Information Modelling and Knowledge Bases XXXV
M. Tropmann-Frick et al. (Eds.)

doi:10.3233/FAIA231163

Browser Fingerprinting: Overview and Open Challenges

Marko HÖLBL[a,1], Vladimir ZADOROZHNY[b], Tatjana WELZER DRUŽOVEC[a], Marko KOMAPARA[a] and Lili NEMEC ZLATOLAS[a]
[a] *University of Maribor, Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia*
[b] *University of Pittsburgh, School of Computing and Information, Pittsburgh, USA*

ORCiD ID: Marko Hölbl https://orcid.org/0000-0002-9414-3189

Abstract. The central concept of browser fingerprinting is the collection of device-specific information for identification or security purposes. This chapter provides an overview of the research conducted in the field of browser fingerprinting and presents an entry point for newcomers. Relevant literature is examined to understand the current research in the field of browser fingerprinting. Both research in the field of crafting browser fingerprints and protection against it is included. Finally, current research challenges and future research directions are presented and discussed.

Keywords. Browser fingerprinting, profiling, user privacy, web tracking

1. Introduction

The web is a platform that we access using browsers. In recent years, with the introduction of technologies such as HTML5 and CSS3, the web has become more dynamic and utilized than ever before. Since the beginning of the web, we strive to improve the user experience by sharing device-specific information. However, this fact and the diversity of the devices connecting to the web have paved the way for device fingerprinting. A device fingerprint collects information about the software and hardware of a device for identification purposes. Typically, a fingerprinting algorithm consolidates the data into an identifier. A browser fingerprint is data collected specifically through interaction with a device's web browser [1]. This data is often needed for browsing to function adequately. Therefore, it cannot be remedied easily.

The concept of browser fingerprinting is simple – collect device-specific data for identification and security purposes through a browser. Websites are often required to track users to maintain a session for various reasons, such as maintaining logged-in status, language preferences, or shopping cart status. The most widely used technology for this purpose are cookies, and in recent years, they have grown increasingly problematic due to their misuse, such as for advertising [2]. Since cookies are stored locally (on the user's computer), user information leakage or tampering can be accomplished easily [3]. This

[1] Corresponding Author: Marko Hölbl, Faculty of Electrical Engineering and Computer Science, University of Maribor, Koroška cesta 46, 2000 Maribor, Slovenia; E-mail: marko.holbl@um.si

resulted in a growing mistrust of cookies. Many browser add-ons were developed to address the issue by disabling or deleting cookies. Additionally, private or incognito browsing modes gained popularity. Given the negative connotation of cookies and techniques for their prevention [4], browser fingerprinting has emerged as a new standard in user tracking. Additionally, in the EU, websites need to issue so-called cookie notifications [5], which can impact the user experience of websites when using cookies [6].

A browser fingerprint is a compilation of information about a user device's hardware, operating system, browser, and configuration. It is the process of collecting data using a web browser to generate a device's (potentially unique) identifier (i.e., fingerprint). A server can collect various data from different available APIs (Application Programming Interfaces) and HTTP metadata interfaces using a simple browser-based script. An API, the interface that provides access to specific objects and methods, even enables access to hardware, such as the microphone and camera. However, it requires authorization to do so. Each browser features many such APIs, which are easily accessible via JavaScript, making information collection effortless. Unlike other identification methods, such as cookies, which rely on a unique identifier (ID) explicitly recorded in the browser, browser fingerprinting is less explicit and more concealed.

More information about the client's software and hardware are required to adapt to a wider variety of devices. These unique details, such as the browser's User-Agent, can be gathered from several sources, such as the HTTP message header, the user's IP address, and the screen resolution. Some examples of data that a website can acquire are shown in Table 1.

Table 1. Sample of Data Acquired by a Web Browser [7,8].

