Fachhochschul-Bachelorstudiengang
SOFTWARE ENGINEERING
A-4232 Hagenberg, Austria

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, Februar 2019

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.

Hagenberg, February 28, 2019

Janine Denise Mayer

Contents

Declaration i

Abstract iv

Kurzfassung v

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Objectives . . . 2
  - 1.2.1 Analysis of different web browser fingerprinting techniques . . . 2
  - 1.2.2 Prototype . . . 2
- 1.3 Overview . . . 2

2 Foundation 3
- 2.1 Fingerprinting . . . 3
- 2.2 Utilisation . . . 4
- 2.3 Threat . . . 6
  - 2.3.1 Under General Data Protection Regulation . . . 7
- 2.4 Mitigation . . . 8
- 2.5 Critical technologies . . . 9
  - 2.5.1 JavaScript . . . 9
  - 2.5.2 Adobe Flash . . . 10
  - 2.5.3 HTML 5 . . . 11
  - 2.5.4 User agent . . . 12
  - 2.5.5 Others . . . 13
- 2.6 Web browser fingerprinting methods . . . 14
  - 2.6.1 Active fingerprinting . . . 14
  - 2.6.2 Passive fingerprinting . . . 14
- 2.7 Web browser fingerprinting techniques . . . 15
  - 2.7.1 Browser specific fingerprinting . . . 15
  - 2.7.2 Canvas fingerprinting . . . 16
  - 2.7.3 JavaScript Engine fingerprinting . . . 17
  - 2.7.4 Cross-Browser fingerprinting . . . 18

3 Analysis based on an application scenario 20
- 3.1 Introduction . . . 20
- 3.2 Browser specific fingerprinting . . . 21

3.3 Canvas fingerprinting . . . 21
3.4 JavaScript Engine fingerprinting . . . 21
3.5 Cross-Browser fingerprinting . . . 21
3.6 Accelerometer fingerprinting . . . 21
3.7 Comparison . . . 21

4 Requirements **22**
4.1 Use cases . . . 22
4.2 Functional requirements . . . 22
4.3 Non functional requirements . . . 22

5 Design and Implementation **23**
5.1 Idea . . . 23
5.2 Architecture . . . 23
5.3 Programming . . . 23
5.4 Obtained Data . . . 23
5.5 Fingerprint . . . 23
5.5.1 Calculation . . . 23
5.6 Visualisation . . . 23

6 Evaluation **24**
6.1 Prototype testcases . . . 24
6.2 Comparison to AmIUnique and Panopticlick? . . . 24

7 Summary **25**

References **26**
Literature . . . 26
Online sources . . . 27

List of Figures **29**

Abstract

This should be a 1-page (maximum) summary of your work in English.

Kurzfassung

An dieser Stelle steht eine Zusammenfassung der Arbeit, Umfang max. 1 Seite. ...

Chapter 1

Introduction

1.1 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 to shop for items. Most of the users do not waste another thought on who could 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 services online. 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 like the internet service provider or at work the 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 was 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 (or point of view). There are actually companies and people out there who are interested in this specific data. But blocking 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 is called *web browser fingerprinting* and can not be shout out or avoided. This is the reason which 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 bachelor thesis will discuss what exactly this tracking method is, how it works and explain what users can do to reduce its effectiveness. perceptions of people about their anonymity (Mayer [10], p.17)

1.2 Objectives

1.2.1 Analysis of different 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:

- used configurations and plug-ins
- fingerprinting methods
- fingerprinting techniques
- analysis of the techniques on the base of an application scenario

1.2.2 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 how it works. Based on different configurations and plug-ins the prototype will create a hash which will help to re-identify users.

The prototype will be in form of a website which will include a fingerprinting script. It's functionality will cover following points:

- reading configurations and plug-ins
- creating a hash
- re-identifying users

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 for chapter 4.
Chapter 3 will specify the requirements which are needed to outline a proper prototype.
Chapter 4 explains the implementation and design of the prototype.
Chapter 5 concludes the previous chapter based on test cases and a proper evaluation.
Chapter 6 sums up the bachelor thesis, summarizes the results and states possible prospects.

Chapter 2

Foundation

This chapter will cover the basic knowledge about fingerprinting. This chapter is all about the what, why and how of fingerprinting. First explaining the term fingerprint and what it can be used for. Further it will discuss the threat is poses as well as possible ways to reduce a fingerprint before covering the used configurations and plug-ins. After the base was layed the chapter closes with the descriptions of the different fingerprinting methods and techniques.

2.1 Fingerprinting

Web browser fingerprinting, also known as device fingerprinting, is the systematic collection of information to identify and later re-identify users. (Doty [5], p.3)(AmIUnique [18]) This data tracking method works by obtaining data from the users web browser and using this data to create an unique fingerprint "hash". This hash is the key for later re-identifying the user. (Upathilake, Li, and Matrawy [16], p. 1)(Havens [8], p. 4)

With millions of computers and mobile devices in use and even more browsers, this identification method seems really unlikely to work at first sight. Nevertheless, it is an extremely reliable method for correlating data across websites or even web browsers with individual users. (Havens [8], p. 3)

Here is how it works:
There are currently over 7.5 billion people populating our planet. Imagine you have to identify a single person. By stating this specific person is male, you already cut the choice by half. Information about the ethnicity and age of this person as well as the time zone he lives in will narrow down the choice even more.
It's basically the same with web browser fingerprinting. The gender can be seen as an equivalent for the used operating system, the ethnicity as the browser and age as the browsers version. Information about the time zone and used language are easily acquirable by the browser and so are more specific details which help narrowing down the choice to nearly always one person.

