Invisible Identifiers - How Browser Fingerprinting Challenges Internet Privacy and User Anonymity

Alexander Lawall
IU International University of Applied Science
Erfurt, Thüringen, Germany
alexander.lawall@iu.org

***Abstract*—Browser fingerprinting has emerged as a sophisticated and increasingly prevalent technique for identifying and tracking users online without relying on traditional methods like cookies. This paper provides a comprehensive overview of browser fingerprinting techniques, ranging from passive and active methods like Hypertext Transfer Protocol (HTTP) header analysis to advanced machine learning-assisted side-channel attacks. By evaluating the uniqueness, stability, and entropy of different methods, the study highlights how the synergistic combination of multiple techniques enhances the accuracy and persistence of user identification. The analysis demonstrates that browser fingerprinting poses a significant challenge to digital privacy by operating invisibly, often without user knowledge or consent. Despite regulatory frameworks such as the General Data Protection Regulation, the widespread use of fingerprinting scripts remains largely unchecked, exploiting legal loopholes and technological asymmetries. The paper also explores the potential of privacy-preserving fingerprinting systems for secure user authentication while emphasizing the urgent need for adaptive countermeasures, regulatory reforms, and increased user awareness to protect individual privacy in the evolving digital landscape.**

Keywords-browser fingerprinting; device fingerprinting; tracking; privacy; active fingerprinting; passive fingerprinting.

I. Introduction

This work is an extended version of *Fingerprinting and Tracing Shadows: The Development and Impact of Browser Fingerprinting on Digital Privacy*, published at SECURWARE 2024 [1], [2]. In the increasingly digitized world, the issues of online privacy and data security are becoming more complex. Particularly in tracking — monitoring users and their devices across different web servers — browser fingerprinting has emerged as an effective technique for creating detailed user profiles. Unlike the storage of information via cookies, which requires explicit user consent as mandated by the European General Data Protection Regulations (GDPR) guidelines, fingerprinting does not require such consent. A browser fingerprint can be generated in the background without any obvious signs to the end user, leaving them unaware of whether and to what extent they are being tracked.

It is possible to manipulate a device locally to alter its fingerprint. This is often not feasible for all users, unlike deleting cookies. This invisible threat is not apparent to the general public and raises significant privacy concerns, as individuals can be tracked unnoticed. These profiles can contain private information, depending on the server operators, including age group, ethnic origin, social circles, and interests of the affected person.

Browser fingerprinting poses a threat to the privacy of the general public. Contrary to being a threat, it is an opportunity to provide valuable information to enhance the authentication mechanisms. Both perspectives are explored throughout this paper. The focus will be on the various techniques of fingerprinting to understand how accurate and detailed user profiles can be created. The main research questions that this paper seeks to answer are:

RQ1 *"What methods are used in browser fingerprinting and what user data are collected in the process?"*

RQ2 *"How has the development of browser fingerprinting as a user identification method influenced user privacy and data protection in the digital space?"*

The paper is structured as follows: Section I introduces browser fingerprinting and its privacy implications. In Section II, the theoretical background explains how fingerprinting works and its legal challenges. Section III outlines techniques like HTTP Headers, Canvas, and WebGL Fingerprinting. Section IV examines the impact of fingerprinting on privacy and the regulatory landscape. Section V concludes with a summary of the findings, emphasizing the need for stronger privacy measures and further research on countermeasures.

II. Theoretical Background

This section lays the conceptual foundation for understanding browser fingerprinting, detailing its underlying mechanisms, legal ambiguities, and role in modern tracking practices. It introduces both passive and active techniques used to collect identifying data from users' browsers without explicit consent, highlighting the technical simplicity yet high effectiveness of these methods. Furthermore, it explores the growing tension between evolving tracking technologies and regulatory protections like the GDPR, illustrating how fingerprinting often operates in legal gray zones that undermine user privacy and control.

A. Fingerprinting

Browser fingerprinting refers to collecting characteristic information that the browser directly or indirectly reveals about itself. Often used to track users, this technology has also found applications in IT security, such as fraud detection. Unlike tracking methods like cookies, browser fingerprinting does not require storing data on the user's computer, allowing the process to occur secretly and without consent [3, p. 1]. Consequently, creating a new identity, similar to deleting

cookies, is not easily achievable, and GDPR privacy laws often provide little protection. Unlike cookie tracking, browser fingerprinting is not explicitly mentioned in the GDPR. It should fall under the collection of identifiable information but website operators frequently claim "legitimate interest", enabling such data collection without the user's consent [4].

Active transmission of data is not required for browser fingerprinting, as loading a webpage can transmit various pieces of information, such as the user's preferred language, within the HTTP headers. This passive data collection provides only a limited amount of information, so it is often supplemented with active data collection methods. An active approach typically employs JavaScript to interface with the browser and gather information, such as screen resolution, installed add-ons, and graphics card data, merging them into a unique fingerprint [5, pp. 1, 3].

Similar to human fingerprints, browser fingerprinting relies on the uniqueness of browser characteristics, which typically do not change significantly with regular use. This allows for accurate user identification over extended periods [5, p. 2]. However, not all collected data points are equally unique or stable, necessitating careful selection of information to achieve accurate results. The fingerprinting algorithm combines both passively and actively collected data into a unique string. Depending on the operator's goals, adjustments can be made; for instance, using cookies, the fingerprint might be less stable but more unique, while tracking users without cookies requires high stability [6, pp. 1-5]. Eckersley's study showed that participant browsers already had high entropy, indicating many unique characteristics sufficient for accurate fingerprinting, though not stable enough for long-term accuracy. In recent years, potential entropy has increased with new techniques like HTML Canvas, WebGL-based hardware fingerprints, audio API fingerprints, plug-in-based fingerprints, and methods utilizing mouse movements or differences in HTML parsing between browsers, making cross-browser user identification possible [5, pp. 4-5].

B. Concerns for Digital Privacy

Historically, the greatest threat to online tracking was posed by cookies, along with other technologies like Flash cookies, which have lost significance in recent years. Changes by browser manufacturers, such as Mozilla, which rendered many exploited technologies, so-called "super-cookies", ineffective [7], and additional browsers planning to block or eliminate third-party cookies in the coming years [8], have shifted the landscape. Following the GDPR, the use of non-essential cookies has been further restricted and standardized for the first time, defining how users share their data through cookies [9]. In contrast, browser fingerprinting occurs in the background and leaves no stored information on the user's computer. Thus, the use of fingerprints not only circumvents previous issues related to local storage, such as privacy laws and technical limitations but also persists even when local data is deleted or when incognito mode is used.

A 2021 study of the Alexa Top 100,000 websites found that nearly 10% of the sites used scripts to generate fingerprints [10, pp. 11-12]. Comparing this to a similar 2014 study, which recorded 5.5% of the top 100,000 sites using canvas fingerprinting scripts, reveals an almost doubling of usage over seven years [11]. This suggests a shift towards online tracking using this technology, which is much harder to detect and prevent compared to cookies. The creation of a fingerprint is imperceptible to the user, with no simple way to effectively change or delete their fingerprint. Cookie banners give a false sense of security while tracking continues in the background without consent.

Thus, browser fingerprinting poses an active threat to privacy, as users often have no control over the collection and use of their data. This stands in opposition to many current data protection principles, such as the GDPR.

