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

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

(**Mayer09**, 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. 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 plugins. 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 [3], p.3)(AmIUnique [16]) 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 [15], p. 1)(Havens [6], 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 [6], 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 [14])

Web browser fingerprinting was first developed roughly before 2010 and due to its effectiveness quickly gained traction in the tracking industry.(Havens [6], 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 [16])(Szymielewicz and Budington [14]), makes it a dangerous tool for deanonymization attacks and maliciously-minded tracking programs. (Havens [6], 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 [6], 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. [18], 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 [15], p.4)(Nikiforakis et al. [18], 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 [15], 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. [18], pp.6)

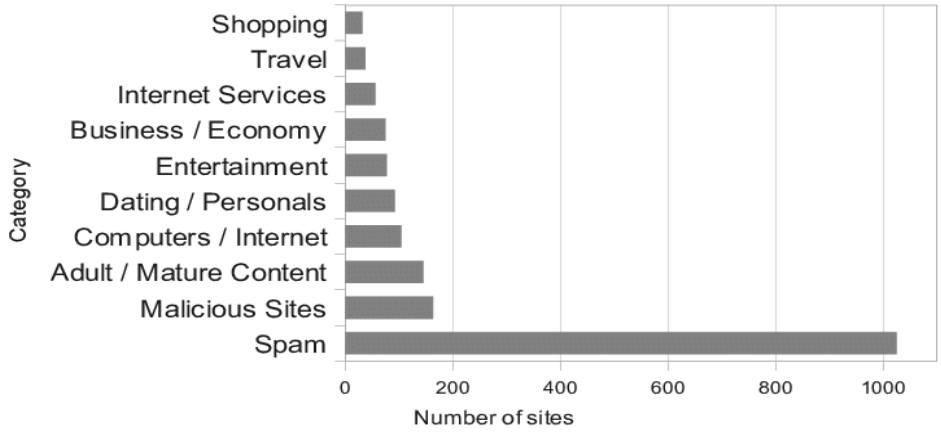

Figure 2.1: Top 10 categories utilizing fingerprinting (N. Nikiforakis [12])

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. [18], p.7)

2.3 Threat

have a look at (Eckersley [4])

As seen in the previous sub-chapter /refsec:utilisation web browser fingerprinting does have its advantages as well as its disadvantages for the user. This section will have a closer look as the threat fingerprinting poses for the users privacy.

Web browser fingerprinting is a tracking method, which can correlate user's browsing activity within as well as across sessions without transparency or control. (Doty [3], p.3) Most fingerprinting techniques are obscure and users usually don't notice that they are being tracked.

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. (Havens [6], p. 4)

Privacy and security

- 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. (Doty [3], p.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. (Doty [3], p.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*/ (Doty [3], p.4f)

go to original source 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** Group -> privacy-conscious users, informed of the dangers of browser fingerprinting ** Simple algorithm for detecting changes in fingerprints. o Check to see if there is any fingerprint which, aside from one non-matching field, is otherwise the same. o Not make guess if there were multiple potential matches o Made 65,56% guesses – 99,1(Havens [6], p.6f)

** 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. (Havens [6], p.9f)

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 [14])

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 [14])

In order to be allowed to use fingerprinting legally the concerned entity has to com-

plete 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 [14])

2.4 Mitigation - max. 1,5 -2 pages

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. So this brings up the question if there is a way to avoid this tracking method. Each of the researched articles about this topic stated the same. No. Unfortunately, there is no way to avoid being fingerprinted completely. There are only ways to try and reduce the risk of being tracked. (analytics [1]) The danger of web browser fingerprinting lays in its stateless nature. * hard to detect - if even possible * impossible to opt-out (Upathilake, Li, and Matrawy [15], p.4)

2.4.1 Best practice

According to N. Doty (Doty [3], p.5), these are the best practices to mitigate the fingerprint we leave online:

- 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

Further Doty mentions that this helps:

- Decreasing fingerprinting surface
- Increasing anonymity set
- detectable fingerprinting
- clearable local state

(Doty [3], p.7)

2.4.2 Prevention?

While N. Doty only listed ways to decrease the chance of being fingerprinted, R. Upathilake suggests the following steps to actually prevent the chance of being tracked via web browser fingerprinting (Upathilake, Li, and Matrawy [15], p.4):

- Having all browser vendors agree on a single set of API calls to expose to the web applica-tions 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 render-ing;
- 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

2.4.3 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. (Upathilake, Li, and Matrawy [15], p.4)

