Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 3363335, 14 pages
https://doi.org/10.1155/2022/3363335

Review Article

A Survey of Browser Fingerprint Research and Application

Desheng Zhang[1], **Jianhui Zhang**[2], **Youjun Bu**[2,3], **Bo Chen**[2], **Chongxin Sun**[2,3] and **Tianyu Wang**[1]

[1]*School of Cyber Science and Engineering, Zhengzhou University, Zhengzhou 450000, China*
[2]*Information Technology Institute, PLA Strategic Support Force Information Engineering University, Zhengzhou 450000, China*
[3]*Purple Mountain Laboratories, Nanjing 210000, China*

Correspondence should be addressed to Youjun Bu; buyoujun@pmlabs.com.cn

Received 13 July 2022; Revised 28 October 2022; Accepted 29 October 2022; Published 9 November 2022

Academic Editor: Pierre-Martin Tardif

With the development of modern browsing, the convenience brought by rich browser features has also produced a large number of features, which are called browser fingerprints. This article surveys the latest research results on browser fingerprinting, hoping to provide a convenient navigation for newcomers to research or apply this technology in the future. This paper first briefly introduces the browser fingerprinting technology itself, then classifies the related research on browsers, and analyzes the development of different research directions of browser fingerprinting in detail. And through the analysis of the existing results, the problems faced by different research directions are pointed out. After that, this paper introduces the application of browser fingerprint technology in detail and discusses the application achievements and technical challenges of this technology. Next, this paper introduces the theoretical tools related to the research of browser fingerprinting technology and introduces the application of different theoretical tools and practical significance. Finally, the research achievements of browser fingerprint recognition are summarized, and the future development trend is pointed out.

1. Introduction

Websites often need to keep track of visitors for a variety of reasons, such as maintaining log-in status or implementing shopping cart functionality. The traditional user identity tracking method is based on cookies, but cookie technology has exposed more and more problems in recent years. For example, many websites abuse cookies for advertising [1]. And because cookies are stored in the local end of users, it is easy to lead to user information leakage and tampering [2]. These problems have led to a growing distrust of cookies, with many people installing add-ons to clear cookies or simply using private mode. While cookie technology is becoming more and more inefficient [3], browser fingerprint technology has gradually developed and become a new mainstream user tracking technology.

Browser fingerprinting technology is an associated product of web technology, and its historical premise lies in the rapid development of web technology. In the ancient era of static web pages, in order to distinguish the Nexus browser and the Mosaic browser, the developers proposed UserAgent, which was initially used to indicate what browser, operating system, and respective version numbers the user was using. Then, with the development of JavaScript, CSS (Cascading Style Sheets), and back-end languages, the functions of web pages have become richer and more interactive, but the rich functions bring the possibility of exposing more user information. After all, personalized services need to be provided with personalized information. Then, with the emergence of mobile devices such as mobile phones and tablets, history has brought technologies such as HTML5 and CSS3. The website can even directly call hardware interfaces such as graphics cards. Diversified device adaptation also requires more software and hardware information for the web. The ways in which these characteristic information are obtained are different, such as information about the browser User-Agent field, by obtaining the HTTP message header, and the IP address is requested by the user through

the header of the IP packet is obtained, the screen resolution. Such information requires JavaScript to obtain and so on. Finally, a browser feature vector containing multiple features is obtained. For example, Table 1 is a sample of browser fingerprints.

The wheel of history is rolling forward, and the emergence of browser fingerprint technology is a historical necessity. When users obtain personalized, rich, and dynamic services, they must expose more characteristic information. This is not a loophole or a backdoor but a functional trade-off.

1.1. Definitions. The browser fingerprint is a collection of all feature information that can be collected through the browser, but the feature information does not include the data that the user actively fills in and submits. Browser fingerprints and fingerprint similar to humans rely on the uniqueness of fingerprints browser browsing device itself, and it does not change with changes in the environment. Therefore, even in the case of encrypted network, anonymous network, mobile network, and even crossdomain, it can be identified. Modern browsers are very complex, each component has different characteristics, and these characteristics combined to form a unique fingerprint of the browser.

1.2. Organization. The main structure of this paper is as follows. Section 1 introduces the background of browser fingerprinting, related definitions, and contributions of this paper. Section 2 briefly introduces browser fingerprinting, and then according to different research directions, it introduces the existing research results and research challenges. Section 3 summarizes the existing application achievements and the application problems faced according to different application scenarios. Section 4 enumerates the mathematical tools and methods in browser fingerprint research, and points out its practical significance. In Section 5, the general situation of browser fingerprinting technology is discussed.