III. Methods of Browser Fingerprinting

In the context of browser fingerprinting techniques, the methods of data collection are varied and comprehensive. Therefore, specific properties and criteria are used to select techniques. The following sections will encompass the explanation of the techniques in terms of their functionality and their applications will be discussed to provide a detailed understanding of their use. An evaluation based on the advantages and disadvantages of each technique is also included to weigh their effectiveness and potential risks. Given the ever-increasing number of techniques, only the most commonly used, established, or novel methods will be presented here.

A. HTTP Header Attributes

1) Definition and Basics: The HTTP request header is a part of every HTTP request exchanged between a client (web browser) and a server, transmitting various functional and compatibility-related information [12]. While individual attributes are typically not unique, their combination can enhance the distinctiveness of a client within a larger population. This explanation is based on HTTP version 1.1, with HTTP/2 introducing fundamental structural changes. However, most attributes remain in use within the modified header frame [13].

2) Analysis: The attributes of HTTP request headers can vary depending on the browser and its version. For fingerprinting purposes, it is crucial to select fields that remain consistent over time and are not easily influenced by user behavior. For example, the Host header, which conveys the target server's domain, should be avoided as it is directly dependent on the request destination. In contrast, the User-Agent field typically exhibits high stability and provides extensive information, making it particularly suitable for fingerprinting [14].

Studies by AmIUnique [15, p. 880] and PanOptiClick [6, p. 5] identify the User-Agent, Accept, Content-Encoding, and Content-Language fields as reliable attributes. These studies collected user fingerprints voluntarily and demonstrated their effectiveness in user identification. The User-Agent field, although not standardized, frequently contains information about browser compatibility, version, and operating system, often

with varying levels of detail. Due to its lack of standardization and manufacturer-specific implementations, the User-Agent field exhibits high entropy, with modifications typically occurring only through browser updates [16].

The Accept, Content-Encoding, and Content-Language fields convey less information individually but can reveal insights into the operating system, browser type, and language preferences. Uncommon languages or specific language-region combinations may yield unique fingerprints [17]–[19]. Additional fields such as Referer, Connection, Content-Length, X-Forwarded-For, Cookie, and Cache-Control can complement fingerprinting but provide minimal uniqueness on their own. However, the presence of certain headers like X-Forwarded-For may indicate specific configurations or proxy usage [15, pp. 879-880].

The DoNotTrack (DNT) header, although originally intended to signal tracking preferences, has paradoxically become a fingerprinting target due to its voluntary nature and lack of enforcement [20, p. 313]. Furthermore, the sequence of header fields may serve as an additional fingerprinting feature, particularly when combined with manipulated User-Agent information. Cookies, while transmitted within HTTP headers, require client-side storage and are thus excluded from this discussion.

3) Advantages: The primary advantage of utilizing HTTP headers for fingerprinting is the entirely passive nature of information collection. As described in the analysis section, header transmission occurs automatically with each request and can be extracted by most web servers, such as Nginx, without significant overhead [21]. Since all processing takes place on the server side, this method remains invisible to the user and does not require client-side scripts, making the network traffic indistinguishable from regular requests. In summary, this method is efficient, unobtrusive, and compatible with most web servers, processing data on the server side without a noticeable impact on the client.

4) Disadvantages: Despite their utility, HTTP headers offer limited information due to the low entropy of most attributes. The User-Agent field, while informative, is widely recognized and can be manipulated using browser extensions like User-Agent Switchers (i.e., User-Agent Switcher for Chrome). Consequently, the reliability of this attribute alone should be critically assessed.

Additionally, the use of HTTP header-based fingerprinting without explicit user consent raises significant privacy concerns under the General Data Protection Regulation (GDPR). Therefore, any implementation should undergo legal review prior to deployment to ensure compliance with data protection regulations [22].

B. Enumeration of Browser Plugins

1) Definition and Basics: Browser plugins, whether pre-installed or user-added, have historically constituted one of the most significant methods for system recognition, alongside font detection. Most browser features are indirectly modified, with the exception of extensions, which maintain their popularity. The capability to obtain precise enumeration of these extensions remains highly sought after [15, pp. 878-880].

2) Analysis: Information-rich plugins, such as Flash, have gradually disappeared from the market. Since 2016, most browsers, including Firefox, no longer support the formerly widespread Netscape Plugin Application Programming Interface (NPAPI). This development has resulted in the detection of installed extensions via JavaScript and the navigator.plugins object in modern browsers primarily revealing only standard plugins like PDF viewers [23]. Although the removal of plugins represents significant progress for privacy protection, the limited capability to read certain plugins for compatibility purposes continues to provide opportunities to identify differences between systems and browsers, thereby enabling inferences about the system. Direct detection of user-installed add-ons is not possible, which restricts the significance of collectible data for fingerprinting [15, pp. 886-887].

Despite the impossibility of directly reading user-installed extensions, researchers have discovered novel methodologies for their enumeration. Chromium-based browsers possess the capability to access extension settings via a local URL. A project in GitHub exploits this vulnerability by requesting internal resources such as images for over 1,000 different extensions in the background. The status code can indicate whether the respective extensions are installed [24].

Ad blockers represent particularly popular add-ons, and their behavior in removing unwanted content from pages can also be detected. Ad blockers typically employ known lists of advertising companies and CSS elements for removal. A script can create such an element and verify whether it has been modified. With a sufficient dataset, the existence of deployed blocklists can be demonstrated [25].

Currently, it is also possible to read a portion of the programs installed on a device beyond extensions. A vulnerability in various browsers allows for reading the status of the handler protocol to determine whether the associated software is installed. Programs such as Skype and Zoom add these protocols within the system to enable launching the corresponding program with parameters via a link.

3) Advantages: Given that extensions are installed by users and considering the extensive market of available extensions, this method offers high uniqueness coupled with stability for fingerprinting purposes.

4) Disadvantages: This technique provides profound insights into the privacy of unsuspecting users. A study demonstrated that beyond less sensitive information like interests, extremely sensitive data can be inferred, including health conditions, practiced religion, and political views [26, pp. 11-12].

Since precise reading of extensions is not possible, this process relies on limited methods, making it error-prone. Therefore, continuous maintenance and updating are required to ensure its reliability.

C. Canvas Fingerprinting

1) Definition and Basics: Canvas fingerprinting represents a technique for generating a digital fingerprint through the utilization of the Canvas element introduced in HTML5. This methodology employs the Canvas API to render a 2D graphic imperceptibly in the background. The manner in which various browsers and devices process this image varies due to differences in hardware acceleration, installed fonts, and graphic libraries. The resultant fingerprint exhibits exceptional stability and uniqueness [3, pp. 1-3].

2) Analysis: A script embedded within a webpage incorporates an invisible Canvas element that renders a predetermined 2D graphic in the background. Utilizing the Canvas context, textual elements can also be rendered with diverse fonts and font sizes. WebFonts additionally facilitate the dynamic loading of fonts from the internet. These can be specifically selected according to purpose to evaluate systems for uniqueness in font rendering.

The resulting image data can be extracted via the functions *getImageData* and *toDataURL*, subsequently formatted into a fingerprint as desired, for instance, through the application of a simple hashing algorithm. The hash is then transmitted to a server via a web request for processing and storage.

Beyond storing the fingerprint for subsequent identification, an alternative application methodology involves comparing the fingerprint with an extensive database of known fingerprints and corresponding system configurations. With a substantial dataset, reliable predictions regarding the system's configuration can be established [3, pp. 2-4].

3) Advantages: The research findings of Mowery and Shacham demonstrated that the implementation of Canvas fingerprinting is exceptionally straightforward, requiring minimal lines of client-side code. It utilizes fundamental JavaScript functions and is deployable across all common web applications. The fingerprint generation process occurs inconspicuously for the user and presents significant challenges for blocking. This is attributable to the frequent deployment of Canvas operations on the web and the complex challenge of distinguishing normal applications from fingerprinting scripts.