This combination of properties can be obtained using code (time zone, screen resolution,

plug-ins) and also be extracted from HTTP headers (user agent string). (Szymielewicz and Budington [15])

Web browser fingerprinting was first developed roughly before 2010 and due to its effectiveness quickly gained traction in the tracking industry.(Havens [8], p. 3)
Many of the fingerprinting methods can not be detected by the user and as it is extremely difficult for users to modify their web browsers in a way that they are less vulnerable to it (AmIUnique [18])(Szymielewicz and Budington [15]), makes it a dangerous tool for deanonymization attacks and maliciously-minded tracking programs. (Havens [8], p.3)

> Example
> Upon accessing a website, the fingerprinting script determines the users fingerprint and pushes it to a central data store. This data store is usually a shared data store which is used by multiple websites. So when the user accesses another website which also has access to this shared data store, the user will be recognized and the user profile can be updated. (Havens [8], p. 8)

2.2 Utilisation

In a study conducted in 2013 by Nikiforakis et al. they found that fingerprinting is a part of some of the most popular websites on the internet and that multiple hundred thousands of users are fingerprinted on a daily basis. (Nikiforakis et al. [22], p.6)

The following paragraphs lists different ways in which fingerprinting can be utilized:

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 re-identify a user and help combat fraud or credential hijacking. (Upathilake, Li, and Matrawy [16], p.4)(Nikiforakis et al. [22], p.2)

  > Example:
  > Services can track the devices used to access an account and inform the user in case an unknown device tries to access it.

Destructive use

- Hacking
  Habits of users can be tracked without their knowledge and attackers can acquire specific knowledge about the user's software setup. (Upathilake, Li, and Matrawy [16], p.4)

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

Con- and destructive use

- Identifying criminals
  As a mean of tracking, fingerprinting can be used to track people or even criminals. But as seen in the previous examples this can used for and against users.

  > 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
  > Citizens of countries with a strict regime can be blacklisted for expressing criticism against the government.

- Commercial use
  Commercial Fingerprinting is used to track people's habits and preferences. The acquired knowledge can be used to either help the user or direct him into a certain direction.

  > Example: positive use
  > A website picks up the user's habit to look for climbing gear and suggests similar products.

  > Example: negative use
  > A website registers that the user recently bought more expensive products and starts to only show items in this certain price range rather than more suitable but cheaper products.

The following paragraphs will summarize the findings of a study conducted by Nikiforakis et al. in cooperation with three commercial fingerprinting companies in order to test multiple websites for the use of fingerprinting scripts.

Nikiforakis et al. used the categorization of TrendMicro and McAfee on a list of 3804 domains. If a domain was neither analysed nor categorized by the both used services they were marked as untested and not used. Therefore only 59.2% of the 3804 domains were included in the results of this testing. If one service declared a domain as unsafe and the other stated the opposite, it was accepted as safe. Some categories were given aliases and

gathered together to a more generalized category.(Nikiforakis et al. [22], pp.6)

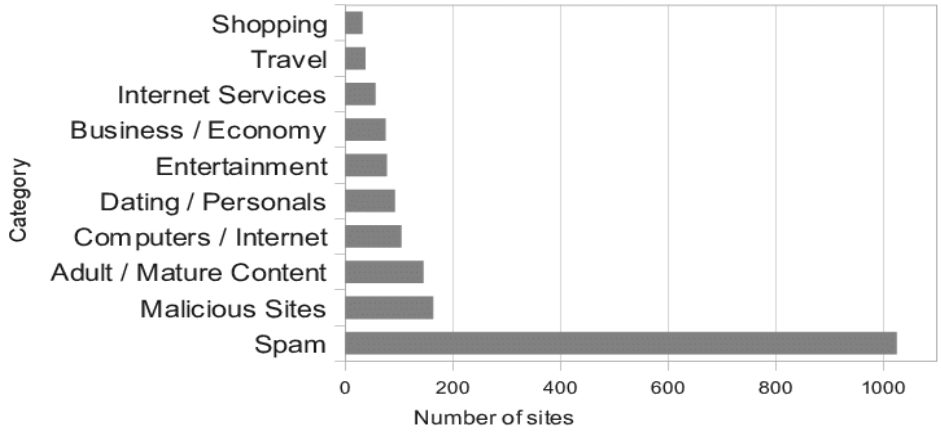

Figure 2.1: Top 10 categories utilizing fingerprinting (N. Nikiforakis [13])

This figure shows 10 categories, 8 of which operate with user subscriptions. These sites usually are interested in preventing hijacking of user accounts and fraudulent activities. The top two categories were identified as malicious (163 domains) or as spam (1063 domains). Nikiforakis et al. noticed that many domains belonging to one of these categories were parked websites which do not include any fingerprinting code. Further many "quiz" or "survey" websites were located which in a later step extract a user's personal details. The domains were categorized as malicious if they exploit vulnerable browsers or extract private data from users.

It was discovered that all these sites were found to include, as some point, fingerprinting code provided by one of the three commercial fingerprinting companies.
This observation, paired with the fact that all of these three studied providers stated that there must be set an appointment in order to acquire fingerprinting services, point to the possibility that fingerprinting companies work with dubious domains in order to expand their fingerprinting database.(Nikiforakis et al. [22], p.7)

2.3 Threat

As seen in the previous section 2.2 web browser fingerprinting can be utilized in destructive ways. This section will have a closer look at the threat fingerprinting poses for the users' privacy and security.