Characteristic	**Value**
User agent	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Content encoding	gzip, deflate, br
Content language	en-US,en,sl
List of plugins	Plugin 0: PDF Viewer; Portable Document Format; internal-pdf-viewer. Plugin 1: Chrome PDF Viewer; Portable Document Format; internal-pdf-viewer. Plugin 2: Chromium PDF Viewer; Portable Document Format; internal-pdf-viewer. Plugin 3: Microsoft Edge PDF Viewer; Portable Document Format; internal-pdf-viewer. Plugin 4: WebKit built-in PDF; Portable Document Format; internal-pdf-viewer.
Cookies enabled	yes
Use of local storage	yes
Use of session storage	yes
Timezone	UTC+02:00 Europe/Paris
Screen resolution and color depth	1512x982x30
Platform	MacIntel
Do Not Track	yes
Canvas	Chi fordhank glyphs vext quiz Cwm fjordbank glyphs vext quiz
WebGL Vendor	Google Inc. (Apple)
WebGL Renderer	ANGLE (Apple, Apple M1 Pro, OpenGL 4.1)

This chapter aims to give an overview of existing work and, in this way, provide an entry point into the field and, secondly, lay the groundwork for future research in the field by identifying current challenges.

This chapter's structure is as follows. This section introduced browser fingerprinting, and related definitions and contributions were described. A discussion of existing research in the field of browser fingerprinting is given in Section 2. Section 3 provides a summary of defense mechanisms to tackle browser fingerprinting. Later, a discussion and open research challenges are discussed in Section 4. Section 5 gives the conclusions.

2. Overview of Browser Fingerprint Research

Before 2010, cookie technology was associated with browser uniqueness. Cookies maintain the user status (the so-called session) and can return this data if needed. Cookies store client data, so it is a challenge to assure privacy. Many browser users disable cookies with plug-ins, and current browsers include privacy options that disable cookies.

Mayer [9] studied Internet anonymity in 2009. He showed in a tiny experiment that browser fingerprints may identify users, although Eckersley [10] of the Electronic Frontier Foundation first demonstrated a practical implementation of the idea in 2010. When visiting a web page, the web server can embed JavaScript code or gather information about the user's browsing device. As opposed to cookies, browser fingerprints cannot be disabled. A cookie a user can delete or deactivate using adequate privacy options. Browser fingerprints may be used for cross-domain identification.

Due to the great attractiveness of user tracking with the help of browser fingerprinting, the field is very active, with much research in the field. Mowery and Shacham [11] investigated HTML Canvas fingerprint characteristics. Faiz Khademi et al. [12] examined browser fingerprint detection and protection. Vastel et al. [13] examined browser fingerprints across time.

Browser fingerprint-related research can be categorized according to several study directions, including feature acquisition or defense mechanisms, both of which are addressed in this chapter.

Since browser fingerprinting seeks to identify the user, researchers focus on high-entropy, long-lasting, and preferably cross-browser fingerprint approaches. Modern browsers have strong functionality and extensive interfaces, giving many possible ways in which to create browser fingerprints.

One of the more widely used techniques for acquiring browser fingerprints is using JavaScript code. In this way, browser information such as operating system or browser version can be gained. Much research has utilized this approach, e.g., [3,4,9,10]. For example, Mowery et al. used a plug-in known as NoScript and its whitelist for the characteristics of a fingerprint [14]. Mulazzani et al. [15] have optimized the techniques to enable JavaScript engine detection to leverage the JavaScript parsing engine's properties and, in this way, fingerprint a browser.

Many browser plug-ins block JavaScript scripts because it is too powerful and thus can be abused. In 2013, Unger et al. used CSS (Cascading Style sheet) for fingerprints [16], while in 2015, Takei et al. used browser CSS features for fingerprint collection [17]. Different browser rendering engines read CSS differently; hence, attribute implementation states vary. Browser fingerprints are created by exploiting Web browser request differences. In 2021, Laperdrix et al. [18] suggested infusing style sheets for fingerprint traits. It uniquely identifies browser extensions from the visited website.

As modern browsers with HTML5 support have many capabilities, they pose a risk, as shown in [11]. In this work, text and WebGL scenes were used to create fingerprints. Compared to others, the homogeneity and high entropy make it very utilizable. Later, Acar et al. [19] advanced this approach.