2. Related Research

2.1. Origin. Before 2010, if the unique identity of a browser was mentioned, people would think of cookie technology. The cookie technology is to store the user's unique identity in the browser locally and return the cookie when a request is made. While this solved some problems and is still an important part of browser fingerprinting today, it was inevitable that cookie technology would decline. Cookie technology is the data stored on the client; after all, the security and availability cannot be guaranteed. To protect their privacy, many users add plug-ins that prohibit the use of cookies when using browsers, and modern browsers also provide privacy modes to invalidate cookies.

In 2009, Mayer [4] published a study on Internet anonymity. In a small sample experiment, he pointed out that users can be identified by collecting characteristic information of browsers, but the concept of browser fingerprints was first proposed in 2010 by Eckersley [5] of the Electronic Frontier Foundation. It takes advantage of the various features offered by modern browsers. When a user requests a web page, the Web server obtains and sends back some unique information about the user's browsing device by embedding JS code or another way. The information includes the de browser version, whether cookie is enabled, screen resolution, browser plug-in, system font, time zone, and so on. It can identify unique user entities. Browser fingerprint technology is stateless; that is, the use of browser fingerprints does not require any information to be stored on the client-side, and naturally, users cannot invalidate browser fingerprints by disabling cookies or privacy modes. In addition, browser fingerprints have high information entropy and are easy to obtain, and users will generate the same fingerprint for multiple visits and can be used for crossdomain identification and other excellent features.

In the ensuing time, scholars continued to conduct research on browser fingerprinting technology, trying to further tap the potential of this technology. For example, Mowery and Shacham [6] in 2012 explored the fingerprint features brought by Canvas in HTML. In 2015, FaizKhademi et al. [7] studied the detection and defense of browser fingerprints. In 2018, Vastel et al. [8]studied the long-term evolution of browser fingerprints. According to different research directions, we have classified and summarized the existing browser fingerprint-related research, which can be roughly divided into three aspects: feature acquisition research, fingerprint defense research, and fingerprint evolution research. These three directions are discussed separately in the following sections.

2.2. Feature Acquisition Research. The ultimate goal of browser fingerprinting is to track the unique user entity. Therefore, obtaining high-entropy, long-lasting, and preferably crossbrowser-related fingerprint features is the main research direction for scholars in obtaining browser fingerprints. Due to the powerful functions and rich interfaces of modern browsers, it also provides many possibilities for researchers to obtain browser fingerprints. At the end of this section, Table 2 is given, which summarizes the characteristics of the various browser fingerprint acquisition methods.

2.2.1. JavaScript-Based Fingerprints. Due to the powerful function of JavaScript, JavaScript code can be easily used to obtain many fingerprint information of the client's browser, such as browser version, operating system, and system architecture. Many researchers use JavaScript technology to obtain browser fingerprint [2–5]. Among them, Mowery et al. also used the NoScript plug in the literature [9] to obtain its whitelist as part of the characteristic fingerprint. Mulazzani et al. [10] later optimized for JavaScript engine recognition, using the different characteristics of the complex JavaScript parsing engine, and reliably identifying a given browser through JavaScript engine fingerprints.

2.2.2. CSS-Based Fingerprints. While JavaScript can be easily accessible for many features, there are negative effects of JavaScript being too powerful, such as many browser plugins that disable all or part of JavaScript scripts. Unger et al. began to use CSS (Cascading Style sheet) as a part of fingerprint in 2013 [11], and Takei et al. made full use of CSS

Table 1: Browser fingerprint sample.

User-Agent	Mozilla/5.0 (Windows NT 10.0; Win64; ×64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4495.0 Safari/537.36
Accept	text/html, application/xhtml+xml, application/xml; $q = 0.9$, image/webp, */*; $q = 0.8$
Accept-language	zh-CN, en-US; $q = 0.5$
IP	104.193.88.123
Platform	Windows
Vendor	Google Inc.
appName	Netscape
Product	Gecko
appVersion	5.0 (Windows NT 10.0; Win64; ×64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4495.0 Safari/537.36
cookieEnabled	True
Logical processors	8
Local time	Mon Jul 05 2021 09 : 57 : 32 GMT+0800 (China Standard Time)
Resolution	1920 × 1080
colorDepth	24
WebGL vendor	Google Inc.
WebGL renderer	ANGLE (Intel(R) HD Graphics 4600 Direct3D11 vs_5_0 ps_5_0)
…	…

Table 2: Summary of browser feature research.