– using private browsing modes or tor anonymity service – tor -> slower, less attrackive browsing experience – torbutton disables WebGL - still allows rendering to a <canvas> -> still partly vulnerable to canvas FP – mainstream users -> fingerprinting unavoidable consequence (Mowery and Shacham [10], p.1)

– increasing the noise, but degrades the performance of <canvas> significantly for legitimate applications – same noise on multiple runs -> aid fingerprinting – therefore adding noise - not a feasible defence – one way would be: browser vendors - agree on a list of "canvas-safe" fonts – to support WebGL, ignore graphics card and render scenes in a generic software renderer – approach might be acceptable - performance impact not – easiest efficient defense: require user approval whenever a script requests pixel data – modern browsers already implement this type for e.g. HTML5 geolocation APIs (Mowery and Shacham [10], p.10)

2.4.4 Extensions and plug-ins

The use of some common extensions and plug-ins were also mentioned in the liter-ature. 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. (Upathilake, Li, and Matrawy [15], p.4)

2.4.5 The simplest solution

– browser extension -> automatically blocks active content e.g. JavaScript, Flash, or Silverlight applications (which don't deliver logically info to server) – these plugins - incl. NoScript (Firefox), ScriptBlock (Chrome) -> protection against canvas fingerprinting – using these plug-ins -> some web services or at least individual content -> stop working – not helpful if not sure if provider is trustworthy or not – these blockers can be directly used for fingerprinting the user – apart form script blocking solution – commonly-used browser and default settings – same with OS – high chance - no unique fingerprint and will be harder to track (analytics [1])

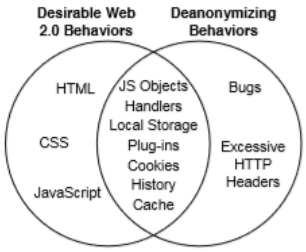

Figure 2.6: Desired Web 2.0 behaviors and deanonymizing behaviors; there is significant overlap, consistently arbitrated in favor of Web 2.0.

Figure 2.2

2.5 Critical configurations and plug-ins

– max. 1/2 page per config

https://www.1and1.ca/digitalguide/online-marketing/web-analytics/browser-fingerprints-tracking-without-cookies/

(Mayer [9], p.43)

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. (Doty [3], 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.

The information that browser fingerprinting reveals typically includes a mixture of HTTP headers (which are delivered as a normal part of every web request) and properties that can be learned about the browser using JavaScript code: your time zone, system fonts, screen resolution, which plugins you have installed, and what platform your browser is running on. Sites can even use techniques such as canvas or WebGL fingerprinting to gain insight into your hardware configuration.(Szymielewicz and Budington [14])

2.5.1 JavaScript

JavaScript is enabled by default in all major browsers and by November 2018 JavaScript was included by 95% of all websites online. (w3techs [20]). – therefore very useful tool - also because can query! Js – the language which never works – for the developer

– what is sputnik and test 262 ()

JS gives access to many browser-populated features like the plugins installed on the user's device (AmIUnique [16])

Identifying web browsers based on the underlying javascript engine (Mulazzani et al. [11])

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, be-cause it's also used to make websites run smoothly on your device. This will impact your browsing experience.(pixelprivacy [19])

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. (Havens [6], p.8)

Javascript has been standardized as ECMAScript [8], and all major browsers implement it in order to allow client-side scripting and dynamic websites. Traditionally, Web developers use the UserAgent string or the navigator object (i.e., navigator.UserAgent) to identify the client's Web browser, and load corresponding features or CSS files. The UserAgent string is defined in RFC2616 [11] as a sequence of product tokens and identifies the software as well as significant subparts. Tokens are listed in order of their significance by convention. The navigator object contains the same string as the UserAgent string. However, both are by no means security features, and can be set arbitrarily by the user. (Mulazzani et al. [11], p.2)

SEE NIKIFORAKIS!!!!

WebGL

2.5.2 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. (Nikiforakis et al. [18], p.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 (Nikiforakis et al. [18], p.3) ** Returns all fonts with one simple call (Havens [6], p.5) ** Its rich programming interface (API) provides access to many system-specific attributes (version of OS, list of fonts, screen resolution, timezone) (AmIUnique [16]) ** Can be disabled without a negative impact on UEX – only impacts browsing experience on old websites(pixelprivacy [19].) ** Vulnerability, called "local storage objects" -> enables "zombie cookie" or "supercookie". Almost impossible to remove. Populate numerous different storage areas with tracking in-formation. 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" (Havens [6], p.8f)

SEE NIKIFORAKIS!!!!