The most tricky part about fingerprinting is that on the one hand users usually do not know that they are being tracked and on the other hand they can't do anything to prevent it (see section 2.4). (Upathilake, Li, and Matrawy [16], p.4)

So for example P. Eckersley was able to generate unique fingerprints for 83.6% of 470,161 browsers who belong to users who have a high awareness of privacy concerns. (Eckersley [6], p.2,8)

Even though the fingerprint can change due to modified characteristics there is a simple algorithm to detect these changes. With heuristics this algorithm was able to re-identify a changed fingerprint with 99.1% accurancy out of 65.56% guesses of changed fingerprints.(Eckersley [6], p.13)

Below are a few threats listed which fingerprinted users might face:

- due to the possibility of being identified some users might face surveillance, risk their personal physical safety or concerns about discrimination against due to their internet activities. (Doty [5], p.4)
- the tracking enables collection without clear indications that such a collection is happening without clear or effective user controls. (Doty [5], pp.4)
- 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. (Havens [8], pp.9)

2.3.1 Under General Data Protection Regulation

This threat posed by web browser fingerprinting as depicted above might be reduced by the *General Data Protection Regulation (GDPR)*, which entered into force on May 25th 2018. This regulation imposed by the European Union (EU) intends to cover exactly this kind of hidden data collection which is used by web browser fingerprinting, by forcing companies to prove they have a legitimate reason for the utilization of any means of tracking. (Szymielewicz and Budington [15])

Even though the GDPR avoids specifying technologies it provides general rules which should keep up with technological development. Due to this regulation browser characteristics are now to be treated like personal data. Personal data has a broad definition as any information that might be linked to an identifiable individual can be passed as such. Examples for such are not only the IP-Address and MAC-Address of users but also less specific features, including the combination of characteristics which web browser fingerprinting relies upon. (Szymielewicz and Budington [15])

In order to be allowed to use fingerprinting legally the concerned entity has to complete the following steps:

- show that the tracking does not violate "the fundamental rights and freedoms of the data subject, including privacy",
- and is in line with "reasonable expectations of data subjects"
- further give a legitimate argument for its interest in tracking,
- and share details about the scope, purposes, and legal basis of the data processing with the person subjected to the fingerprinting

Due to this regulation, the only step the user has to take to avoid fingerprinting is to say "no".

Even though the rules imposed by this regulation seem to help prevent unwanted tracking, it only helps mitigate it. There will certainly be fewer entities making use of fingerprinting

though web browser fingerprinting is not expected to disappear, no matter how high the penalties are on its illegal utilisation.

Anyway, the GDPR only applies on processed personal data of individuals living in the *European Economic Area (EEA)* for commercial purposes, or for any purposes when the behaviour is within the EEA. There will always be companies which either think they can escape the consequences or which claim to have "legitimate interest" in tracking users. Further, no matter how strict regulations are, it always comes down to the user. Due to the plentiful requests for consent as they are found on the web nowadays many users are worn out and do not take a second look at the privacy policy regulated by the GDPR. (Szymielewicz and Budington [15])

2.4 Mitigation

As seen in the previous chapters there are many ways of utilizing web browser fingerprinting. Many of these possible applications pose a threat to the users privacy which brings up the question if it can be avoided. Each of the researched papers about this topic states the same - No. Unfortunately, it is impossible to opt-out fingerprinting. (analytics [2])(Upathilake, Li, and Matrawy [16], p.4)

Even if it is not possible to prevent the possibility of being fingerprinted there are many suggestions on how to at least mitigate the characteristics which are used to generate an unique fingerprint. Some of these suggestions are stated below:

- Blocking tools which are maintained by regular web-crawls to detect tracking and incorporate blocking mechanisms (Acar et al. [1], pp.11);
  This might work with active fingerprinting methods but not with the passive methods.
- Having flash provide less system information and report only a standard set of fonts (Nikiforakis et al. [22], p.13);
  This would require flash developers to take enough interest in this topic to change security settings as well this might cause some rendering problems.
- Use private browsing modes or Tor anonymity service;
  Tor is slower and while it disables WebGL it still allows rendering to a <canvas> which partly exposes the user to canvas fingerprinting. (Mowery and Shacham [11], p.1)
- Browser vendors agree on a single set of API calls to expose to the web applications as well as internal implementation specifics (Nikiforakis et al. [22], p.13);
  This would require all browser vendors to take enough interest in this topic to change their API calls.
- Browser vendors agree on an universal list of "canvas-safe" fonts (Mowery and Shacham [11], p.10)(Boda et al. [3], p.16);
  This would again require cooperation of all browser vendors. If only a few browsers adapt their settings this might make them more distinct.
- Support WebGL which ignore graphic cards and render scenes in a generic software renderer;

This approach might be good while the performance impact is not. (Mowery and Shacham [11], p.10)

- Require the users approval whenever a script requests pixel data (Mowery and Shacham [11], p.10);
  The reason why this might be can be concealed and the casual user will not know the effects of accepting
- Modification of the user agent string;
  Opinions vary on this ones. While Yen et al. claim that this measure helps, Nikiforakis et al. as well as other authors state otherwise. (Yen et al. [17], p.5) (Nikiforakis et al. [22], p.13) (Eckersley [6], p.4)
- Decrease the verbosity of plug-in versions and the user agent string (Boda et al. [3], p.16);
  This measurement might help to reduce the uniqueness of said characteristics.
- Enable the user to set fake system properties in browser like hiding the operating system, time zone and screen resolution (Boda et al. [3], p.16);
  This measurement might increase the fingerprintability if the real properties leak or are maliciously set.
- Browser extension which automatically blocks active content e.g. NoScript, ScriptBlock (analytics [2])
  Helps against canvas fingerprinting but does not prevent passive methods.
- Use commonly-used web browsers and default settings, including the operating system (analytics [2]);
  Good suggestions as no fingerprinting technique is able to distinguish identically configured devices

2.5 Critical technologies

As mentioned in the previous sections 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. (Doty [5], p.3) The following paragraphs will outline some of these configurations.