The creation of the fingerprint, due to its simplicity, can be executed with high velocity and exhibits high stability in conjunction with high uniqueness and entropy. Consequently, its application is particularly valuable in real-time tracking applications [3, pp. 1-5].

4) Disadvantages: Alterations in browser environments, such as updates or graphic settings, may influence the stability of the fingerprint. Additionally, the variability of hardware and software configurations can lead to inconsistencies. As an active technique, the execution of code on the client side is unavoidable and entails the risk of detection and potential blockage by, for example, blocklists targeting known fingerprinting scripts [3, pp. 3-7].

Although the utilization is imperceptible to the user, the limited number of interfaces for retrieving generated Canvas data ensures that these can be monitored and manipulated by extensions [10, p. 4]. Add-ons such as CanvasBlocker exploit this to provide users with the option to prevent data extraction or manipulate the data in the Canvas, thereby generating a continuously new fingerprint and rendering identification impossible [27].

Finally, while the implementation of Canvas fingerprinting is relatively straightforward, the data analysis and interpretation are comparatively complex and may require a certain level of expertise in the field to be processed correctly [3, pp. 6-8].

D. WebGL Fingerprinting

1) Definition and Basics: WebGL fingerprinting is a technique utilizing the WebGL JavaScript API, based on OpenGL ES 2.0, allowing web applications to render both 2D and 3D graphics with high performance by directly accessing the GPU [28]. Unlike Canvas fingerprinting, which focuses on 2D graphics and identifies software differences mainly through fonts and graphic libraries, WebGL fingerprinting provides deeper and more precise detection capabilities. It captures unique hardware information, particularly details about the graphics processor, distinguishing it significantly from Canvas fingerprinting and broadening its application for tracking purposes [3, p. 4]. The inherent trade-offs between WebGL and Canvas fingerprinting ensure that neither method entirely supplants the other; their complementary nature makes them suitable for different scenarios.

2) Analysis: WebGL fingerprinting uses a Canvas element to access the API. Similar to Canvas fingerprinting, it creates an invisible element performing 3D operations in the background to collect data without user interaction. A straightforward application involves accessing specific variables, such as *UNMASKED_VENDOR_WEBGL* and *UNMASKED_RENDERER_WEBGL*, using the *getParameter* function in the WebGL context. These variables provide information about the graphics hardware manufacturer (Vendor) and model (Renderer). For example, a Vendor entry like "Intel" indicates an integrated graphics unit, while "Nvidia" combined with "GeForce GTX 970" as Renderer indicates a dedicated graphics card. These details can reveal insights into the system being used [29, p. 17]. Privacy concerns have led browsers like Apple's WebKit to provide generic information instead of specific data to protect user privacy. Since 2020, WebKit has masked Vendor and Renderer information, as well as shading language details [30]. Firefox similarly groups graphics processor models into categories instead of displaying specific models [31]. In practice, this means that an Nvidia card from the 900 series onward, for example, is reported as "GeForce GTX 980" [32]. In summary, research investigating hardware fingerprinting using HTML5 demonstrated the capability to identify devices based on GPU performance. It utilizes the graphics processor's clock frequency and clock skew to render complex 3D graphics, measuring GPU performance based on the number of frames rendered within a period, providing insights into the GPU's frequency and core count [33, pp. 3-4].

Furthermore, WebGL fingerprinting can render graphics, employing techniques like shadows, textures, lighting, anti-

aliasing, and transparency, to generate system-specific unique outputs. However, the three-dimensional environment results in increased client-side resource utilization and more complex code compared to the simpler 2D Canvas [3, p. 4]. While Laperdrix et al. initially deemed WebGL unreliable for fingerprinting in 2016, subsequent research demonstrated otherwise. Cao et al. [34] refuted Laperdrix et al.'s findings, attributing the inconsistencies to non-standardized rendering tasks and uncontrolled variables such as canvas size and anti-aliasing settings. By implementing 20 consistently defined tasks rendered under carefully controlled parameters, Cao et al. achieved a 99.24% success rate, surpassing Laperdrix et al.'s 90.84%. Their work also demonstrated the ability to identify a system across different browsers with a 91.44% stability [34, p. 2].

To enhance fingerprint stability, the "DrawnApart" project focuses on subtle variations in GPU Execution Units (EUs) rather than relying on differences in graphic rendering. This method exploits the unique characteristics of a device's GPU stack to detect speed variations across different EUs, creating a robust and reliable GPU signature. Experiments involving over 2,500 devices showed a fingerprint stability increase of up to 67% compared to other current techniques [35, pp. 1, 6-12].

3) Advantages: As demonstrated by Cao et al., WebGL can offer high uniqueness and stability [34]. Its direct interface with the system ensures consistency across browsers, making it challenging for users to evade identification through simple browser changes or reinstalls. Despite changes to enhance WebGL's resistance to fingerprinting, it reliably identifies users. The successor to WebGL, WebGPU, is currently in development, promising even more privacy risks due to its closer hardware access, allowing for classifications with up to 98% accuracy in 150 milliseconds, a reduction from the 8 seconds WebGL took [36].

4) Disadvantages: The complexity of WebGL fingerprinting is significantly higher compared to previous techniques, necessitating careful consideration whether a simpler Canvas approach combined with other methods might be accurate enough for specific use cases. Intensive tasks in a 3D environment can also strain the target system, leading to longer fingerprint creation times [3, p. 4]. Implementing WebGL requires caution, as shown by the cases of Laperdrix et al. and Cao et al., and opting for a ready-made solution might be advisable. Moreover, WebGL shares Canvas's vulnerability to blocked or misread data if detection methods rely on differences in rendered graphics. Even novel methods like DrawnApart can be mitigated through countermeasures, such as limiting to a single EU [35, p. 12]. WebGL may also not be available or disabled on some devices, necessitating consideration of alternatives, such as using the 2D Canvas.

E. Audio Fingerprinting

1) Definition and Basics: The Web Audio API is a JavaScript interface for processing and synthesizing audio signals in web browsers, part of the HTML5 standard. It can identify systems through manufacturing differences in audio hardware. Methods analyze signal processing characteristics, hardware differences, and system responses to specific audio signals for fingerprinting [37, pp. 1107-1109]. The API's indirect access to audio hardware allows for system identification based on subtle variations introduced during manufacturing.

2) Analysis: Audio fingerprinting involves various acoustic measurements to create a unique device fingerprint. It requires an AudioContext linking an AudioBuffer, Oscillator, and Compressor. The AudioBuffer represents a small audio segment, while the Oscillator generates a waveform at a defined frequency using a mathematical function. The Compressor manipulates the audio signal. The unique waveform generated and manipulated reflects system characteristics, allowing a unique fingerprint to be created by applying a hash function to the final waveform [38], [39]. This method, known as "Dynamic Compressor (DC)", is highly stable, producing the same fingerprint for the user each time using a reliable hash function [37, pp. 1109-1111].

Another method is the "Fast Fourier Transform" (FFT), converting audio signals from the time domain to the frequency domain. It measures hardware implementation differences to identify characteristics. FFT is less stable than DC, often requiring multiple attempts for consistent results. DC and FFT are often used together for more reliable outcomes [37, pp. 1111-1114].

Researchers from New Orleans compared various techniques, including custom-designed ones, alongside DC and FFT. These included creating "Custom Signals", "Merged Signals", and analyzing generated AM and FM waves. All techniques showed good stability, averaging two to four attempts for fingerprint matching [40, pp. 3-5].