As discussed in [10], WebGL properties can be used to demonstrate how different hardware renders WebGL. However, in 2015, Nakibly et al. [20] suggested fingerprinting the device by detecting the CUP and GPU clock variations during a difficult rendering workload. The WEBGL_debug_renderer_info interface provides the precise device model, according to Laperdrix et al. [21]. Google's Bursztein et al. [22] created a browser fingerprint mechanism utilizing JavaScript and Canvas in 2016. Cao et al. enhanced WebGL hardware fingerprint detection in 2017 [23]. They uniquely identified over 99% of test devices via 31 rendering jobs. Schwarz et al. [24] used numerous JavaScript functionalities not described in MDN docs in 2019.

In 2016, Englehardt et al. developed a Web Audio API-based fingerprint [25] similar to WebGL. Oscillator Node, an audio script, generates the unique audio fingerprint in this study [26]. Many browsers were tested, as well as many hardware and software combinations to get fingerprint data. However, it turned out that Web Audio API alone is unreliable.

Browser plugins add convenience and additional functionality to browsing. Sjosten et al. [27] suggested using Web Accessible Resources to identify browser plug-in installations in 2017. Chrome and Firefox need web page extension resources, and the URL "extension:///" lets you check if the plug-in exists. In this way, most plug-ins can be detected. However, specific extensions do not have this property available. Starov et al. [28] used several approaches to identify browser plug-ins. Namely, many plug-ins alter web page DOMs, and detecting relevant modifications reveals relevant users' plug-in installations and consequentially exposes a user. Sanchez-Rola et al. [29] presented an attack for access control to identify browser plug-ins using time side channels. In 2019, Starov et al. [30] upgraded past browser plug-ins' side effects studies, including injecting script or style tags, empty placeholders, or page messages.

Fuhl et al. [31] correlated the mouse movement trajectory to the human eye, which could be used as a fingerprint. However, these techniques need further validation and research. Abgral et al. utilized cross-site scripting attacks [32] to fingerprint HTML parsers in different browsers. This method yields fingerprints that are hard to mislead and difficult to reproduce since they presume a running HTML parser. Fifield and Egelman [33] suggested measuring font glyph screen sizes to recognize web browser fingerprints in 2015. The authors mainly utilize the rendering of browsers for identification. In a test of over 1,000 browsers, 34% could be identified in this way. Authors in [34] examined HTML5's misuse of the battery API to utilize the properties of short-term batteries to identify users. Sanchez-Rola et al. [29] introduced time-based device fingerprint recognition in 2018, which measures execution clock difference using JavaScript codes to identify users. Wu et al. [35] suggested a website user delay fingerprint in 2021. After IP address translation, users may switch browsers and use virtual machines with 80% recognition.

Based on the overview of current research, the following challenges in browser fingerprint can be highlighted: (1) Most techniques depend on JavaScript, which is an omnipresent and vital part of most of today's web pages. Nevertheless, research in the field of browser fingerprinting should try to develop non-JavaScript-dependent techniques. Some examples include research by Takei et al. [17] and Wu et al. [35]. (2) Research in the field of cross-browser fingerprinting should address aspects like matching recognition featuring weighting and techniques for obtaining more reliable and high-entropy fingerprint characteristics. There is already some research conducted in this direction [4,16]. (3) From the research overview, we can see that most of the fingerprinting properties depend on a device's software (e.g., plug-ins) or hardware properties that can be gathered through the web browser. The challenge here is to fingerprint co-used devices (e.g., in public places, using the same networks). Research is already ongoing in this direction. Fuhl et al. showed how to exploit user activity to create fingerprints and proved its practicality [31].

A brief overview of different approaches to browser fingerprinting is presented in Table 2.

Table 2. Categorization of Research on Brower Fingerprinting Techniques.