2.5.3 HTML 5 /Canvas

– HTML5 is the coding language used to build websites -> core fundamentals of every website – has the element "canvas" – originally html <canvas> element was used to draw graphics on a web page (pixelprivacy [19])

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 (AmIUnique [16])

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 [21]) It is automatically sent with each page call (Doty [3], p.6), except if it is specifically configured not to. (Xovi [21])(Fielding and Reschke [5], 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 [21])(Fielding and Reschke [5], p.5.5.3) This field value is often called user agent string. (Hoffman [7])

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 [7]) (Xovi [21]) The passed-on information is used to display different content in different browsers, on different oper-ating systems, or to gather statics (Hoffman [7]) (Arntz [17]) (Xovi [21])

If wanted, changes on the user agent string are easily made. (Arntz [17]) 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 [5], p.5.5.3)

2.5.5 Others

Quickly describe which else there are: (Havens [6], 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 [6], p.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

//probably why it's needed, which can be blocked

2.6 Web browser fingerprinting methods

2.6.1 Active fingerprinting

The active fingerprinting method needs to actively query information about the client that isn't automatically provided by the browser. (analytics [1]) To obtain additional characteristics the site needs to run a JavaScript code or use plug-ins like Adobe Flash which extend the browsers functionality. (analytics [1])((Doty [3], p.6)

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

- user's screen (width, height, resolution)
- 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. (Doty [3], p.6) Active fingerprinting can be discovered in case the user analyses the outgoing data packages, HTML or the JavaScript source code, which in the majority of cases does not happen. (analytics [1])

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 [3], p.6)(analytics [1]) 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)
- 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 via a from collected data generated fingerprint. (Upathilake, Li, and Matrawy [15], p.2) This study has shown that 94.2% of the uses which use flash or java could be distinguished with browser fingerprinting. (Eckersley [4], p.2)
Possible even if the browser only reveals as much as its version and configuration information which is in most cases enough to create a distinct fingerprint.(Eckersley [4], p.16)

How it works

Browser specific fingerprinting is a technique which only uses browser-dependent features (like flash and JavaScript or be retrieved from a http request) to collect a number of common or and less-common known browser characteristics.
This retrieved browser information typically includes the installed fonts, plug-ins (including version numbers), user agent, http accept and screen resolution. (Upathilake, Li, and Matrawy [15], 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 [4], p.4) (Upathilake, Li, and Matrawy [15], p.2)

The interesting fact about browsers is that sometimes, when they try to hide something, this helps even more to make the fingerprint distinct.

> Example:
> When a browser uses a flash blocking add-on, the browser still displays flash in the plugin 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 was not explicitly detected. (Eckersley [4], p.4)

Weaknesses: (Upathilake, Li, and Matrawy [15], p.2)

- unable to distinguish between instances of identically configured devices
- fingerprint can change across browsers
- fingerprint can be easily changed (upgrade, plug-in, etc.)

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".
As the title teasers, this web browser fingerprinting technique works by using the new coding features in HTML5.(pixelprivacy [19]) 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 [10], p.1)(Upathilake, Li, and Matrawy [15], p.2)

For rendering text fonts, webfonts were used as they are more distringuishable than expected. (Mowery and Shacham [10], p.6) Further WebGL was used as graphic cards leave a detectable fingerprint while rendering even the simplest scenes.(Mowery and Shacham [10], p.8)

> Example text rendering:
> Out of 300 samples the text font Arial was rendered in 50 distinct ways. (Mowery and Shacham [10], p.6)

The outcome of their research and tests was that the canvas fingerprint is consistent but would change depending on hardware and software configuration. Further Mowery and Shacham discovered that the canvas fingerprint is unable to distinguish between users who use the exact same hadware and software.(Mowery and Shacham [10], p.5)

Only 2 years after its first use a study conducted by Acar et al.(2014) ascertained that canvas fingerprinting is the most common form of fingerprinting with an approximately occurrence of 5.5% in the top 100,000 websites. (Upathilake, Li, and Matrawy [15], p.2)

Weaknesses

(Upathilake, Li, and Matrawy [15], p.2)

- unable to distinguish between instances of identically configured devices
- fingerprint can change across browsers

How it works

Canvas fingerprinting renders text and WebGL scenes onto an area of the screen using the HTML5<canvas> element programmatically and reads the pixel data back 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 [15], 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' contents. This Base 64 encoded pixel data url is used to create a hash.(Mowery and Shacham [10], p.3)(Upathilake, Li, and Matrawy [15], 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 [15], p.2)