3) Advantages: The generated fingerprints are highly stable and can differentiate systems based on their properties. Queiroz and Feitosa showed that mobile devices using Firefox could be consistently recognized and grouped by their stable fingerprints [37, p. 1119]. Techniques like DC are simple to implement and offer high stability. Other promising techniques, especially when used together, could enhance potential but are more challenging to implement [40, pp. 1-3].

4) Disadvantages: While audio fingerprinting offers high stability, it lacks uniqueness and accuracy on its own and should be used with other fingerprinting techniques [37, p. 1119]. Additionally, the Web Audio API can be disabled on devices or manipulated by add-ons like "Canvas Blocker", which also blocks and manipulates Canvas and WebGL [27].

F. Font Fingerprinting

1) Definition and Basics: Font fingerprinting is a browser fingerprinting technique that identifies devices by recognizing installed fonts. This method operates on the premise that each device possesses a specific combination of fonts. This combination can be unique or, when combined with other fingerprinting techniques, contribute to a unique and relatively stable digital fingerprint. Installed fonts are among the more unique identifiers of a device, often providing the

highest entropy, especially when considered alongside other data points such as installed plugins, information gleaned via the Canvas API, and the browser's User-Agent [20, p. 314]. These elements together enable the creation of a detailed and individualized device profile, which can be used for tracking and identification purposes.

2) Analysis: Until the end of 2020, Adobe Flash was frequently used to enumerate installed fonts. With the deprecation of Flash Player and its removal from common browsers, new methods had to be developed [34, p. 10]. Since pure JavaScript does not offer a direct function to detect installed fonts, a fallback mechanism is employed. This involves applying a specific font, and if it is unavailable, the system defaults to a standard fallback font. The technique leverages the different dimensions that fonts require to render the same text. A text string is rendered in a specific font, and the resulting dimensions are compared to expected values. This allows the determination of whether a specific font is available or a fallback was used [20, p. 311].

Using JavaScript, invisible *div* elements can be created, containing selected texts with specified fonts. The dimensions of the element are then compared with known target values, and a match is recognized as the font being installed. The list of all installed fonts can then be combined into a fingerprint via a hash algorithm [20, p. 311].

Another method is using the Canvas element. As described in Section III-C, the Canvas can render texts in requested fonts and use fallbacks if these are not available. Unlike direct text, the Canvas element has a fixed size, but the *measureText* function of the Canvas context allows reading the width of the drawn text, allowing further inferences about available fonts [41, p. 12].

It should be noted that JavaScript under Chrome and Edge currently allows reading local fonts, but the Local Font Access API used for this is experimental, only available in these two browsers, and requires user consent, making it unsuitable for fingerprinting purposes [42].

3) Advantages: Font recognition offers high entropy and stability since fonts are rarely changed. Fonts can be installed by the user or by software, with each operating system pre-installing different fonts. This allows the identification of the operating system and potentially its version, as manufacturers can make adjustments. It also allows the detection of installed software packages like Office or Photoshop, which installs fonts for use [5, p. 7].

4) Disadvantages: Without Flash, font recognition is done through "brute-force" methods, reducing accuracy if unknown fonts are installed. This requires selecting a list of fonts to test and measuring them against the values to be tested. If fonts are installed that are not within the list, they cannot be detected, reducing the accuracy of the result [34, p. 10]. Another problem is fonts that have too strong similarities in their dimensions to possible fallbacks. This can lead to false positives, so a forced fallback test should be performed for a text beforehand. Since the fallback font is unknown, a non-existent font is requested, and the resulting dimensions are used to recognize other non-existent fonts [20, p. 311].

Finally, it is still possible to manipulate the read fonts through extensions [43] or, as in the case of Apple's WebKit, to only deliver values pre-installed by the operating system, causing users to blend into the crowd [44].

G. Screen Fingerprinting

1) Definition and Basics: Screen fingerprinting identifies a device by analyzing various screen-related characteristics, including screen resolution, pixel depth, color depth, and browser window size. This method leverages the uniqueness of screen configurations and browser modifications, which can create rare resolution combinations [45, p. 20].

2) Analysis: JavaScript provides attributes for screen and browser window characteristics through the *window.screen object*, offering details like color depth (*colorDepth*), screen orientation (*screenOrientation*), and screen dimensions (*screenHeight*, *screenWidth*). Values, such as *window.innerWidth* and *window.innerHeight*, determine the browser window's inner area, which can be altered by toolbars or bookmark bars [34, p. 3].

3) Advantages: Screen and window resolution information typically have high entropy, making them useful for stabilizing fingerprints when combined with other techniques. This method is particularly effective for distinguishing between desktop, tablet, and mobile devices, as these have distinct resolutions and aspect ratios compared to standardized desktop screens [37, p. 277].

4) Disadvantages: Since values are derived from browser attributes rather than hardware tests, they can be limited or altered by extensions or privacy settings. Browsers like TOR set the window to a fixed size of 1000x1000 pixels, reducing uniqueness, and browsers like Firefox always report a color depth of 24. Additionally, users with multiple monitors or those using zoom functions can affect the accuracy of screen fingerprinting, as there is no reliable way to determine the zoom factor directly, which reduces entropy [34, p. 10].

H. WebRTC Fingerprinting

1) Definition and Basics: WebRTC is a standard and accessible JavaScript interface available in most browsers. It facilitates real-time communication over stateless HTTP by establishing direct connections between participants, allowing the extraction of local network adapter information. This can reveal private and public IP addresses, which can be used for fingerprinting or identifying users behind proxies or VPNs [41, p. 12]. It also provides information about connected devices, such as microphones, webcams, and speakers.

2) Analysis: Unlike other browser mechanisms like camera or microphone access, establishing a WebRTC connection requires no permissions or user notifications. After successfully connecting to the target computer via a Session Traversal Utilities for NAT (STUN) server, the individual addresses can be read from the RTCPeerConnection object in the form of iceCandidates [46, p. 667].

This data can be used for fingerprinting, but the data collection does not have to stop there. Since WebRTC always seeks the shortest path for a connection, it is possible to enumerate the local network through, for example, port scanners, creating a unique picture of the target's environment. Furthermore, it is possible to read all local addresses of the adapters, which, in addition to connections to VPNs, can also include set-up virtual adapters for Virtual Machines [46, pp. 667-668].

The DetectRTC project [47] demonstrates what functions are directly available through WebRTC. The most important are the microphones, webcams, and speakers. However, the exact device names are not possible without the necessary permissions. WebRTC does, however, allow reading the Media Device IDs of the respective devices, which, in connection with the respective active WebRTC functions, lead to unique fingerprints [48].

3) Advantages: Extracting private and public IPs provides deep insights, especially for identifying targets behind VPNs or proxies. No other technique can silently reveal addresses behind Network Address Translation (NAT) [49, p. 273]. The collected data is highly unique; a study with 80 devices found over 97% uniqueness using only WebRTC [46, p. 668].

4) Disadvantages: WebRTC might be disabled in the target browser, or extensions might block its usage without user consent. To read the Media IDs of the devices, a request for access rights for the respective devices is required. This can alert the user that a page may be performing dubious actions in the background. This is therefore not recommended for a secret operation.

Finally, WebRTC requires an infrastructure in the form of a STUN server, which must be set up independently or used by third parties. This makes it a technique that requires further dependencies and should therefore be considered depending on the intended use.