Technique is based on	Example Reference
JavaScript	[3,4,9,10,14,15]
CSS	[16–18]
Hardware	[20,21,23,29,34]
HTML5 features	[11,19–26]
Plug-ins / Extensions	[27,28,30,36]

3. Overview of Browser Fingerprint Defense Research

Browser fingerprints, especially those acquired without the user's knowledge, pose a major threat to privacy. Browser fingerprints are best used to precisely monitor and secure users when they don't wish to be tracked. Scholars explore browser fingerprint defense to provide a secure and effective way for users who want to remain concealed.

Browser fingerprint protection research increased after Eckersley et al.'s [10] study highlighted the browser tracking potential. However, there are examples of browser plug-ins or add-ons that further facilitate fingerprinting, like Firegloves [37]. This plug-in returns random results when data on browser properties is gathered, which makes identifying such users simpler. On the other hand, tools like FP-Block [38] generate site-specific fingerprints without affecting continuous or cross-domain tracking. Additionally, authors in [12] proposed to monitor web objects running on the user's browser to check for the intention of fingerprinting. Additionally, they employ protection techniques using randomization, filtering, and even blacklists of relevant websites.

In 2014, Besson et al. noted that randomization is not difficult, but how to randomize is. This work models trackers and fingerprint recognition tools using information theory channels and presents a randomization approach to assure program privacy without fingerprints. Nikiforakis et al. [39] proposed a randomization approach where developers can balance effectiveness and usability using different randomization algorithms. Laperdrix et al. [40] use software variety and dynamic reconfiguration to automatically construct varied browsers for the randomized return of phony fingerprints. Since a virtual machine environment is needed for the implementation, this can significantly impact efficiency. Another study by Trickel et al. [41] created CloakX to hide browser plug-in fingerprints by randomizing the accessible resource path. The technique uses JavaScript code rewriting and the DOM proxy Droxy to intercept and rewrite extension requests, thus assuring protection during browser plug-in installation.

Another direction of browser fingerprinting defense was proposed by Wu et al. [42], namely unification. In their work, the authors suggested unifying WebGL and proposed an approach called UNIGL. Additionally, Fiore et al.'s [43] proposed a concept in which fake data (used for fingerprinting) are generated to cope with browser fingerprinting. However, it needs to be changed continuously to protect regardless of whether genuine and false fingerprint tracking would be possible.

An interesting technique was proposed by Yokoyama and Uda [44]. The authors employ local agents to modify the browser fingerprint value and, in this way, prevent fingerprinting. Another approach was proposed in [45], where Chromium was changed to protect against Flash and Canvas browser fingerprinting, but without influencing the two technologies. Laperdrix et al. [46] also offered a Firefox-based upgrade with fingerprint protection against AudioContext, and Mitropoulos et al. presented a training technique [47] for known cross-site scripting attacks to gather browser fingerprints [32]. ElBanna and Abdelbaki later created a method to reduce browser fingerprinting [48] for WebGL and Canvas fingerprint monitoring.

Based on the overview of current research on browser fingerprint protection, it is evident that this can be done using additional plug-ins or modified versions of browsers. Still, the main remaining challenges include: (1) It is difficult for a user (browser) to determine whether the website's intention is legitimate or malicious. For instance, it is unclear to the user whether or not their screen resolution is being considered when designing the site's layout. For example, it is difficult to determine if retrieving the screen resolution information is to adapt the web page layout or for browser fingerprinting purposes. (2) The use of unification with a small number of users is questionable. It requires the support of vendors, international standards organizations, and technical committees to, for example, unify WebGL and Canvas rendering.

4. Discussion and Open Challenges

The development of browser fingerprinting technology is consistent with the growing concern for privacy among individuals. Traditional tracking using cookies has shown shortcomings, as cookies can be stolen [49], modified or forged, and even injected [50]. Google recently announced that they plan to ban third-party cookies as more and more users block cookies or install protection plug-ins. If this happens, browser fingerprinting will become more important to assure statefulness and legitimate user tracking.