In their thesis K. Mowery and H. Shacham used two types of image comparisons: pixel-level difference and difference maps.(Mowery and Shacham [10], pp.4)

Pixel-level difference works by setting each pixel's color to a channel-wise difference between tow images at that location. If the color is something else thatn transparent pure black it indicates that there is a difference between the two pixels. They set the alpha value of differing pixel to 255 to render it completly opaque. (Mowery and Shacham [10], 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 [10], pp.4)

2.7.3 JavaScript Engine fingerprinting

In 2011 P. Reischl et al. published their first paper about the use of the JavaScript Engine to uniquely identify a web browser and major version. Comparing it to the Panopticlick project (Eckersley [4]) it was stated in the paper that the new approach is multiple times faster and error prone. (Reschl et al. [8], 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. [11], p.1)

JavaScript Engine fingerprinting uses testsuits like Sputnik to determine the used browser and version. 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. [11], p.1)

How it works

JavaScript Engine fingerprinting works by running test cases of conformance tests like Sputnik and test 262 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. [11], p.3)

There are two methods which can be used to identify the webbrowser and the major version.

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. They select a test case which only one browser failed. This browser can be uniquely identified by this test and is therefore removed from the set. This process is redone for each browser until there is a unique fingerprint for each browser in the set or there is no test case which only one browser failed.

If there is no unique fingerprint for a browser the selection is simply changed.(Mulazzani et al. [11], p.3)

The figure below 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 2.3: 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. [11], p.4)

The figure below **reference** give 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. [11], p.4)

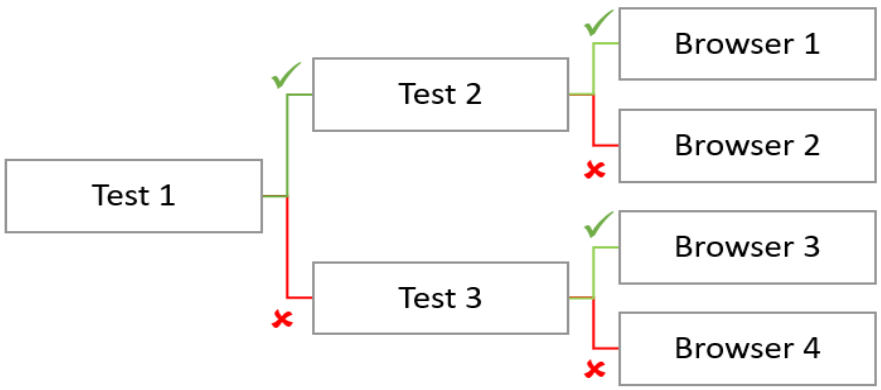

Figure 2.4: Example for the structure of a decision tree

2.7.4 Cross-Browser fingerprinting

different kinds of cross-browser finger printing - first by boda - using ip - not so good in 2017 cao proposes- better option

– uses browser-independent features for the fingerprint – can rely solely on js and use font detection as a techniue – since a js engine is available and enabled by default on

all modern browsers – differs form techniques which rely on browser dependent plug-ins Adobe Flash or Java for obtaining a font list (Upathilake, Li, and Matrawy [15], pp.2)

Weaknesses: (Upathilake, Li, and Matrawy [15], pp.2)

- unable to distinguish between instances of identically configured devices
- fingerprint can change????

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 ** (Nordrum [13])

** Rely on corresponding OS and hardware functionalities ** 99,24% successfully identify (**Cao17**)

– first by boda – then extended by cao

(Boda et al. [2]) – a part of the ip address, availability of a specific font set, time zone, screen resolution – uniquely identify most users of the 5 most popular web browsers – user agent strings - effective but fragile identifiers of browser instance (Boda et al. [2], p.1)

– browser-independent features as a basis of identification – eg. networking information (ip address, hostname, tcp port number) – eg. application layer information (uas in http header, name/version of os, extemsions, etc.) – eg. info gained by querying - over js - list of fonts, plugins(+version), screen resolution, time zone (Boda et al. [2], p.2)

–create a unique identifier from specific browser data using JavaScript and server-side algorithms (Boda et al. [2], p.4)

– crossdomain tracking is achieved by using only the first two octets of the IP address, which remains constant in many cases even if the IP address of the client changes dynamically – may change after switching to a different ISP or another service – use 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. [2], p.5)