I. CSS Fingerprinting

1) Definition and Basics: Different to the active fingerprinting techniques using JavaScript, CSS fingerprinting is a passive method. CSS is a stylesheet language primarily used to enhance the presentation of HTML elements. Over time, the CSS specification has expanded to include selectors and filters, enabling limited dynamic selections, which this technique leverages [50, p. 10].

2) Analysis: Until 2010, the *:visited* selector could identify if a website had been visited by changing the link color, detectable via JavaScript. This was possible because browsers displayed already visited links in a different color, and this color difference was read out by JavaScript. After this was patched, researchers explored time-based methods to read user history, but these required JavaScript and were impractical [51, p. 4].

In 2015, Takei et al. introduced a JavaScript-free method using CSS properties and multiple *@media* queries to fetch URLs based on defined rules. By considering the requesting IP address and URL parameters, the server could then identify system properties like screen dimensions, resolution, touchscreen presence, installed fonts, browser, and OS [52, pp. 3-5]. A current GitHub project demonstrates this method's practical capabilities [53]. Individual CSS properties were used together with a variety of *@media* queries to call up URLs according to defined rules.

3) Advantages: CSS fingerprinting's independence from JavaScript allows it to identify even cautious users who block JavaScript or use extensions like NoScript. Software projects like TOR usually block JavaScript or use extensions like NoScript to give the user the possibility to execute selected scripts. This technique can even detect if JavaScript is disabled via noscript tags [52, p. 2]. Since this method is currently little used and rather unknown, further research has shown that no practical solution currently exists for users to effectively prevent it.

4) Disadvantages: Takei et al.'s method provides limited data, which, without JavaScript, can only be supplemented by techniques like header analysis (as presented in Section III-A). Oliver Brotchie notes in his project repository that the method is not currently scalable, as each request requires over 1MB of CSS files to be downloaded. However, he warns that upcoming CSS Values 4 implementation could reduce download sizes significantly, making the method more practical. Additionally, font recognition relies on brute-forcing, which, considering network traffic, can be noticeable [53]. The font recognition, as presented in Section III-F, is based on the principle of brute forcing, i.e., the massive trying out of fonts, which can be conspicuous when considering the network traffic.

J. Additional JavaScript Attributes

1) Definition and Basics: Most of the previously discussed techniques actively use JavaScript to extract information from various interfaces. Additional possibilities are briefly mentioned here to provide a more comprehensive picture. Since these techniques share many characteristics with other JavaScript-based methods, listing their pros and cons is omitted.

2) Analysis: The *navigator* object in browsers provides information, such as DoNotTrack status, user agent details, platform, languages, cookies usage, granted and available permissions, and time zone [20, p. 9]. JavaScript implementation varies between browsers and versions, and Mowery et al. demonstrated that these differences are measurable and can indicate the software and hardware used [3].

However, the implementation of JavaScript itself can also vary from browser to browser and version to version. Mowery et al. proved in 2011 that the different implementations of functions are measurable and can therefore provide a conclusion about the software and hardware used [3]. In addition to the differences in the execution itself, there are also differences in whether various functions are built into the browser and usable on the platform. This offers an alternative way of UserAgent detection, should this have been manipulated by extensions, for example [54].

Another technique caused uncertainty among Tor users in the past. Despite disabled Canvas, the *getClientRects* function could be used to obtain the exact data of DOM elements. Similar to the Canvas fingerprint, these factors could change greatly depending on implementation, font sizes, and screen resolutions, enabling identification in the otherwise anonymous browser [55]. The vulnerability has been fixed in Tor but remains exploitable in other browsers [56].

3) Advantages: JavaScript-based fingerprinting techniques are highly versatile and widely applicable since JavaScript is essential for web functionality. These methods can collect a broad range of information, such as user agent details, time zones, and system settings, often without requiring user consent or visibility. The stealthy nature of JavaScript fingerprinting allows it to operate in the background, making it difficult for users to detect. Moreover, JavaScript-based attributes work consistently across different browsers, enabling effective cross-browser tracking.

4) Disadvantages: However, JavaScript fingerprinting is limited by browser-specific implementations, which can result in inconsistent data collection. Privacy-focused browsers like Tor or extensions, such as NoScript, actively block or obscure JavaScript-based tracking, reducing its effectiveness. Additionally, users are becoming more aware of privacy risks and increasingly use tools to disable or modify JavaScript functions. Finally, updates to browsers may close vulnerabilities or alter features that JavaScript fingerprinting relies on, decreasing its long-term viability.

K. Advanced Techniques Using Machine Learning

1) Definition and Basics: Most active techniques discussed so far use JavaScript to gather hardware and software information. They rely on unique data combinations based on implementation quirks or directly available information. Newer methods often employ "side-channels", capturing additional data by observing behavioral differences during various operations within the execution environment. Methods like plugin enumeration (cf. Section III-B), font fingerprinting (cf. Section III-F), and CSS fingerprinting (cf. Section III-I) use this approach in simple forms by testing known combinations to gain indirect information. These side-channel methods can be implemented with minimal effort but can also be used in more sophisticated ways with machine learning to gather otherwise unobtainable information [57, p. 1].

2) Analysis: Wang et al. explored using techniques such as cache usage, memory consumption, and CPU activity to identify visited websites. In earlier methods, CSS selectors were leveraged to glean browsing history, revealing significant privacy risks and prompting swift remedial actions. Side-channel techniques utilize an array of strategies to yield more accurate analyses of system behavior. These methods involve complex calculations that impose a load on the hardware, with machine learning models categorizing the results against expected values from known sites. Their tests demonstrated an accuracy rate of 80-90% in identifying websites [57, pp. 3-5]. While Wang et al. addressed multiple attack vectors, including compromised machines with direct operating system access, the feasibility of executing such attacks solely through JavaScript measurements remains uncertain. Further research is needed, but implementations using WebAssembly [58] and the Performance API [59] are conceivable.

3) Advantages: This method is invisible to the user and provides insightful information not available through conventional means. Currently, there are no effective methods to protect users from such techniques [57, pp. 1-3].

4) Disadvantages: While previous techniques aimed to identify a user over time, this method has the potential to offer dangerous insights into the individual's behavior behind the screen. However, the technique is still in its initial stage and remains a theoretical approach not yet tested in in real-world scenarios. It is unlikely to be reliably utilized by malicious actors in the near future [57, p. 6].

IV. Discussion

Browser fingerprinting can be used positively for security, as shown by technologies like BrFast and private, passive user recognition methods. Such technologies offer promising alternatives for user authentication by leveraging device-specific attributes without the need for intrusive cookies or explicit user interaction. They provide a non-invasive method to identify users, particularly for fraud detection and bot prevention. However, there's a significant risk of misuse, especially in the field of advertising and mass surveillance. The advertising industry, driven by creating accurate user profiles, heavily invests in digital advertising, with data-driven ads accounting for 60-70% of digital ad revenue in Germany. Personalized ads significantly impact Generation Z, who discover products primarily through social media and whose purchasing decisions are increasingly influenced by algorithmic recommendations.

Traditionally, data collection relied on cookies, but users developed ways to avoid tracking, such as deleting cookies or using incognito mode. However, unlike cookies, browser fingerprints are collected in the background, making them invisible and far more persistent. Fingerprints are difficult to alter, and their cross-browser and cross-device capabilities exacerbate the problem by enabling long-term tracking across multiple platforms [34]. GDPR regulations mandate user consent for data collection, but enforcement is inconsistent, and compliance with fingerprinting guidelines remains unclear, even with new laws like Germany's Telecommunications Telemedia Data Protection Act (TTDSG) [60].