2.5.1 JavaScript

JavaScript is a web development language for web functionality. (Wood [26]) It allows client-side scripting and gives access to many browser-populated features like the installed plug-ins. (AmIUnique [18]) It is enabled by default in all major browsers and by November 2018 JavaScript was included by 95% of all websites online. (javascript.info [21]) (w3techs [25])

JavaScript can be executed on any device which has a *JavaScript Engine*. (Mulazzani et al. [12], p.2) (javascript.info [21]) This engine is responsible for parsing the script to compile it into machine language which can be exeucted. During this process the engine applies optimizations on every state of the process. (javascript.info [21])

Known JavaScript Engines:

- V8 in Chrome and Opera
- SpiderMonkey in Firefox
- ChakraCore for Microsoft Edge

JavaScript Engines usually implement features which are specified by *ECMAScript*.(javascript.info [21]) (Mulazzani et al. [12], p.2)
ECMAScript provides rules, details, and guidlines which have to be regarded for a scripting language in order to be ECMAScript compliant. It is represented by ECMA-262 which is a standard published by Ecma International (responsible for creating standards for technologies). (Aranda [19])

In-Browser JavaScript is seen as "safe" programming language as it only has a low-level access to memory or CPU. Its abilities are therefore limited and it has no direct access to OS system functions. (javascript.info [21])

JavaScript can also be used to retrieve browser characteristics like the user's time zone and system color. Further there are two properties which can be used to acquire additional information about the user's screen and window.(analytics [2])

The *navigator object* is a possible property for window objects and supported by all major browsers. Informations which can be acquired through the navigator object include: browser name and version, if cookies are enabled, browser language, platform and the useragent (which does not differ from the user agent in subsection 2.5.4)(analytics [2])

The *screen* is a property which holds information about the user's screen. For example the screen width and height, the acutal available display width and height and the color depth which is supported by the user's device.(analytics [2])

When JavaScript is disabled, websites are not able to detect the list of active plug-ins and fonts, and neither will be able to install certain cookies on the targeted browser. The disadvantage of disabling JavaScript is that websites won't always function properly, because it's also used to make websites run smoothly and therefore will impact the browsing experience.(pixelprivacy [23])

2.5.2 Adobe Flash

Adobe Flash is a proprietary browser plug-in which provides different ways of delivering rich media content which is usually not displayed in HTML. (Nikiforakis et al. [22], pp.2)
It used to be the most commonly used video format on the internet but can now be replaced with safer technologies like HTML5. Eventhough it could be replaced it is still installed on the vast majority of browsers.(analytics [2])(Nikiforakis et al. [22], p.3)

This plug-in can be used to retrieve sensitive data from the user for example can it

detect HTTP proxies. Besides, has it reimplemented certain APIs which already exist in browers and are accessible through the use of JavaScript. (Nikiforakis et al. [22], p.2) This rich programming interface (API) can be used to access system-specific attributes, such as the operating system and its version, a list of fonts, screen resolution and time zone.(AmIUnique [18])(Havens [8], p.5) Furthermore, the API does not provide the same result as the original browser API, but even more detailed return values. (Nikiforakis et al. [22], p.2)

> Example:
> Firefox might return “Linux x86 64” if it is queried about the platform execution while Flash might provide the full kernel version "Linux 3.2.0-26-generic". (Nikiforakis et al. [22], p.2)

Apart from these security risks is Adobe Flash also being criticized for its poor performance, as well as lack of stability. (Nikiforakis et al. [22], p.3) Another vulnerability is its local storage objects which enable so called *zombie cookies*. These zombie cookies populate in various storage locations on the targeted device and re-populate in case one of these storage areas is cleared. These cookies can also be used paired with fingerprinting in order to keep track of changes in fingerprints ("cookie syncing"). (Havens [8], p.8f)

Adobe Flash can be disabled or even uninstalled which might effect the user experience on old websites negatively. (pixelprivacy [23])

2.5.3 HTML 5

HTML5 is the latest version of the Hypertext Markup Language which is a coding language used to build websites and supported by all major browsers. (pixelprivacy [23])(**marshal17**) It can be used to build complicated applications, as well as provide animations, movies, music, and the likes. This without the need of any additional software (plug-ins). Summed up, it provides features like geolocation, graphics, video and webapps. (**marshal17**)

Apart from its incredible capabilities there is a slight problem concerning HTML5 video. As it would not be agreed upon which format(s) should be supported which ended in each browser supporting different HTML5 video formats.
As HTML5 does not include digital rights management technology which is used to prevent the user from copying the content, content owner usually prefer Flash. (**marshal17**)

One of HTML5’s new elements is <canvas> element which is used to draw graphics on a web page.(pixelprivacy [23]) Through this process it is possible to acquire any differences in the hardware and software configurations due to slight image rendering differences. (AmIUnique [18]) This provides a way to inspect the data with pixel accuracy which later can be used in fingerprinting techniques like described in subsection 2.7.2 (Mowery and Shacham [11], p.2)

In subsection 2.7.2 *Base64 encoded images* are used which is one of the oldest ways to encode something in html. Base64 encoded images are a part of the HTML and displays it without loading. It turns different types of data into a combination of numbers and letters which is safe for HTML. One of the main benefits is that the webpage does not have to load an external ressource which helps the page to load faster. This performance benefit only works if small images are used. (Sexton [24])

2.5.4 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. (Xovi [27]) It is automatically sent with each page call, except if it is specifically configured not to. (Xovi [27])(Fielding and Reschke [7], p.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. (Xovi [27])(Fielding and Reschke [7], p.5.5.3) This field value is often called user agent string. (Hoffman [9])