Ways to get fingerprints	Representative work	Features
JavaScript-based fingerprints	[2, 3]	Its essence is to use JS to call various APIs to obtain various information.
	[10]	Use the features of the JS engine as the browser feature to identify users.
CSS-based fingerprints	[11, 13]	Use the parsing feature of CSS as a feature to avoid the problem of users banning JS.
Canvas-based fingerprint	[14]	Canvas fingerprints have the characteristics of consistency, high entropy, and orthogonality with other fingerprints.
Hardware-based fingerprints	[15, 16]	Most of them have formulated different tasks for the hardware to execute and infer the hardware characteristics based on the results.
Fingerprint based on Audio API	[20]	Use the difference in audio processing of different browser devices as a feature.
Plugin-based fingerprint	[22–24]	The browser plug-in will modify the web page and the browser itself, because these are all features. However, the ways of obtaining them are more diversified.
Other browser fingerprint acquisition technologies	[26]	The author's goal is to capture the position of the user's gaze, but since the gaze is difficult to obtain, the replacement of the mouse track is used.
	[27]	The reason is that different browsers parse some HTML differently. The author constructs multiple XSS vectors, and the generated parsing features are returned as fingerprints.
	[30]	The author uses JS code to construct a bunch of computing tasks and combines the system's time API to obtain the running time of the user's browser, which can distinguish user devices by task execution, but this method requires a certain performance cost on the web front end.
	[31]	The author uses network delay as a fingerprint feature. Obviously, this technology cannot cope with network proxy and VPN technology and has strong limitations.

features of browsers in 2015 [12] for fingerprint acquisition. The principle is that when different rendering engines of browsers parse CSS, the implementation states of each attribute are different. Differences occur in Web browser requests, and these differences are exploited to construct unique browser fingerprints. In 2021, Laperdrix et al. [13] proposed a new method to obtain fingerprint features by injecting style sheets. It can uniquely identify browser extensions from the context of the visited page. This is a new way to get plugin features via CSS.

2.2.3. Canvas-Based Fingerprint. Modern browser for HTML5 support offers many powerful features to the user but also left a risk, in order to further exploit the browser

get more unique fingerprint, Mowery and 2012 documents in [6]. By rendering the text and WebGL scene to the <canvas> element, a brand new fingerprint is obtained. The new fingerprint having uniformity, high entropy, orthogonal to the other fingerprints, transparent to the user, and easily accessible is good property. Then, Acar et al. [14] on the canvas on a large scale study and the use of modern literature browsers fallback font mechanism to generate more between devices fingerprint high entropy.

2.2.4. Hardware and Software-Based Fingerprints. The characteristics of WebGL have been mentioned before [5], but it is mainly used to prove the difference of different hardware rendering WebGL. In a 2015 study by Nakibly et al. [15], it was proposed that the device be fingerprinted by measuring the relative clock deviations of the device's CUP and GPU through some complex rendering task. In Laperdrix et al.'s research [16], it is mentioned that the precise device model can be obtained directly through the WEBGL_debug_renderer_info interface. In 2016, Bursztein et al. [17] of Google designed a browser fingerprint protocol by using JS and Canvas, which can accurately distinguish the software and hardware stack of mobile or desktop clients. In 2017, Cao et al. further improved WebGL's recognition of hardware fingerprints in literature [18]. Through a series of 31 rendering tasks, they were able to uniquely identify more than 99% of the test devices in 1,903. In 2019, Schwarz et al. made use of many JavaScript features not mentioned in MDN documents in literature [19]. The system architecture, memory allocation, operating system fingerprint, and to a certain extent can be immune to the browser fingerprint technology.

2.2.5. Fingerprint Based on Audio API. Similar to the above WebGL technology, Englehardt et al. proposed the fingerprint based on Web Audio API in literature [20] in 2016. In this paper, the signal generated by Oscillator Node, a script for processing audio, is used as the unique audio fingerprint. Queiroz et al. went a step further [21]. The author tested various browsers and related hardware and software combinations in detail to obtain detailed fingerprint data, but the author still pointed out that using Web Audio API alone as a fingerprint is not very reliable.