A. Affected Demographics

Online tracking is ubiquitous, affecting nearly all user groups. A 2016 study of the top 1 million websites revealed extensive tracking, with services like Google and Facebook present on over 10% of sites [41]. Following the GDPR, fingerprinting scripts increased significantly, with 68.8% of the top 10,000 websites employing such methods by 2020 [10]. This shift illustrates how fingerprinting has replaced traditional cookie-based tracking in response to regulatory pressure.

TABLE I
OVERVIEW OF FINGERPRINTING METHODS

Fingerprinting Method	Uniqueness	Stability	Entropy	Impact on User Privacy	Defense Techniques
HTTP Header Attributes	Low	Moderate	Low	Moderate impact: limited detail but useful when combined with other methods.	Altering or masking headers (e.g., randomizing User-Agent).
Enumeration of Browser Plugins	Moderate	High	High	High impact: reveals sensitive data, such as installed plugins.	Disabling plugin enumeration, avoiding unnecessary add-ons.
Canvas Fingerprinting	High	Moderate	High	High impact: generates unique fingerprints based on rendering.	CanvasBlocker extension to block or manipulate rendering.
WebGL Fingerprinting	High	High	High	High impact: collects detailed hardware data for tracking.	Block or manipulate WebGL outputs.
Audio Fingerprinting	Moderate	High	Moderate	High impact: captures unique audio processing details.	Disable Web Audio API, use privacy extensions.
Font Fingerprinting	High	High	Moderate	High impact: identifies installed fonts, making it persistent.	Limit font access with privacy-focused browsers (e.g., Tor).
Screen Fingerprinting	Moderate	High	Low	Moderate impact: uses screen resolution and window size but less effective on mobile devices.	Fix window size or limit resolution reporting with privacy browsers.
WebRTC Fingerprinting	Very High	High	Very High	Very high impact: exposes real IP addresses, even behind VPNs.	Disable WebRTC, use extensions that block data collection.
CSS Fingerprinting	Low	Moderate	Low	Low impact: provides limited system and style information.	Limit or disable CSS fingerprinting through extensions or scripts.
JavaScript Attributes	Moderate	High	Moderate	Moderate impact: uses various browser features for tracking.	Disable unnecessary JavaScript functions or use privacy extensions.
Advanced Machine Learning Fingerprinting	Very High	Very High	Very High	Very high impact: uses side-channel data (e.g., CPU/cache) for tracking.	Limit access to Performance API and WebAssembly, emerging defenses needed.

However, fingerprinting does not affect all users equally. A study with 234 participants found that demographics like age, gender, education, IT background, and privacy awareness influence trackability. Men and those with higher education were found to be less trackable, while users with lower privacy knowledge or older devices were more easily identified [61]. Despite this, many participants believed they could protect themselves from fingerprinting, underestimating its stealth and technical complexity.

Additionally, fingerprinting poses a disproportionate risk to marginalized communities. Research by Queiroz and Feitosa shows that low-income users and those in the Global South — who are more likely to use older mobile devices — are significantly more identifiable through audio fingerprinting [37]. This privacy divide creates a vulnerability gap, where the users least capable of protecting themselves are the most exposed.

B. Convergence of Fingerprinting Techniques

Browser fingerprinting, as explored through various methods in this paper (cf. Table I), represents a comprehensive and evolving threat to digital privacy. Each fingerprinting technique, from HTTP Header Attributes to sophisticated methods like Canvas, WebGL, and Audio Fingerprinting, offers unique data points, but their power lies in their combinatorial use. This synergistic exploitation of passive and active methods creates a multi-dimensional profiling system capable of identifying users with extraordinary precision and stability.

The cross-browser stability of WebGL and machine learning-based techniques enables tracking across different devices and sessions, while WebRTC Fingerprinting reveals network-level information like private IP addresses. These methods complement traditional fingerprinting approaches by exposing additional system and network data layers, making countermeasures significantly more difficult.

Furthermore, machine learning-based fingerprinting represents the next evolutionary step in this domain. Research by Wang et al. demonstrated that side-channel attacks exploiting CPU cache timing and memory consumption can identify users with up to 90% accuracy without relying on any standard browser attributes [57]. This convergence of fingerprinting techniques into multi-layered profiling systems renders current countermeasures increasingly ineffective.

C. Ethical and Legal Implications

The stealthy nature of browser fingerprinting raises significant ethical concerns regarding user autonomy and consent. Although the GDPR explicitly defines personal data as any information that can identify an individual, browser fingerprinting often circumvents this regulation under the guise of legitimate interest [62].

However, recent court rulings suggest a tightening regulatory landscape. In 2023, the French data protection authority CNIL fined Criteo for failing to obtain consent for fingerprinting-based tracking, marking one of the first legal cases explicitly addressing browser fingerprinting under GDPR.

Nonetheless, global regulatory frameworks remain fragmented, and the majority of fingerprinting scripts operate without user knowledge or legal verification. This regulatory vacuum risks turning browser fingerprinting into a normalized surveillance practice embedded within the digital economy.

D. Towards Privacy-Respecting Fingerprinting

While fingerprinting is primarily associated with surveillance, several emerging technologies seek to repurpose it for privacy-enhancing applications. Projects like BrFast [15] and Apple's Private Access Tokens leverage ephemeral, cryptographically unlinkable fingerprints to authenticate users without persistent tracking.

However, the implementation of privacy-respecting fingerprinting requires transparent system design and regulatory oversight. Without proper safeguards, even privacy-preserving systems risk reinforcing the same surveillance mechanisms they aim to replace.

E. Future Outlook

The future of browser fingerprinting lies in the convergence of machine learning, side-channel attacks, and cross-device tracking. This hybrid approach creates persistent, adaptive tracking systems capable of circumventing existing countermeasures.

Future research should prioritize:

- Developing adaptive defenses against machine learning-assisted fingerprinting.
- Investigating cross-device tracking prevention methods.
- Designing transparent fingerprinting APIs that separate security-related use cases from surveillance.
- Studying the privacy divide and ethical implications of fingerprinting on vulnerable populations.

F. Consequences

Browser fingerprinting represents one of the most pervasive and least transparent forms of online tracking. Its rapid evolution from basic HTTP headers to machine learning-assisted side-channel attacks highlights the growing asymmetry between users and data collectors. The convergence of passive and active methods creates a multi-dimensional profiling system that is increasingly resistant to countermeasures, challenging both privacy frameworks and user efforts to remain anonymous online.

Despite its invasive applications, fingerprinting could also be repurposed for privacy-enhancing authentication systems — provided that transparent design principles and strict regulatory safeguards are enforced. Bridging the gap between security and privacy will be one of the defining challenges of digital privacy in the coming decade.

V. Conclusion

In this final section, the paper synthesizes its findings to assess the broader impact of browser fingerprinting on digital privacy. It reflects on the dual-use nature of fingerprinting—both as a security tool and as a surveillance threat—and reaffirms the urgent need for stronger countermeasures, privacy-oriented browser practices, and regulatory interventions. The conclusion also identifies key areas for further research and policy action, emphasizing that safeguarding user anonymity in the digital space requires a coordinated effort between technologists, regulators, and informed users.

A. Summary of the Research Outcome

This contribution has examined browser fingerprinting, a growing technique in online tracking. It has demonstrated that browser fingerprinting is a sophisticated method for identifying and tracking users online without traditional methods like cookies.