Based on the review of existing research, we anticipate the following directions for future research:

(1) **Machine learning and AI** will play an important role. One of the directions will be algorithms automatically matching fingerprints, as presented in [16,51]. Considering the evolution of fingerprinting techniques and approaches [10,13], a matching algorithm is required. Efficient rule-based matching algorithms were already developed [10,13,52,53]. With further progress in machine and deep learning, this technology is becoming preferable when developing browser fingerprinting techniques. For instance, [9,12] present a clustering algorithm to extract fingerprint signs autonomously. Additionally, machine learning algorithms, such as neural networks, are becoming increasingly popular [11,13] for fingerprinting. It is anticipated that in future research, combining browser fingerprinting and machine learning will increase.

(2) **Browser fingerprinting applications** [54–57] exploit two aspects: the immutability of browser fingerprints and the use of browser fingerprints – gathering them through browser feature collection. However, with research in the field of browser fingerprinting, hardware fingerprinting, and the evolution of browser fingerprinting

[58–61], additional potential applications are emerging (e.g. cross-browser fingerprinting or cross-domain tracking).

(3) As browsers and network technologies continuously develop, many technologies will disappear or be discontinued. For example, Microsoft, Google, and Adobe have discontinued technical support for Flash. Therefore, new approaches that are **less dependent on specific technology** need to be developed.

Despite the fact that browser fingerprinting has been around for a significant amount of time and its maturity, legislation, regulation, and technical specifications have fallen behind practice [62]. Regarding information leakage, previous studies [63] have focused more on the technical aspect of securing device information. Research in [36] demonstrates that browser fingerprinting technology can impact personal privacy. Vendors are continuously monitoring the progress in the field and upgrading their products to prevent the acquisition of specific features that could help with browser fingerprinting. However, in the long term, the fundamental remedy still lies in regulation, legislation, and governance to guide technology development.

5. Conclusions

Current research on browser fingerprinting has yielded significant results that can be used for tracking users. There are two sides to the coin – on the one hand, browser fingerprinting can be used instead of cookies for maintaining the state of a user and, on the other, misused for tracking. The combination of browser fingerprinting and traditional user identity tracking can be applied positively, like identity tracking, user authentication, and for security. In this chapter, we have given an overview of browser fingerprinting from two aspects – acquiring and protecting against it. Further, we have discussed various challenges and future directions of the research field, which is intended to help facilitate further research in this interesting and, for online privacy, very important field.

Acknowledgement

The authors acknowledge the financial support from the Slovenian Research and Innovation Agency (Research Core funding No. P2-0057, the bilateral project BI-US/22-24-147) and the financial support of the ATHENA (Advanced Technology Higher Education Network Alliance) European University project, funded by the European Union, Erasmus+, European universities initiative, grant agreement number 101004096.

References

1. Wikipedia. Device fingerprint [Internet]. 2023 [cited 2023 Jul 6]. Available from: https://en.wikipedia.org/wiki/Device_fingerprint
2. Mathews-Hunt K. CookieConsumer: Tracking online behavioural advertising in Australia. Comput Law Secur Rep. 2016;32(1):55–90.