– font lists provide a solid base for unique identification under the Windows and Mac OSes (Boda et al. [2], p.8)

The latter is true largely due to the fact that the list of fonts is easily accessible through the JavaScript API. Therefore, if resistance of identification is to be secured with the web browsers of today, the only option seems to be to disable such scripts. That way, one's anonymity set can be greatly increased. Users of computers in a set of identically configured machines are likely to have the best odds against our fingerprinting algorithm. If such computers are centrally managed, it is unlikely that any of them has a unique feature based on which it could be identified. Therefore, passive methods are insufficient in such a context. Our research has some important implications regarding the user's behaviour. We have provided recommendations about browser types and versions that are likely indistinguishable from other such browsers. Furthermore, it can be seen that resistance to identification is easier achieved with popular system settings. It must be noted, however, that these measures are not necessarily easy to carry out without causing inconvenience to the user. We have also given some guidelines for developers to make their browsers resist our fingerprinting algorithm. In particular, the verbosity of the UAS and the plugin versions should be decreased, and a standard 'substitute font set' – preferably

supported by all browsers – should be introduced. Furthermore, users should be able to set fake system properties in the browser in order to hide their real operating system, time zone and screen resolution. Finally, we have proposed some improvements for our method, and defined certain directions for future experiments; an explicitly cross-browser experiment would provide a high-precision dataset. (Boda et al. [2], p.16) unavailable behind an anonymous network or a proxy (**Cao17**, p.1)

– based on many novel OS and hardware level features, e.g., these from graphics card, CPU, audio stack, and installed writing scripts. Specifically, because many of such OS and hardware level functions are exposed to JavaScript via browser APIs, we can extract features when asking the browser to perform certain tasks through these APIs. The extracted features can be used for both single- and cross-browser fingerprinting. (**Cao17**, p.1)

– use many novel OS and hardware features, especially computer graphics ones – successfully fingerprint 99.24% of users – 83.24% uniqueness with 91.44% cross-browser stability – Boda excluding IP address only have 68.98% uniqueness with 84.64% cross-browser stability. – current measurement of screen resolution, e.g.AmIUnique, Panopticlick, Boda - unstable – resolution changes in Firefox and IE when zooms – zoom level into consideration, normalize the width and height in screen resolution – DataURL and JPEG formats are unstable across different browsers, because these formats are with loss and implemented differently in multiple browsers and the server side as well. – need to adopt lossless formats for server-client communications in cross-browser fingerprinting. – use PNG, a lossless format, for cross-browser fp – Prior Fingerprintable Features – 4 features for cross-browser fp (screen res., color depth, list of fonts, platform) – nr of CPU virtual cores – screen resolution (in js simply "screen") - firefox/ie - change resolution value according to zoom level – use zoom levels, adjust resolution, screen ration also important (**Cao17**, p.2)

– nr of CPU virtual cores - obtained by browser feature "hardwareConcurrency" – audiocontext - provides bundle of audio signal processing functionalities from signal generation to signal –peak values and their corresponding frequencies are relatively stable across browsers. – create a list of bins with small steps on both the frequency and value axes, and map the peak frequencies and values to the corresponding bins. – If one bin contains a frequency or value, we mark the bin as one and otherwise zero: such list of bins serve as our cross-browser feature. – List of Fonts width and height of a certain string is measured to determine font type – Line, curve, and anti-aliasing. Line and curve are 2D features supported by both Canvas (2D part) and WebGL. –Vertex shader. A vertex shader, rendered by GPU and the driver, converts each vertex in a 3D model to its coordinate in a 2D clip-space. – Fragment shader. A fragment shader, rendered by GPU and the driver as well, processes a fragment, such as a triangle outputted by the rasterization, into a set of colors and a single depth value. — MANY MORE - not write down all, als need to cut short - max. 1page per technique!!!! (**Cao17**, p.3)

HOW it works: – task manager at server side - send various randering tasks – e.g. drawing curves, lines to client side – rendering task - also obtaining OS and hw level info – client-side browser renders and produces results (images, sound waves) – results convert into hashes -> to send to server – meantime - browser collect browser-specific info (for fp composition) – fp is generated from list of hashes from client side and a mask – that is a list of 1 or 0 corresponding to the hash list – perform "and" operation between list of

hashes and mask -> create another hash as fp – mask: if browser not support anti-aliasing - all bit in mask involve anti-alisaing -> 0 – second part of mask -> each browser pair - different mask (**Cao17**, p.4)

– add rendering task? p-5

FP composition – how to make fp on server side from client-side hashes – computed with -and- operation – generation of mask for every two browsers is training-based approach (**Cao17**, p.7)

– go through results?