The analysis highlighted that browser fingerprinting poses a complex challenge from both technical and privacy perspectives. While it provides companies and advertisers with detailed insights into user behavior for targeted advertising, it raises significant privacy concerns as users are often tracked without their knowledge or consent. Despite stricter privacy laws like the GDPR in the EU, browser fingerprinting remains a grey area. Anti-fingerprinting techniques are limited and continually evolving to keep up with new tracking methods.

In conclusion, browser fingerprinting plays and will continue to play a significant role in the digital landscape. Both users and regulatory bodies must increase awareness of browser fingerprinting practices and their implications.

B. Implications for Practice

Consent and Cookies: Always accept only the necessary cookies in cookie banners and regularly delete cookies to hinder tracking and fingerprinting. This is particularly important for news sites, which often misuse collected data without user consent.

Blending in with the Masses: Reducing APIs and data sources for fingerprinting can ironically make users more identifiable [63]. Thus, widely adopted browsers and protection mechanisms should be used to stay less conspicuous.

Browser Choice: Choose browsers with robust privacy protections. On iOS, Safari is recommended due to its advanced tracking protection and large user base [64]. For Android, the Mull browser is highly rated for fingerprinting protection, while Brave is a good, widely-used alternative. On desktops, Brave, Librewolf, and Mullvad browsers are recommended for their privacy features and user bases [65].

Browser Extensions: Limit the use of browser extensions, as they can become sources of unique information. While some extensions block known trackers or modify API outputs, these protections are often already built into recommended browsers like Brave and Librewolf [26] [63].

C. Future Research

Future research in browser fingerprinting should focus on several key areas. First, countermeasures and defense mechanisms need to be explored further, especially in mitigating the newer techniques that leverage machine learning and side-channel attacks. These advanced methods can bypass traditional privacy safeguards, such as disabling JavaScript or using incognito modes, making the development of more robust anti-fingerprinting technologies imperative. Additionally, research should explore the ethics and regulatory frameworks surrounding fingerprinting, examining how existing privacy and data protection laws like GDPR can be adapted to better address fingerprinting practices. Another promising direction

is improving cross-device tracking prevention by understanding how fingerprinting works across different platforms and hardware. Lastly, investigating user awareness and educational tools on fingerprint privacy risks will help empower the general public to protect their digital identities more effectively. Thus, future research should focus on developing more effective privacy techniques to balance commercial interests and user privacy rights.

REFERENCES

[1] A. Lawall, "Fingerprinting and Tracing Shadows: The Development and Impact of Browser Fingerprinting on Digital Privacy," in *Proceedings of the IARIA SECURWARE 2024 Conference*. IARIA, November 2024, pp. 132–140.

[2] ——, "Fingerprinting and Tracing Shadows: The Development and Impact of Browser Fingerprinting on Digital Privacy," *arXiv preprint arXiv:2411.12045*, 2024.

[3] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in HTML5," *Proceedings of W2SP*, vol. 2012, 2012.

[4] K. Szymielewicz and B. Budington. (2018) The GDPR and Browser Fingerprinting: How It Changes the Game for the Sneakiest Web Trackers. Accessed: 2024-09-27. [Online]. Available: https://www.eff.org/de/deeplinks/2018/06/gdpr-and-browser-fingerprinting-how-it-changes-game-sneakiest-web-trackers

[5] D. Zhang, J. Zhang, Y. Bu, B. Chen, C. Sun, and T. Wang, "A Survey of Browser Fingerprint Research and Application," *Wireless Communications and Mobile Computing*, vol. 2022, no. 1, p. 3363335, 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/2022/3363335

[6] P. Eckersley, "How unique is your web browser?" in *Privacy Enhancing Technologies*, M. J. Atallah and N. J. Hopper, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–18.

[7] S. Englehardt and A. Edelstein. (2021) Firefox 85 Cracks Down on Supercookies. Accessed: 2024-09-27. [Online]. Available: https://blog.mozilla.org/security/2021/01/26/supercookie-protections/

[8] E. Woollacott. (2021) Browser fingerprinting more prevalent on the web now than ever before. Accessed: 2024-09-27. [Online]. Available: https://portswigger.net/daily-swig/browser-fingerprinting-more-prevalent-on-the-web-now-than-ever-before-research

[9] R. Koch. (2019) Cookies, the GDPR, and the ePrivacy Directive. Accessed: 2024-09-27. [Online]. Available: https://gdpr.eu/cookies/

[10] U. Iqbal, S. Englehardt, and Z. Shafiq, "Fingerprinting the Fingerprinters: Learning to Detect Browser Fingerprinting Behaviors," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 05 2021, pp. 1143–1161.

[11] G. Acar. (2014) Browser Fingerprinting and the Online-Tracking Arms Race. Accessed: 2024-09-27. [Online]. Available: https://www.esat.kuleuven.be/cosic/news/the-web-never-forgets-persistent-tracking-mechanisms-in-the-wild/

[12] "Request header," accessed: 2024-09-27. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/Request_header

[13] "HTTP/2 fingerprinting: A relatively-unknown method for web fingerprinting," accessed: 2024-09-27. [Online]. Available: https://lwthiker.com/networks/2022/06/17/http2-fingerprinting.html

[14] "HTTP Headers - MDN Web Docs," accessed: 2024-07-27. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

[15] P. Laperdrix, W. Rudametkin, and B. Baudry, "Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints," in *2016 IEEE Symposium on Security and Privacy (SP)*, 2016, pp. 878–894.

[16] "User-Agent," accessed: 2024-09-27. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent

[17] "Accept - HTTP," accessed: 2024-02-27. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Accept

[18] "Content-Encoding - HTTP," accessed: 2024-02-27. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Encoding

[19] "Content-Language - HTTP," accessed: 2024-02-27. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Language

[20] A. Gómez-Boix, P. Laperdrix, and B. Baudry, "Hiding in the Crowd: an Analysis of the Effectiveness of Browser Fingerprinting at Large Scale," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2018, p. 309–318. [Online]. Available: https://doi.org/10.1145/3178876.3186097

[21] NGINX, "Managing request headers," accessed: 2024-04-22. [Online]. Available: https://nginx.org/en/docs/http/ngx_http_headers_module.html

[22] B. Wolford. (2024) What are the GDPR consent requirements? Accessed: 2024-09-27. [Online]. Available: https://gdpr.eu/gdpr-consent-requirements/

[23] "Navigator: plugins property," accessed: 2024-09-27. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Navigator/plugins

[24] "Extension Detector," accessed: 2024-09-27. [Online]. Available: https://github.com/z0ccc/extension-detector

[25] "How ad blockers can be used for browser fingerprinting," accessed: 2024-09-27. [Online]. Available: https://fingerprint.com/blog/ad-blocker-fingerprinting/

[26] S. Karami, P. Ilia, K. Solomos, and J. Polakis, "Carnus: Exploring the Privacy Threats of Browser Extension Fingerprinting," in *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020.

[27] "CanvasBlocker," accessed: 2024-09-27. [Online]. Available: https://github.com/kkapsner/CanvasBlocker

[28] "WebGL: 2D and 3D graphics for the web," accessed: 2024-09-27. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API

[29] T. Stephenson, "A Comparative Study on Analyses of Browser Fingerprinting," Ph.D. dissertation, Wesleyan University, 2023.

[30] "WebKit," accessed: 2024-09-27. [Online]. Available: https://github.com/WebKit/WebKit/commit/ae710d34c23858295b385e3f95ad7f6edd29f9d7

[31] S. Lee, Y. Kim, J. Kim, and J. Kim, "Stealing webpages rendered on your browser by exploiting gpu vulnerabilities," in *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 19–33.