3. Kwon H, Nam H, Lee S, Hahn C, Hur J. (In-)security of cookies in HTTPS: cookie theft by removing cookie flags. IEEE Transactions on Information Forensics and Security. 2020;15.
4. Cranor LF. Cookie monster. Commun ACM. 2022;65(7):30–2.
5. GDPR.eu. Cookies, the GDPR, and the ePrivacy Directive [Internet]. 2023 [cited 2023 Jul 7]. Available from: https://gdpr.eu/cookies/
6. Kulyk O, Hilt A, Gerber N, Volkamer M. this website uses cookies": Users' perceptions and reactions to the cookie disclaimer. In: European Workshop on Usable Security (EuroUSEC). 2018.
7. Electronic Frontier Foundation (EFF). Cover Your Tracks [Internet]. 2023 [cited 2023 Jul 6]. Available from: https://coveryourtracks.eff.org/
8. AmIUnique.org. Am I Unique? [Internet]. 2023 [cited 2023 Jul 6]. Available from: https://amiunique.org/
9. Mayer JR. Any person… a pamphleteer: Internet Anonymity in the Age of Web 2.0. 2009.
10. Eckersley P. How unique is your web browser? In: Privacy Enhancing Technologies. Berlin, Heidelberg: Springer Berlin Heidelberg; 2010. p. 1–18.
11. Mowery K, Shacham H. Pixel Perfect: Fingerprinting canvas in HTML5. In: Proceedings of W2SP. San Francisco, CA, USA; 2012. p. 1–12.
12. Khademi A, Zulkernine M, Weldemariam K. FPGuard: detection and prevention of browser fingerprint- ing. In: DBSec 2015: Data and Applications Security and Privacy XXIX. Cham: Springer; 2015. p. 293–308.
13. Vastel A, Laperdrix P, Rudametkin W, Rouvoy R. FP-STALKER: Tracking Browser Fingerprint Evolutions. In: 2018 IEEE Symposium on Security and Privacy (SP). IEEE; 2018.
14. Mowery K, Bogenreif D, Yilek S, Shacham H. Fingerprinting information in JavaScript implementations. In: Proceedings of W2SP. Krakow, Poland; 2011.
15. Mulazzani M, Reschl P, Huber M. Fast and reliable browser identification with javascript engine fingerprinting. In: Workshop on Security and Privacy (W2SP). Citeseer; 2013.
16. Unger T, Mulazzani M, Fruhwirt D, Huber M, Schrittwieser S, Weippl E. SHPF: Enhancing HTTP(S) session security with browser fingerprinting. In: 2013 International Conference on Availability, Reliability and Security. IEEE; 2013.
17. Takei N, Saito T, Takasu K, Yamada T. Web browser fingerprinting using only cascading style sheets. In: 2015 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA). IEEE; 2015.
18. Laperdrix P, Starov O, Chen Q, Kapravelos A, Nikiforakis N. Fingerprinting in Style: Detecting Browser Extensions via Injected Style Sheets. In: 30th USENIX Security Symposium (USENIX Security 21) [Internet]. USENIX Association; 2021. p. 2507–24. Available from: https://www.usenix.org/conference/usenixsecurity21/presentation/laperdrix
19. Acar G, Eubank C, Englehardt S, Juarez M, Narayanan A, Diaz C. The web never forgets: Persistent tracking mechanisms in the wild. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. Vienna, Austria; 2014. p. 674–89.
20. Nakibly G, Shelef G, Yudilevich S. Hardware Fingerprinting Using HTML5. CoRR [Internet]. 2015;abs/1503.01408. Available from: http://arxiv.org/abs/1503.01408