2.2.6. Plugin-Based Fingerprint. Browser plugins bring convenience to users but also bring more characteristic information. In 2017, Sjosten et al. [22] proposed to use Web Accessible Resources to detect whether the specified browser plug-in is installed. Both Chrome and Firefox require extension resources referenced in regular web pages. That is, you can determine whether the specified plug-in exists by accessing the URL in the form "extension:///." While most plugins can already be detected this way, not every extension has this accessible resource; so, there will still be some plug-ins that will not be detected by this technique. In the same year, Starov et al. [23] adopted different methods to detect the installation of browser plug-ins. The principle is that many plug-ins will modify the DOM of web pages. By detecting the relevant modifications, the plug-in installation of relevant users can be learned, and then the unique user can be determined. In the same year, Sanchez-Rola et al. [24] proposed a new time-side channel attack for access control to detect the installation of browser plug-ins. The author claimed that his detection effect was better than all previous extended fingerprint detection methods. In 2019, Starov et al. [25] further advanced the previous research on the side effects of browser plug-ins modifying web elements, including the possibility of injecting empty placeholders, injecting script or style tags, or sending messages on the page. The authors analyzed 58,034 extension stores from Chrome and found that 5.7% of them contained fingerprint able bloat. 61% of these extensions are recognized.

2.2.7. Other Browser Fingerprint Acquisition Technologies. Recently, Fuhl et al. [26] did some very groundbreaking work. The author first did the correlation between the human eye and the mouse movement trajectory and then demonstrated the possibility of using the mouse movement as a fingerprint, but the author only proves this research. There are still many areas to be studied for the possible effectiveness of future applications. In 2012, Abgral et al. used XSS attacks in a different way in the literature [27] to obtain the characteristics of different browsers' HTML parsers as fingerprints. The fingerprints obtained in this way are difficult to be deceived, and it is very difficult to simulate a behavior without running the HTML parser itself. In 2015, Fifield and Egelman [28] proposed a web browser fingerprint recognition technology based on measuring the screen size of font glyphs. The author mainly uses different browsers and fonts to achieve different final rendering effects. The author tested more than 1,000 web browsers. Users finally able to identify 34% of them. In 2016, Egelman et al. [29] analyzed HTML5's abuse of battery API, and short-term battery can be used as a sign of short-term user identification. In 2018, Sanchez-Rola et al. [30] proposed a time-based device fingerprint recognition, which is to measure the execution clock difference through a series of JavaScript codes, and finally achieve the effect of identifying users. In 2021, Wu et al. [31] proposed a characteristic fingerprint based on the time delay of users to websites. For users to switch browsers and use virtual machines, the recognition rate still exceeds 80% after IP transformation.

By summarizing and sorting out the above content, we can roughly see that there are three problems to be solved in browser fingerprint research:

(1) JS dependency problem: Most of the above fingerprint feature acquisition relies on JavaScript technology. The first part of the feature is directly based on JS, Canvas relies on JS drawing, and the fingerprint feature based on audio or hardware either depends on JS to call the operating system API or depends on JS to build rendering tasks. Putting eggs in the same basket is not a good thing; although, in the short-term, there is no possibility that JS will be eliminated like Flash, but user disabling and throttling and normalization of OS API calls are already happening. How to reduce the dependence of JS and develop a feature acquisition method that does

not depend on JS is the first problem faced by the future research of browser fingerprinting. Takei et al. [12] and Wu et al. [31] gave us a good start in this regard, using CSS and network latency to construct fingerprints, respectively. These two technologies still have the potential for further development in both application scenarios and application effects

(2) Crossbrowser fingerprinting issues: a large part of the characteristics of browser fingerprinting is beyond the specific browser, such as screen resolution, operating system, time zone, and IP, which reveal the possibility of crossbrowser fingerprinting applications. Articles [3, 11], respectively, show two challenges in implementing crossbrowser fingerprinting, one is how to reasonably set the weights of different features in matching recognition, and another is how to obtain more stable and higher entropy fingerprint features

(3) Obtain differential fingerprints in a homogeneous environment: from the above introduction, we can intuitively understand that more than 90% of the above fingerprint features are based on the hardware and software information of the device. For homogeneous environments, such as Internet cafes and computer rooms, all computers have the same hardware and software devices and are in the same subnet. Most of the above methods can effectively distinguish them. For this problem, Fuhl et al.'s research [26] is creative, and they use the user's behavioral characteristics to construct fingerprints and demonstrate the feasibility of the method through experiments. It is believed that future scholars can propose more creative solutions to the problem of fingerprint acquisition in a homogeneous environment

2.3. Fingerprint Defense Research. Browser fingerprints are a huge hazard to privacy for identification, especially fingerprint acquisition in most cases without the user noticing it. The use of browser fingerprints is best to be safe and controllable; that is, in addition to accurately tracking users when they need to be identified, they can also be protected when users do not want to expose their browser fingerprints. Scholars' technical research on browser fingerprint protection is dedicated to providing safe and effective protection methods when users want to hide themselves. At the end of this section, Table 3 is presented to compare the advantages and disadvantages of various protection methods.