> 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. (Hoffman [9]) (Xovi [27]) The passed-on information is used to display different content in different browsers, on different oper-ating systems, or to gather statics (Hoffman [9]) (Arntz [20]) (Xovi [27])

If wanted, changes on the user agent string are easily made. (Arntz [20]) 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. (Fielding and Reschke [7], p.5.5.3)

2.5.5 Others

Quickly describe which else there are: (Havens [8], p.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

(Havens [8], p.6) Browser plug-ins – a list of plug-ins 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

WebGL

2.6 Web browser fingerprinting methods

2.6.1 Active fingerprinting

Active fingerprinting actively queries information about the client by running JavaScript code or using plug-ins like Adobe Flash which extend the browsers functionality.(analytics [2])((Doty [5], p.6)

Some of the additional characteristics which can be retrieved with this method are:

- user's screen (width, height, resolution) and window size
- enumerating fonts
- time zone
- plug-ins
- evaluating performance characteristics
- rendering graphical patterns

The only potential disadvantage active fingerprinting provides is that it can be detected on the client side in case the user analyses the outgoing data packages, HTML or the JavaScript source code, which in the majority of cases does not happen. (Doty [5], p.6)(analytics [2])

Active fingerprinting techniques

- Canvas fingerprinting
- JavaScript Engine fingerprinting
- Cross-Browser fingerprinting
- Browser specific fingerprinting

2.6.2 Passive fingerprinting

In contrast to active fingerprinting methods, passive fingerprinting does not need to execute any code on the client side. This method works based on observable characteristics which is contained in the header data of the IP packet by default. (Doty [5], p.6)(analytics [2]) There is no way for the user to learn if a website is using a passive fingerprinting method and secretly storing data.

Some of the provided characteristics are:

- cookies
- http request headers
- ip address and port
- character sets (e.g. UTF-8) and languages

Passive fingerprinting techniques

- Browser specific fingerprinting

2.7 Web browser fingerprinting techniques

As described in section 2.6 there are different methods of fingerprinting. These methods are implemented in different techniques which will be discussed in this chapter.

2.7.1 Browser specific fingerprinting

In 2010, P. Eckersley conducted a study in the project "Panopticlick" which checks how trackable users are. The tracking is done with the help of a fingerprint which is generated from previously collected data. (Upathilake, Li, and Matrawy [16], p.2) This study has shown that 94.2% of the users which use Flash or JavaScript could be distinguished with browser fingerprinting. (Eckersley [6], p.2)

How it works

Browser specific fingerprinting is a technique which only uses browser-dependent features to collect a number of common or and less-common known browser characteristics.

This kind of fingerprinting is possible even if the browser only reveals so much as its version and configuration information which is in most cases enough to create a distinct fingerprint. (Eckersley [6], p.16) Even though the retrieved browser information typically includes the installed fonts, plug-ins (including version numbers), user agent, http accept and screen resolution. (Upathilake, Li, and Matrawy [16], p.2)

As this kind of fingerprint pracitcally is just a concatination of this information it is easily affected by changes, like upgrades, newly installed features or extermal change of the system. Usually, simple heuristics can be used to adjust the browser specific fingerprint accordingly.(Eckersley [6], p.4) (Upathilake, Li, and Matrawy [16], p.2)

In some cases when the user tries to hide browser features, this accomplishes just the opposit, as it makes the fingerprint more distinct. See the example below.

> Example:
> Even when using a flash blocking add-on, the browser still displays flash in the plug-in list. When due to the blocking add-on the list of system fonts can't be obtained this creates a distinct fingerprint even though the flash blocker was not explicitly detected. (Eckersley [6], p.4)

Like many other web browser fingerprinting techniques, browser specific fingerprinting is not able to distinguish instances of identically configured devices. Further, other than cross-browser fingerprinting, this is a single browser fingerprint technique, which means it can not re-identify a user in another browser as the fingerprint may change across browsers. Upathilake, Li, and Matrawy [16], p.2)

2.7.2 Canvas fingerprinting

Canvas fingerprinting was first mentioned and implemented by Mowery and Shacham in 2012 as part of their work "Pixel Perfect: Fingerprinting Canvas in HTML5".
Any website that runs JavaScript on the user's browser can use the HTML5<canvas> element and observe its rendering behaviour. There is no need for any special access to system resources. (Mowery and Shacham [11], p.1)(Upathilake, Li, and Matrawy [16], p.2)

The outcome of their research and tests was that the canvas fingerprint is consistent but would change depending on hardware and software configuration. Only 2 years after its first use, a study conducted by Acar et al. ascertained that canvas fingerprinting is the most common form of fingerprinting with an approximately occurrence of 5.5% in the top 100,000 Alexa websites. (Acar et al. [1], p.5)

How it works

Canvas fingerprinting renders text and WebGL scenes onto an area of the screen using the HTML5<canvas> element programmatically. Each browser renders differently which helps acquiring a distinctive part for the fingerprint. So for example out of 300 samples the text font Arial was rendered in 50 distinctive ways.(Mowery and Shacham [11], p.6)

After rendering the text or WebGL scene using HTML5<canvas> the pixel data is read to generate a fingerprint. Through this process a 2D graphic context is obtained and a text or image is drawn to the canvas.(Upathilake, Li, and Matrawy [16], p.2)
The obtained canvas object provides a to-DataUrl(type) method which gives a data url consisting of a Base64 encoding of a png image which contains the canvas' content. This Base64 encoded pixel data url is used to create a hash.(Mowery and Shacham [11], p.3)(Upathilake, Li, and Matrawy [16], p.2)

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.(Upathilake, Li, and Matrawy [16], p.2)

