any existis – combined with brower fingerprinting, the cookies can track changes in fingerprints while also relying upon the fingerprint as a finale line of defense against the most privacy-conscious users. “cookie syncing” [2, 8-9]
  + 1. 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]

* + 1. Others

Quickly describe which else there are:

[2, 5]:

Language – Language the browser is in; used for localization

Screen Resolution – the with and height of the user’s screen

// here reference to why ratio is better -> zoom

Timezone – the local timezone of the user’s computer

DoNotTrack – weather the user has indicated they wish to opt-out of tracking

Installed Fonts – the fonts supported in the browser, typically retrieved from Adobe Flash but can also be detected by other means

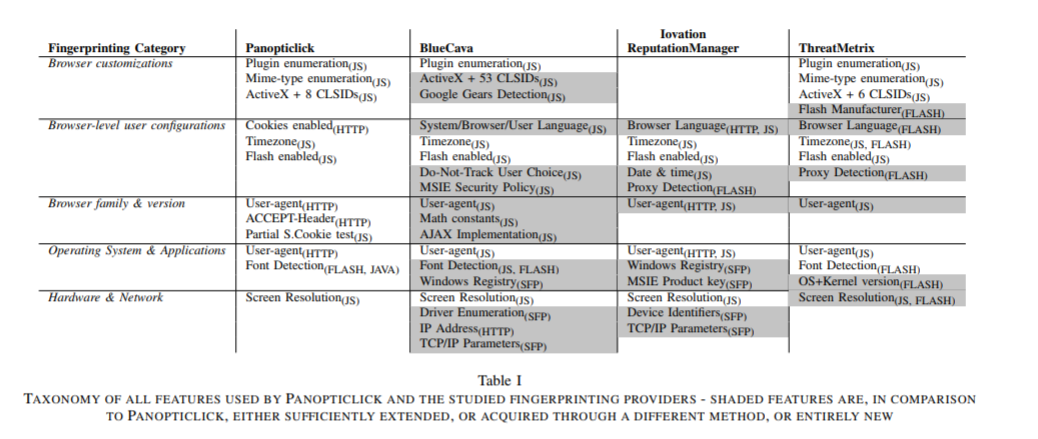
Canvas Fingerprinting – The program attempts to render an image on the webpage using the HTML5 canvas. Subtle differences in the rendering of the image can lead to identification

[2, 6]

Browser Plugins – a list of plugins installed in the browser

Cookies enabled – whether or not the user allows cookies on the page

Benchmark tests – evaluate how the browser’s js engine performs against a list of published bench marks. This information is often used to supplement that of the user agent string



[13,3]

* Probably which to block? Which are really needed?
  1. Web browser fingerprinting methods
     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]

* + 1. 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]

* + 1. 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]

* 1. Techniques of fingerprinting
     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]

* + 1. 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]

* + 1. 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]

* + 1. 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]

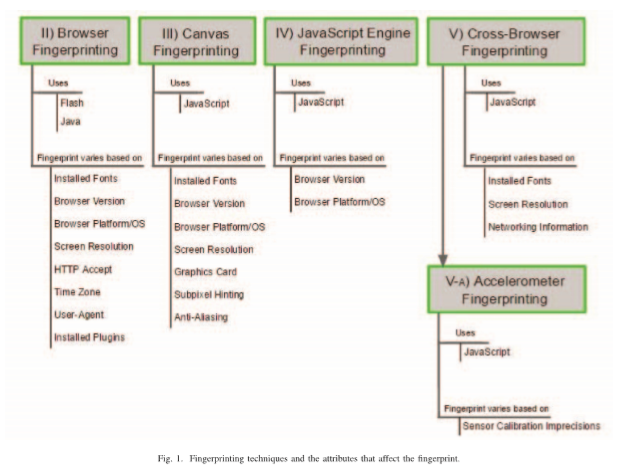
A technique developed by Yinzhi Cao

* New method, created by Yinzhi Cao
* E.g. used screen’s ratio of width to height as it remains consistent even when using zoom
* Adapted 4 such features from AmIUnique (which used screen resolution – not consistent)
* Examination of a user’s audio stack, graphics card, and CPU
* Relies on 29 features
* Script forces system to perform 36 tasks – results include info, takes less than a minute
* Only on Tor it didn’t work
* [6]
* Rely on corresponding OS and hardware functionalities
* 99,24% successfully identify
* http://yinzhicao.org/TrackingFree/crossbrowsertracking\_NDSS17.pdf
  + 1. 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]