[32] S. J. Vaughn-Nichols, "Vendors draw up a new graphics-hardware approach," *Computer*, vol. 42, no. 05, pp. 11–13, 2009.

[33] G. Nakibly, G. Shelef, and S. Yudilevich, "Hardware Fingerprinting Using HTML5," *arXiv preprint arXiv:1503.01408*, 03 2015.

[34] Y. Cao, S. Li, and E. Wijmans, "(Cross-)Browser Fingerprinting via OS and Hardware Level Features," in *Network and Distributed System Security Symposium*, 2017.

[35] Laor et al., "DRAWNAPART: A Device Identification Technique based on Remote GPU Fingerprinting," *ArXiv*, vol. abs/2201.09956, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:246276013

[36] M. Mantel. (2022) Browser-Fingerprinting: PCs, Smartphones & Co. lassen sich über die GPU tracken. Accessed: 2024-09-27. [Online]. Available: https://www.heise.de/news/Browser-Fingerprinting-PCs-Smartphones-Co-lassen-sich-ueber-die-GPU-tracken-6345233.html

[37] J. S. Queiroz and E. L. Feitosa, "A Web Browser Fingerprinting Method Based on the Web Audio API," *Comput. J.*, vol. 62, pp. 1106–1120, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:86644316

[38] T. S. Brandes, S. Kuzdeba, J. McClelland, N. Bomberger, and A. Radlbeck, "Rf waveform synthesis guided by deep reinforcement learning," in *2020 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2020, pp. 1–6.

[39] E. Cheek, D. Khuttan, R. Changalvala, and H. Malik, "Physical fingerprinting of ultrasonic sensors and applications to sensor security," in *2020 IEEE 6th International Conference on Dependability in Sensor, Cloud and Big Data Systems and Application (DependSys)*. IEEE, 2020, pp. 65–72.

[40] S. Chalise and P. Vadrevu, "A Study of Feasibility and Diversity of Web Audio Fingerprints," *arXiv preprint arXiv:2107.14201*, 2021.

[41] S. Englehardt and A. Narayanan, "Online Tracking: A 1-million-site Measurement and Analysis," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1388–1401. [Online]. Available: https://doi.org/10.1145/2976749.2978313

[42] "Local Font Access API," accessed: 2024-09-27. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Local_Font_Access_API
[43] "Font Fingerprint Defender," accessed: 2024-01-20. [Online]. Available: https://mybrowseraddon.com/font-defender.html
[44] "Tracking Prevention in WebKit," accessed: 2024-01-20. [Online]. Available: https://webkit.org/tracking-prevention/
[45] P. Laperdrix, N. Bielova, B. Baudry, and G. Avoine, "Browser Fingerprinting: A Survey," *ACM Trans. Web*, vol. 14, no. 2, apr 2020. [Online]. Available: https://doi.org/10.1145/3386040
[46] A. Reiter and A. Marsalek, "WebRTC: your privacy is at risk," in *Proceedings of the Symposium on Applied Computing*, ser. SAC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 664–669. [Online]. Available: https://doi.org/10.1145/3019612.3019844
[47] "DetectRTC," accessed: 2024-09-27. [Online]. Available: https://github.com/muaz-khan/DetectRTC
[48] "Fingerprinting WebRTC," accessed: 2024-01-20. [Online]. Available: https://privacycheck.sec.lrz.de/active/fp_wrtc/fp_webrtc.html
[49] V. Bernardo and D. Domingos, "Web-based Fingerprinting Techniques," in *Proceedings of the 13th International Joint Conference on E-Business and Telecommunications*, ser. ICETE 2016. Setubal, PRT: SCITEPRESS - Science and Technology Publications, Lda, 2016, p. 271–282. [Online]. Available: https://doi.org/10.5220/0005965602710282
[50] T. Bujlow, V. Carela-Español, J. Solé-Pareta, and P. Barlet-Ros, "A Survey on Web Tracking: Mechanisms, Implications, and Defenses," *Proceedings of the IEEE*, vol. 105, no. 8, pp. 1476–1510, 2017.
[51] L. Olejnik, C. Castelluccia, and A. Janc, "Why Johnny Can't Browse in Peace: On the Uniqueness of Web Browsing History Patterns," *12th Privacy Enhancing Technologies Symposium (PETS 2012)*, 07 2012. [Online]. Available: https://petsymposium.org/2012/papers/hotpets12-4-johnny.pdf
[52] N. Takei, T. Saito, K. Takasu, and T. Yamada, "Web Browser Fingerprinting Using Only Cascading Style Sheets," in *2015 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*, 2015, pp. 57–63.
[53] "CSS-Fingerprint," accessed: 2024-09-27. [Online]. Available: https://github.com/OliverBrotchie/CSS-Fingerprint
[54] "Feature Detection," accessed: 2024-01-23. [Online]. Available: https://privacycheck.sec.lrz.de/active/fp_fd/fp_feature_detection.html
[55] "Advanced Tor Browser Fingerprinting," accessed: 2024-09-27. [Online]. Available: http://jcarlosnorte.com/security/2016/03/06/advanced-tor-browser-fingerprinting.html
[56] "Investigate impact of fingerprinting via getClientRects()," accessed: 2024-09-27. [Online]. Available: https://gitlab.torproject.org/tpo/applications/tor-browser/-/issues/18500
[57] H. Wang, H. Sayadi, A. Sasan, P. D. Sai Manoj, S. Rafatirad, and H. Homayoun, "Machine Learning-Assisted Website Fingerprinting Attacks with Side-Channel Information: A Comprehensive Analysis and Characterization," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, 2021, pp. 79–84.
[58] "WebAssembly," accessed: 2024-09-27. [Online]. Available: https://developer.mozilla.org/en-US/docs/WebAssembly
[59] "High precision timing," accessed: 2024-09-27. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Performance_API/High_precision_timing
[60] "Browser Fingerprinting und das TDDDG: Erlaubt oder nicht? [Browser Fingerprinting and the TDDDG: Allowed or not?]," accessed: 2024-09-27. [Online]. Available: https://dr-dsgvo.de/browser-fingerprinting-und-das-ttdsg/
[61] G. Pugliese, C. Riess, F. Gassmann, and Z. Benenson, "Long-Term Observation on Browser Fingerprinting: Users' Trackability and Perspective," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, pp. 558–577, 05 2020.
[62] K. Szymielewicz and B. Budington. (2018) The gdpr and browser fingerprinting: How it changes the game for the sneakiest web trackers. Electronic Frontier Foundation (EFF). Accessed: 2024-09-27. [Online]. Available: https://www.eff.org/de/deeplinks/2018/06/gdpr-and-browser-fingerprinting-how-it-changes-game-sneakiest-web-trackers
[63] N. Al-Fannah and C. Mitchell, "Too little too late: can we control browser fingerprinting?" *Journal of Intellectual Capital*, vol. ahead-of-print, 01 2020.
[64] K. Kollnig, A. Shuba, M. Van Kleek, R. Binns, and N. Shadbolt, "Goodbye Tracking? Impact of iOS App Tracking Transparency and Privacy Labels," in *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 508–520. [Online]. Available: https://doi.org/10.1145/3531146.3533116
[65] X. Lin, F. Araujo, T. Taylor, J. Jang, and J. Polakis, "Fashion Faux Pas: Implicit Stylistic Fingerprints for Bypassing Browsers' Anti-Fingerprinting Defenses," in *2023 IEEE Symposium on Security and Privacy (SP)*, 2023, pp. 987–1004.