In their paper K. Mowery and H. Shacham used two types of image comparisons to check if the user's configuration matches a fingerprint: *pixel-level difference* and *difference maps.*(Mowery and Shacham [11], pp.4)

Pixel-level difference works by setting each pixel's color to a specific color. If the pixel's color isn't a transparent pure black it indicates that two pixels don't match and therefore its alpha value is set to 255 to render it completly opaque. (Mowery and Shacham [11], pp.4)

Difference maps works in a similar way, only that this method sets a pixel either black or white, depending on if they differ. If an image is purely white it indicates that they are the identical image. (Mowery and Shacham [11], pp.4)

Canvas fingerprinting can not distinguishing identically configured devices and can not

fingerprint a user across different browsers.(Mowery and Shacham [11], p.5) Upathilake, Li, and Matrawy [16], p.2)

2.7.3 JavaScript Engine fingerprinting

In 2011 P. Reschl et al. published their first paper about the use of the JavaScript Engine to uniquely identify a web browser and its version. Comparing it to the Panopticlick project (Eckersley [6]) it was stated in the paper that the new approach is multiple times faster and error prone. (Reschl et al. [14], p.2) Two years later, in 2013, the same team with some additional authors published another paper in which this fingerprint technique was described in more detail.(Mulazzani et al. [12], p.1)

Compared to other methods this technique does not rely on the user agent string (which can be modified, see subsection 2.5.4) which makes it a more robost fingerprinting technique.(Mulazzani et al. [12], p.1)

How it works

JavaScript Engine fingerprinting works by running test cases of conformance tests like Sputnik and test 262 (see subsection 2.5.1) to identify browsers and their major versions. Even though these test suits consist of multiple thousands test cases the browser only needs to fail one particular test which every other browser passed to be identified. Therefore these fingerprinting technique only needs a fraction of a second to be executed. (Mulazzani et al. [12], p.3)

There are two methods which can be used for identification:

For a rather small test set the *minimal fingerprint method* can be used. First test262 is started for each browser in the set and the results are compared. For each browser a test is selected which only this specific browser failed (uniqueness = 1). This browser can be uniquely identified by this test and is therefore removed from the set. This process is repeated for each browser until there is a unique fingerprint for each browser in the set or there is no test case which was failed by only one browser. If there is no unique fingerprint for a browser the selection is simply changed.(Mulazzani et al. [12], p.3)

Figure 2.2 shows how the uniqueness of browsers and tests is obtained. The check marks mean, that the respective browser has passed the test. Each test which has a uniqueness of 1 can be used to uniquely identify a browser.

Web Browser	Test 1	Test 2	Test 3	Test 4
Browser 1	✓	×	×	×
Browser 2	×	✓	×	✓
Browser 3	×	✓	✓	×
Uniqueness	**1**	**2**	**1**	**1**

Figure 2.2: Example for the structure of a minimal fingerprint

If the test set is rather large it is more effective to use a *binary decision tree*. This method runs multiple test rounds to determine a web browser and major version. There is no need for a unique fingerprint as the tests split the browsers to identify them. Therefore there is no need to run a test for each browser and version like the minimal fingerprint would which reduces the total number of executed tests. (Mulazzani et al. [12], p.4)

Figure 2.3 gives a good overview how such a tree can be structured. The leaf nodes pose as browsers, the inner nodes display the used tests and the edges indicate the success. (Mulazzani et al. [12], p.4)

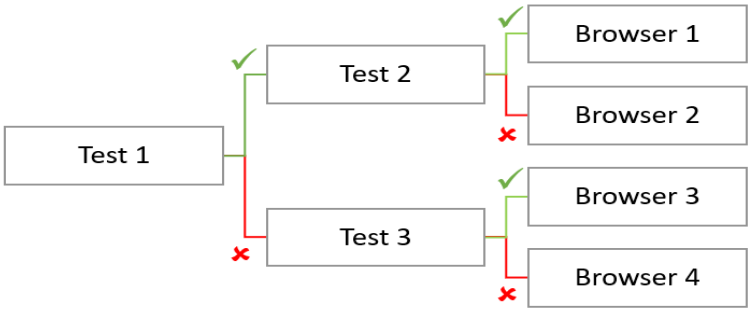

Figure 2.3: Example for the structure of a decision tree

2.7.4 Cross-Browser fingerprinting

In 2011 Boda et al. conducted the first study about cross-browser fingerprinting, utilizing a part of the user's IP address as part of the fingerprint. Even though this technique is not optimal due to the fact that the IP address can be modified by e.g. proxies it was the only paper published about cross-browser fingerprinting until Cao et al. published an improved version in 2017. (Boda et al. [3], p.14)(Cao, Li, and Wijmans [4], p.1)

How it works

Cross-Browser fingerprinting uses browser-independent features to be able to track a user regardless of what browser they are using. This can be done by relying only on the use of JavaScript and font detection. (Upathilake, Li, and Matrawy [16], pp.2) The fingerprint is created from specific browser data with the help of JavaScript and server-side algorithms.(Boda et al. [3], p.4) To be able to track a user across different web browsers a set of browser-independent features is used as a basis of identification. Part of this set is for example (Boda et al. [3], p.2)

- networking information (ip address, hostname, TCP port number)
- application layer information (user agent string, name and version of the operating system, extensions)
- information gained by quering (list of fonts, screen resolution, timezone, plug-ins and their version)

The difference to other fingerprints is that the first two octets of the IP address are used, which usually remains constant even if the IP address of the client changes dynam-

ically. The downside is that it may change when switching ISP or other service (Boda et al. [3], p.5) uses locality, short user id(script-generated identifier, derived from the first two octets of the IP address), usa, os, screen, timezone, basic fonts, all fonts, browser name and version (Boda et al. [3], p.5) The downside of using the IP address as part of the fingerprint is that proxies modify it for privacy reasons. Boda et al. dealt with that problem saying that the other parameter suffice to identify a user.