2.3.1. Browser Protection Plugin. After the publication of Eckersley et al.'s study [5], people became more and more aware of the harm of browser fingerprint to privacy, and more and more researches on browser fingerprint protection began to be carried out. Boda has released a browser plug-in for browser fingerprints called Firegloves [32]. The plug-in returns random values when querying certain properties, but because the same properties can be retrieved through different browser API, users of Firegloves are easier to identify than users who have not installed the extension. Torres et al. developed FP-Block [33] to address the problem of crossdomain tracing of browser fingerprints. It generates different fingerprints for different sites, without affecting continuous tracking and isolating cross-domain tracking. Faiz Khademi et al. proposed [7] to detect whether websites were collecting fingerprints by monitoring web objects running on users' browsers, protect users from fingerprint identification by randomization strategy and two filtering technologies, and put relevant websites into blacklist, but this approach relies on the ability to identify anomalies on the site. Since both of these plug-ins return random values, they have similar problems with Firegloves.

2.3.2. Randomization Method. Besson et al. pointed out in the 2014 literature [34] that the idea of randomization is not a problem, but how to randomize is a problem that should be further studied. In this paper, the author uses information theory channels to model the knowledge of trackers and fingerprint recognition programs and finally proposes a randomization mechanism to ensure the privacy of any program, reducing the need to provide fingerprints. In 2015, Nikiforakis et al. [35] introduced the concept of randomization strategy. Different randomization strategies can cope with different environments, making it convenient for developers and their own needs to balance effectiveness and usability. Laperdrix et al. [36] adopts the idea of mobile target defense for the randomized return of fake fingerprints, using software diversity and dynamic reconfiguration to automatically assemble diverse browsers. However, its specific implementation is through a virtual machine, and the performance consumption cannot be ignored. Aiming at the fingerprint of the browser plug-in, Trickel et al. [37] designed CloakX. The author randomized the network accessible resource path. Through static rewriting of the extended JavaScript code and dynamic DOM proxy Droxy, it instantly intercepts and rewrites extension requests to achieve protection in the user's browser plug-in installation situation.

2.3.3. Uniform Methods. The opposite of randomization is unification. Wu et al. proposed a method of unifying WebGL [38] to combat browser fingerprints. The author analyzed the reasons for the differences in WebGL and proposed a new system called UNIGL to rewrite GLSL. The program is that WebGL presents the same rendering effect to erase the fingerprint of WebGL. Also in 2014, Fiore et al.'s idea [39] is even simpler. The author directly constructs a set of fake fingerprint information to deal with browser fingerprint tracking, but if it cannot be changed reasonably, the goal of tracking the user's identity, regardless of true and false fingerprints, will provide tracking effects.

2.3.4. Other Protective Way. In 2015, Yokoyama and Uda [40] proposed a method of using local agent to rewrite the browser fingerprint value to prevent the third party from pursuing individual users. Its advantage is that for a single user in the LAN, there is no need to install redundant hardware and software locally, but the disadvantage is that there

Table 3: Summary of browser fingerprinting defense.

Methods of defense	Representative work	Advantage	Disadvantage
Browser protection plug-in	[32, 33]	Development and application are convenient	The browser plug-in itself can also become an identified feature
Randomization method	[35–37]	Randomization can disguise as a desired feature while protecting user privacy	Abnormal combinations of random features are quickly identified as false identities
Uniform methods	[38, 39]	Unification facilitates application development and processing	The unification of a single or part of the user still has the recognition value, and the advantages of unification require industry-wide efforts
Other protective way	[40–44]	Hiding browser fingerprints through browsers or modifying network agents is generally better defensive	Users are required to change their browser or modify their network devices