– tor browser -> many features unavailable – disables canvas by default -> not fingerprintables if disabled? (**Cao17**, p.13)

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.

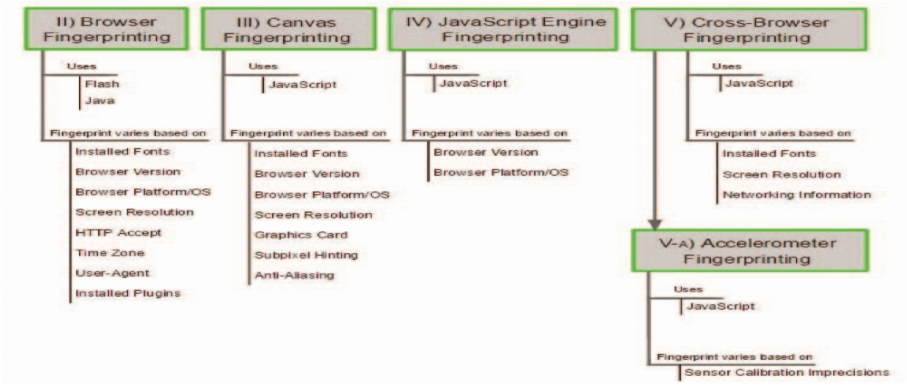

Fig. 1. Fingerprinting techniques and the attributes that affect the fingerprint.

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 [15], 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] 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, 15).

[2] 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. 20, 21).

[3] N. Doty. *Mitigating Browser Fingerprinting in Web Specifications*. Tech. rep. May 2018. URL: https://w3c.github.io/fingerprinting-guidance/ (cit. on pp. 3, 6–9, 11, 13, 15).

[4] Peter Eckersley. "How unique is your web browser?" In: *International Symposium on Privacy Enhancing Technologies Symposium*. Springer. 2010 (cit. on pp. 6, 16, 18).

[5] 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 p. 13).

[6] 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, 12, 14).

[7] 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. 13).

[9] 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. 11).

[10] 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. 10, 17, 18).

[11] 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. 11, 12, 18, 19).

[12] et al. N. Nikiforakis. *Cookieless monster: Exploring the ecosystem of web-based device fingerprinting*. Figure 3 in Nikiforakis et al. [18]. 2013 (cit. on p. 6).

[13] A. Nordrum. *Browser Fingerprinting Tech Works Across Different Browsers for the First Time*. Tech. rep. 2017. URL: https://spectrum.ieee.org/tech-talk/telecom/internet/new-online-fingerprinting-technique-works-across-browsers (cit. on p. 20).

[8] P. Reschl et al. "Efficient browser identification with JavaScript engine fingerprinting". In: *Proc. of Annual Computer Security Applications Conference (ACSAC)*. 2011. URL: www.researchgate.net/profile/Edgar_Weippl/publication/230035801_Poster_ACSAC_2011_Efficient_Browser_Identification_with_JavaScript_Engine_Fingerprinting/links/5862902e08ae8fce49098567/Poster-ACSAC-2011-Efficient-Browser-Identification-with-JavaScript-Engine-Fingerprinting.pdf (cit. on p. 18).

[14] 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. 3, 4, 7, 8, 11).

[15] R. Upathilake, J. Li, and A. Matrawy. "A Classification of Web Browser Fingerprinting Techniques". IEEE, 2015. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7266460 (cit. on pp. 3, 4, 8–10, 16–18, 20, 24).

Online sources

[16] AmIUnique. URL: https://amiunique.org/faq (visited on 10/22/2018) (cit. on pp. 3, 4, 11, 12).

[17] P. Arntz. *Explained: user agent*. 2017. URL: https://blog.malwarebytes.com/security-world/technology/2017/08/explained-user-agent/ (cit. on p. 13).

[18] 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, 12, 30).

[19] pixelprivacy. *Browser Fingerprinting*. URL: https://pixelprivacy.com/resources/browser-fingerprinting/ (visited on 10/15/2018) (cit. on pp. 11, 12, 17).

[20] w3techs. *Usage of JavaScript for websites*. URL: https://w3techs.com/technologies/details/cp-javascript/all/all (visited on 12/10/2018) (cit. on p. 11).

[21] Xovi. *User Agent*. URL: https://www.xovi.de/wiki/User_Agent (visited on 10/15/2018) (cit. on p. 13).