Cao et al. improved this method by excluding the IP address, modifying the use of some data and adding some features.(Cao, Li, and Wijmans [4], p.1) This technique forces the system to perform 36 tasks which return the required information within less than a minute. This fingerprinting technique is based on different novel operating system and hardware level features (e.g. from graphic card, CPU, audio stack, and installed writing scripts). Many of which are exposed to JavaScript over web browser APIs, which are used to extracted them.(Cao, Li, and Wijmans [4], pp.1) 99,24% successfully identify users 83.24% uniqueness with 91.44% cross-browser stability(Cao, Li, and Wijmans [4], p.2) compared to Boda excluding ip address -68.98% uniqueness with 84.64% cross-browser stability. Improvements towards Boda et al. technique. E.g. Change of the screen resolution. In browsers like Firefox and Internet Expolorer the resolution changes when scrolling. Therefore this method takes the zoom level into consideration and normalizes the width an dheight of the screen resolution. E.g. DataUrl + JPEG are unstable across different browsers, therefore use a lossless format like PNG (Cao, Li, and Wijmans [4], p.2) Various rendering tasks (e.g. drawing curves and lines to client side) are sent. Also obtains operating system and hardware level information. The client side browser renders and returns the results (images, sound waves) which are converted into a hashes. In the meantime the browser collects browser-specific information. FP composition - Fingerprint is generated from list of hashes which is intertwined with a mask through a "and" operation which creates the fingerprint hash. The collected browser information corresponds to the mask. Each browser has its own mask. (Cao, Li, and Wijmans [4], p.4) Contrary to single browser techniques cross-browser fingerprinting has no problem identifying users across browsers. Anyway. It has the same weakness as all fingerprints: identically configured devices. (Upathilake, Li, and Matrawy [16], pp.2) Weaknesses: (Upathilake, Li, and Matrawy [16], pp.2)

Chapter 3

Analysis based on an application scenario

3.1 Introduction

The following graph outlines techniques outlines in this thesis, excluding accelerometer fingerprinting which is not covered in this thesis.

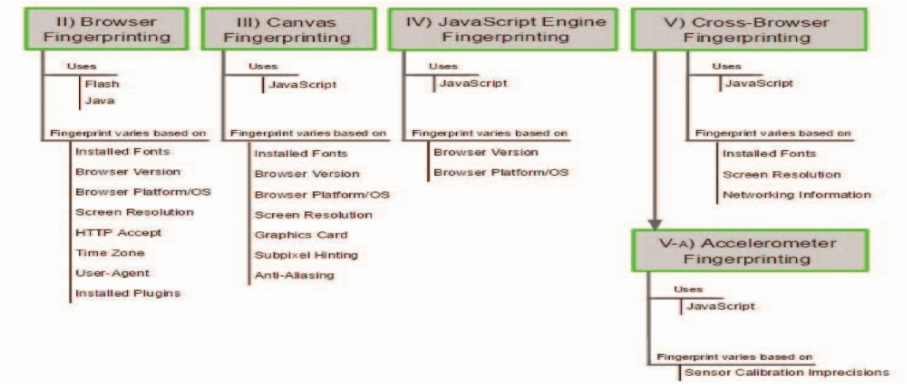

Fig. 1. Fingerprinting techniques and the attributes that affect the fingerprint.

Figure 3.1: Fingerprinting techniques and the attributes that effect the fingerprint

– refere to previous chapter – give good overview what needs what?

3.2 Browser specific fingerprinting

3.3 Canvas fingerprinting

3.4 JavaScript Engine fingerprinting

3.5 Cross-Browser fingerprinting

3.6 Accelerometer fingerprinting

3.7 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. (Upathilake, Li, and Matrawy [16], p.5)

Chapter 4

Requirements

4.1 Use cases

4.2 Functional requirements

- measure data
- create hash
- re-identify users

4.3 Non functional requirements

e.g. not -> to adjust the fingerprint to re-identify users even if some data changes

Chapter 5

Design and Implementation

5.1 Idea

5.2 Architecture

5.3 Programming

5.4 Obtained Data

5.5 Fingerprint

5.5.1 Calculation

5.6 Visualisation

Chapter 6

Evaluation

6.1 Prototype testcases

6.2 Comparison to AmIUnique and Panopticlick?

*AmIUnique – 90,84% of identifying across browsers * 17 features included to observe [6]

Chapter 7

Summary

References

Literature

[1] Gunes Acar et al. "The web never forgets: Persistent tracking mechanisms in the wild". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2014. URL: https://www.ftc.gov/system/files/documents/public_comments/2015/10/00064-98109.pdf (cit. on pp. 8, 16).

[2] Web analytics. *Browser fingerprints: the basics and protection options*. Tech. rep. 2017. URL: https://www.1and1.ca/digitalguide/online-marketing/web-analytics/browser-fingerprints-tracking-without-cookies/ (visited on 12/08/2018) (cit. on pp. 8–10, 14).

[3] K. Boda et al. "User tracking on the web via cross-browser fingerprinting". In: *Nordic Conference on Secure IT Systems*. Springer. 2011. URL: https://pet-portal.eu/files/articles/2011/fingerprinting/cross-browser_fingerprinting.pdf (cit. on pp. 8, 9, 18, 19).

[4] Yinzhi Cao, Song Li, and Erik Wijmans. "(Cross-)Browser Fingerprinting via OS and Hardware Level Features". In: *Proceedings of Network & Distributed System Security Symposium (NDSS)*. 2017. URL: http://yinzhicao.org/TrackingFree/crossbrowsertracking_NDSS17.pdf (cit. on pp. 18, 19).