21. Laperdrix P, Rudametkin W, Baudry B. Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. In: 2016 IEEE Symposium on Security and Privacy (SP). IEEE; 2016.
22. Bursztein E, Malyshev A, Pietraszek T, Thomas K. Picasso: lightweight device class fingerprinting for web clients. In: Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices. Vienna, Austria; 2016.
23. Cao Y, Li S, Wijmans E. (Cross-) browser fingerprinting via os and hardware level features. In: 24th Annual Network and Distributed System Security Symposium. Scottsdale, Arizona, USA; 2017.
24. Schwarz M, Lackner F, Gruss D. JavaScript template attacks: Automatically inferring host information for targeted exploits. In: Proceedings 2019 Network and Distributed System Security Symposium. Reston, VA: Internet Society; 2019.
25. Englehardt S, Narayanan A. Online tracking: a 1- million-site measurement and analysis. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. Vienna, Austria; 2016.
26. Queiroz JS, Feitosa EL. A Web Browser Fingerprinting method based on the Web Audio API. Comput J. 2019;62(8):1106–20.
27. Sjösten A, Van Acker S, Sabelfeld A. Discovering browser extensions via web accessible resources. In: Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy. New York, NY, USA: ACM; 2017.
28. Starov O, Nikiforakis N. XHOUND: Quantifying the fingerprintability of browser extensions. In: 2017 IEEE Symposium on Security and Privacy (SP). IEEE; 2017.
29. Sanchez-Rola I, Santos I, Balzarotti D. Clock around the clock: time-based device fingerprinting. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. Toronto, Canada; 2018.
30. Starov O, Laperdrix P, Kapravelos A, Nikiforakis N. Unnecessarily Identifiable: Quantifying the fingerprintability of browser extensions due to bloat. In: The World Wide Web Conference. New York, NY, USA: ACM; 2019.
31. Fuhl W, Sanamrad N, Kasneci E. The Gaze and Mouse Signal as additional Source for User Fingerprints in Browser Applications. CoRR [Internet]. 2021;abs/2101.03793. Available from: https://arxiv.org/abs/2101.03793
32. Abgrall E, Traon Y Le, Monperrus M, Gombault S, Heiderich M, Ribault A. XSS-FP: Browser Fingerprinting using HTML Parser Quirks. CoRR [Internet]. 2012;abs/1211.4812. Available from: http://arxiv.org/abs/1211.4812
33. Fifield D, Egelman S. Fingerprinting web users through font metrics. In: Financial Cryptography and Data Security. Berlin, Heidelberg: Springer Berlin Heidelberg; 2015. p. 107–24.
34. Olejnik Ł, Acar G, Castelluccia C, Diaz C. The leaking battery - a privacy analysis of the HTML5 Battery Status API. In: Data Privacy Management, and Security Assurance. Cham: Springer International Publishing; 2016. p. 254–63.
35. Wu T, Song Y, Zhang F, Gao S, Chen B. My Site Knows Where You Are: A Novel Browser Fingerprint to Track User Position. In: ICC 2021 - IEEE International Conference on Communications. 2021. p. 1–6.
36. Karami S, Ilia P, Solomos K, Polakis J. Carnus: Exploring the privacy threats of browser extension fingerprinting. In: Proceedings 2020 Network and Distributed System Security Symposium. Reston, VA: Internet Society; 2020.

37. Boda K, Földes ÁM, Gy. Gulyás–portal. eu G. Cross-browser fingerprinting test 2.0.
38. Ferreira Torres C, Jonker HL, Mauw S (Sjouke). FP-Block: Usable Web Privacy by Controlling Browser Fingerprinting. In: Pernul G, Ryan PYA, Weippl E, editors. Computer Security – ESORICS 2015. Switzerland: Springer Nature Switzerland AG; 2015. p. 3–19. (Lecture Notes in Computer Science (LNCS) series; vol. 2).
39. Nikiforakis N, Joosen W, Livshits B. PriVaricator: deceiving fingerprinters with little white lies. In: Proceedings of the 24th International Conference on World Wide Web. Florence, Italy; 2015.
40. Laperdrix P, Rudametkin W, Baudry B. Mitigating Browser Fingerprint Tracking: Multi-level Reconfiguration and Diversification. In: 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. 2015. p. 98–108.
41. Trickel E, Starov O, Kapravelos A, Nikiforakis N, Doupé A. Everyone is Different: Client-side Diversification for Defending Against Extension Fingerprinting. In: 28th USENIX Security Symposium (USENIX Security 19) [Internet]. Santa Clara, CA: USENIX Association; 2019. p. 1679–96. Available from: https://www.usenix.org/conference/usenixsecurity19/presentation/trickel
42. Wu S, Li S, Cao Y, Wang N. Rendered Private: Making GLSL Execution Uniform to Prevent WebGL-based Browser Fingerprinting. In: 28th USENIX Security Symposium (USENIX Security 19) [Internet]. Santa Clara, CA: USENIX Association; 2019. p. 1645–60. Available from: https://www.usenix.org/conference/usenixsecurity19/presentation/wu
43. Fiore U, Castiglione A, Santis Alfredo D, Palmieri F. Countering browser fingerprinting techniques: Constructing a fake profile with Google chrome. In: 2014 17th International Conference on Network-Based Information Systems. IEEE; 2014.
44. Yokoyama S, Uda R. A proposal of preventive measure of pursuit using a browser fingerprint. In: Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication. New York, NY, USA: ACM; 2015.
45. Baumann P, Katzenbeisser S, Stopczynski M, Tews E. Disguised Chromium browser: Robust browser, Flash and Canvas fingerprinting protection. In: Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society. Sofia, Bulgaria; 2016.
46. Laperdrix P, Baudry B, Mishra V. FPRandom: Randomizing core browser objects to break advanced device fingerprinting techniques. In: Lecture Notes in Computer Science. Cham: Springer International Publishing; 2017. p. 97–114.
47. Mitropoulos D, Stroggylos K, Spinellis D, Keromytis AD. How to train your browser: preventing XSS attacks using contextual script fingerprints. ACM Transactions on Privacy and Security. 2016;19(1):1–31.
48. ElBanna A, Abdelbaki N. NONYM!ZER: Mitigation framework for browser fingerprinting. In: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C). IEEE; 2019.
49. Sivakorn S, Polakis I, Keromytis AD. The cracked cookie jar: HTTP cookie hijacking and the exposure of private information. In: 2016 IEEE Symposium on Security and Privacy (SP). IEEE; 2016.