* 1. Analysis based on an application scenario



* + 1. Introduction
    2. Browser specific fingerprinting
    3. Canvas fingerprinting
    4. JavaScript Engine fingerprinting
    5. Cross-Browser fingerprinting
    6. Accelerometer fingerprinting
  1. 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]

1. Requirement (Prototype)
   1. Use cases
   2. Functional requirement

Messing von x daten

Messing von y daten

Erstellen eines hashes

User wiedererkennen

Viasualisierung des wiedererkennens

* 1. Non functional requirement

1. Design and Implementation
   1. Idea
   2. Architecture
   3. Programming
   4. Obtained Data
   5. Fingerprint
      1. Calculation
      2. Possibility to adjust
   6. Visualisation
2. Evaluation
   1. Prototype testcases
   2. Evaluation of web browser fingerprinting methods
   3. Comparison to AmIUnique and Panopticlick

\*AmIUnique – 90,84% of identifying across browsers

\* 17 features included to observe

[6]

1. Summary
   1. Result

What do I figure from all that

How important security is, how fraguile or better said non existent anonymity/privacy online

* 1. Prospects

How could I improve the prototype

e.g. using heuristics to distinguish better

1. List of literature
2. #a) Randika Upathilake, Yingkun Li, Ashraf Matrawy, “A Classification of Web Browser Fingerprinting Techniques”, IEEE, 2015, <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7266460>

Upi15

1. #b) Ryan Havens, “Browser Fingerprinting: Attacks and Applications”, 2016, <http://www.cs.tufts.edu/comp/116/archive/fall2016/rhavens.pdf>

Havens16

1. #c) “Mitigating Browser Fingerprinting in Web Specifications”, W3C Editor’s Draft 21.05.2018, <https://w3c.github.io/fingerprinting-guidance/>

Doty18

1. #d) “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)>
2. #e) <https://amiunique.org/faq>

amiunique

1. #f) Amy Nordrum, “Browser Fingerprinting Tech Works Across Different Browsers for the First Time”, 2017, <https://spectrum.ieee.org/tech-talk/telecom/internet/new-online-fingerprinting-technique-works-across-browsers>

Nordrum17

1. #g) Yinzhi Cao, Song Li, Erik Wijmans, “(Cross-)Browser Fingerprinting via OS and Hardware Level Features”, 2017, <http://yinzhicao.org/TrackingFree/crossbrowsertracking_NDSS17.pdf>

Cao17

1. #h) Roy Fielding, J. Reschke, “Hypertext Transfer Protocol (http/1.1): Semantics and Content”, 2014, <https://tools.ietf.org/html/rfc7231#section-5.5.3>

Fielding14

1. #i) „User Agent“, <https://www.xovi.de/wiki/User_Agent>

Xovi18

1. #j) Pieter Arntz, “Explained: user agent”, 2017, <https://blog.malwarebytes.com/security-world/technology/2017/08/explained-user-agent/>

Arntz17

1. #k) Chris Hoffman, ”What is a Browser’s User Agent?”, 2016 <https://www.howtogeek.com/114937/htg-explains-whats-a-browser-user-agent/>

Hoffman16

1. #l) Martin Mulazzani, Philipp Reschl, Markus Huber, Manuel Leithner, Sebastian Schrittwieser, Edgar Weippl, “Fast and Reliable Browser Identification with JavaScript Engine Fingerprinting”, 2013, <http://www.ieee-security.org/TC/W2SP/2013/papers/s2p1.pdf>

Mulazzani13

1. #m) N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, G. Vigna, „Cookieless monster: Exploring the ecosystem of web-based device fingerprinting”, 2013, https://seclab.cs.ucsb.edu/media/uploads/papers/sp2013\_cookieless.pdf

Nikiforakis13

1. #n) <https://pixelprivacy.com/resources/browser-fingerprinting/>

Pixel18

1. #o) J. R. Mayer, “Any person... a pamphleteer,” Senior Thesis, Stanford University, 2009.
2. List of figures