[5] N. Doty. *Mitigating Browser Fingerprinting in Web Specifications*. Tech. rep. May 2018. URL: https://w3c.github.io/fingerprinting-guidance/ (cit. on pp. 3, 7, 9, 14).

[6] Peter Eckersley. "How unique is your web browser?" In: *International Symposium on Privacy Enhancing Technologies Symposium*. Springer. 2010 (cit. on pp. 6, 7, 9, 15, 17).

[7] R. Fielding and J. Reschke. *Hypertext transfer protocol (HTTP/1.1): Semantics and content*. Tech. rep. 2014. URL: https://tools.ietf.org/html/rfc7231#section-5.5.3 (cit. on pp. 12, 13).

[8] R. Havens. "Browser Fingerprinting: Attacks and Applications". 2016. URL: http://www.cs.tufts.edu/comp/116/archive/fall2016/rhavens.pdf (cit. on pp. 3, 4, 7, 11, 13).

[9] C. Hoffman. *What is a Browser's User Agent?* Tech. rep. 2016. URL: https://www.howtogeek.com/114937/htg-explains-whats-a-browser-user-agent/ (cit. on p. 12).

[10] J. R. Mayer. "Any person... a pamphleteer": Internet Anonymity in the Age of Web 2.0". *Undergraduate Senior Thesis, Princeton University* (2009) (cit. on p. 1).

[11] K. Mowery and H. Shacham. "Pixel perfect: Fingerprinting canvas in HTML5". *Proceedings of W2SP* (2012). URL: https://hovav.net/ucsd/dist/canvas.pdf (cit. on pp. 8, 9, 11, 16, 17).

[12] M. Mulazzani et al. "Fast and reliable browser identification with javascript engine fingerprinting". In: *Web 2.0 Workshop on Security and Privacy (W2SP)*. Citeseer. 2013. URL: http://www.ieee-security.org/TC/W2SP/2013/papers/s2p1.pdf (cit. on pp. 9, 10, 17, 18).

[13] et al. N. Nikiforakis. *Cookieless monster: Exploring the ecosystem of web-based device fingerprinting*. Figure 3 in Nikiforakis et al. [22]. 2013 (cit. on p. 6).

[14] P. Reschl et al. "Efficient browser identification with JavaScript engine fingerprinting". In: *Proc. of Annual Computer Security Applications Conference (ACSAC)*. 2011. URL: https://publik.tuwien.ac.at/files/PubDat_202737.pdf (cit. on p. 17).

[15] K. Szymielewicz and B. Budington. *The GDPR and Browser Fingerprinting: How It Changes the Game for the Sneakiest Web Trackers*. Tech. rep. EFF, 2018. URL: https://www.eff.org/de/deeplinks/2018/06/gdpr-and-browser-fingerprinting-how-it-changes-game-sneakiest-web-trackers (cit. on pp. 4, 7, 8).

[16] Randika Upathilake, Yingkun Li, and Ashraf Matrawy. "A classification of web browser fingerprinting techniques". In: *New Technologies, Mobility and Security (NTMS), 2015 7th International Conference on*. IEEE. 2015. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7266460 (cit. on pp. 3, 4, 6, 8, 15–19, 21).

[17] T. Yen et al. "Browser fingerprinting from coarse traffic summaries: Techniques and implications". In: *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer. 2009. URL: https://www.cs.unc.edu/~reiter/papers/2009/DIMVA.pdf (cit. on p. 9).

Online sources

[18] AmIUnique. URL: https://amiunique.org/faq (visited on 10/22/2018) (cit. on pp. 3, 4, 9, 11).

[19] Michael Aranda. *What's the difference between JavaScript and ECMAScript?* URL: https://medium.freecodecamp.org/whats-the-difference-between-javascript-and-ecmascript-cba48c73a2b5 (visited on 12/13/2018) (cit. on p. 10).

[20] P. Arntz. *Explained: user agent*. 2017. URL: https://blog.malwarebytes.com/security-world/technology/2017/08/explained-user-agent/ (cit. on p. 12).

[21] javascript.info. *An introductin ot JavaScript*. URL: https://javascript.info/intro (visited on 12/13/2018) (cit. on pp. 9, 10).

[22] N. Nikiforakis et al. *Cookieless monster: Exploring the ecosystem of web-based device fingerprinting*. IEEE. 2013. URL: https://seclab.cs.ucsb.edu/media/uploads/papers/sp2013_cookieless.pdf (visited on 10/27/2018) (cit. on pp. 4, 6, 8–11, 27).

[23] pixelprivacy. *Browser Fingerprinting*. URL: https://pixelprivacy.com/resources/browser-fingerprinting/ (visited on 10/15/2018) (cit. on pp. 10, 11).

[24] Patrick Sexton. *Base64 encoding images*. URL: https://varvy.com/pagespeed/base64-images.html (visited on 12/13/2018) (cit. on p. 12).

[25] w3techs. *Usage of JavaScript for websites*. URL: https://w3techs.com/technologies/details/cp-javascript/all/all (visited on 12/10/2018) (cit. on p. 9).

[26] Adam Wood. *JavaScript*. URL: https://html.com/javascript/ (visited on 12/13/2018) (cit. on p. 9).

[27] Xovi. *User Agent*. URL: https://www.xovi.de/wiki/User_Agent (visited on 10/15/2018) (cit. on p. 12).

List of Figures

2.1 Top 10 categories utilizing fingerprinting (N. Nikiforakis [13]) 6
2.2 Example for the structure of a minimal fingerprint 17
2.3 Example for the structure of a decision tree 18

3.1 Fingerprinting techniques and the attributes that effect the fingerprint . . 20