50. Calibri F, Chen H, Duan X, Zheng J, Jiang J. Path leaks of HTTPS Side-Channel by cookie injection. In: International Workshop on Constructive Side-Channel Analysis and Secure Design. Springer; 2018. p. 189–203.
51. Laperdrix P, Avoine G, Baudry B, Nikiforakis N. Morellian analysis for browsers: Making web authentication stronger with canvas fingerprinting. In: Detection of Intrusions and Malware, and Vulnerability Assessment. Cham: Springer International Publishing; 2019. p. 43–66.
52. Liu X, Liu Q, Wang X, Jia Z. Fingerprinting web browser for tracing anonymous web attackers. In: 2016 IEEE First International Conference on Data Science in Cyberspace (DSC). IEEE; 2016.
53. Liangfeng Z, Yi W, Yuanyi W, Rui K. Statistics-based browser fingerprinting technology. Information Network Security. 2019;11:49–55.
54. Nikiforakis N, Kapravelos A, Joosen W, Kruegel C, Piessens F, Vigna G. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In: 2013 IEEE Symposium on Security and Privacy. IEEE; 2013.
55. Rochet F, Efthymiadis K, Koeune F, Pereira O. SWAT: Seamless web authentication technology. In: The World Wide Web Conference. New York, NY, USA: ACM; 2019.
56. Jia Z, Cui X, Liu Q, Wang X, Liu C. Micro-honeypot: Using browser fingerprinting to track attackers. In: 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC). IEEE; 2018.
57. Qingxuan X. Fake order recognition system based on browser fingerprint. Electronic Production. 2019;2.
58. Li X, Cui X, Shi L, Liu C, Wang X. Constructing browser fingerprint tracking chain based on LSTM model. In: 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC). IEEE; 2018.
59. Bird S, Mishra V, Englehardt S, Willoughby R, Zeber D, Rudametkin W, et al. Actions speak louder than words: Semi-supervised learning for browser fingerprinting detection. CoRR [Internet]. 2020;abs/2003.04463. Available from: https://arxiv.org/abs/2003.04463
60. Qixu L, Xinyu L, Cheng L, Junnan W, Langping C, Jiaxi L. An android browser fingerprint recognition method based on bidirectional recurrent neural network. Computer Research and Development. 2020;57(11):2294–311.
61. Muñoz-Garcia Ó, Monterrubio-Martin J, Garcia-Aubert D. Detecting browser fingerprint evolution for identifying unique users. International Journal of Electronic Business. 2012;10(2):120–41.
62. Laperdrix P, Bielova N, Baudry B, Avoine G. Browser Fingerprinting: A Survey. ACM Trans Web [Internet]. 2020 Apr;14(2). Available from: https://doi.org/10.1145/3386040
63. Jérôme Segura. Operation Fingerprint: A Look Into Several Angler Exploit Kit Malvertising Campaigns [Internet]. MalwareBytes. 2016 [cited 2023 Jul 2]. Available from: https://www.malwarebytes.com/blog/news/2016/03/ofp