is nothing you can do with a local technique that returns a HASH value after calculating the fingerprint. In Baumann et al.'s literature [41], the author made a revision to protect browser fingerprint directly based on Chromium, so that Flash and Canvas fingerprint recognition can be prevented without disabling Flash and HTML5 canvas functions, and the returned fingerprints are all real collected fingerprints, which are fixed in the whole browsing session. It reduces the possibility of being identified by the site as an abnormal fingerprint. Later, Laperdrix et al. [42] also adopted the same idea and proposed a revision based on Firefox, adding fingerprint protection against AudioContext. As for obtaining browser fingerprints from XSS attacks mentioned above [27], Mitropoulos et al. proposed a training method in the literature [43] to deal with known XSS attacks. Later, ElBanna and Abdelbaki proposed a framework to reduce browser fingerprint [44], which is mainly aimed at fingerprint tracking of WebGL and Canvas.

Although there seems to be a lot of research on browser fingerprint defense, for individual users, there are almost only random ways to protect themselves, that is, using plug-ins or browsers that randomly generate fingerprints. The reason lies in two aspects:

(1) It is almost impossible for the client to determine whether the website's call to feature information is illegal or legal. For example, for a request for screen resolution information, the user cannot determine whether the website is for adapting the web page layout or just for recording user device information

(2) The unification of a small number of users is meaningless: the unification of the interface requires the cooperation of various manufacturers and related technical institutions to achieve, such as formulating a unified WebGL and Canvas rendering effect, which seems to be a very ideal solution, but there is almost nothing individual users can do about it

2.4. Fingerprint Evolution Research. In life, we judge whether a person has touched something by directly comparing his fingerprint with the fingerprint on the object. But with browser fingerprinting, the situation gets more complicated, and leaving aside the issue of fake fingerprints, a single user's browser fingerprint can change even if he uses the same browser on the same device for multiple visits. We call this the evolution of browser fingerprinting. There may be various reasons for this, such as upgrading the browser version, installing certain plug-ins, and using only certain settings, which may cause changes in the browser fingerprint of the same user. At the end of this section, Table 4 compares the browser fingerprint evolution tracking algorithms proposed by different articles.

In the early days of browser fingerprinting concepts, at the time of Eckersley's original article [5], the author stated that 37.4% of users who allowed cookies to visit the site multiple times showed more than one fingerprint over time. But fortunately, the paper also points out that these changes are not random, and a reasonable matching algorithm can be used to continuously track the evolving fingerprint. A simple correlation algorithm is given in this paper, and the feasibility of tracking the evolving fingerprint is proved by experiments.

Munoz-Garcia et al. [45] put forward a clustering algorithm in 2012, which can cluster different fingerprints of the same browser, but its clustering algorithm needs to compare more times, and the amount of extra JavaScript code in the web page to obtain some fingerprint attributes. The interpretability of the clustering algorithm is not particularly strong. The author analyzed disagreement decay of various features in his research on browser fingerprint evolution patterns, and its data and research methods are very instructive. In 2015, Yamada et al. [46] used enhanced editing distance to measure the degree of evolution of browser fingerprint. However, the limitation is that the authors only use this method to measure a list of browser plug-ins and do not extend it to all fingerprint features of the browser. Later, Vastel et al. [8] propose two methods to track browser fingerprint evolution, one is based on rules, and the other is based on the random forest algorithm. It can track browsers for 54.48 days and can track 26% of browsers for more than 100 days. Given that browsers change with time, Li et al. constructed time series based on browser fingerprint samples with different time steps as input vectors in reference

Table 4: Fingerprint evolution tracking algorithm.

Algorithm	Ref.	Feature processing method	The experimental results
Rule-based algorithm	[5]	Eight features such as User-Agent and http-accept are selected.	The algorithm requires some prerequisites, there is a 65% probability that the algorithm will start, and the accuracy rate after startup will be 99.1%.
Clustering algorithm	[45]	Four feature weight distribution methods were selected: (1) average weight, (2) entropy as a weight, (3) disagreement decay as a weight, and (4) consider both entropy and disagreement decay.	It is best in the case of feature processing scheme 4, with an accuracy rate of 99.98%, the precision is 93%, and the recall is 87%.
Algorithm based on feature similarity	[46]	Measured by Levenshtein distance of Pluginlist in the browser, the author selected different thresholds and access intervals for the experiment.	When the threshold is set to 60, the accuracy reaches 97.94%. When the threshold is 53, the accuracy is still 97.57 when the access interval exceeds four weeks
Random forest	[8]	Eight features were selected according to their influence on the classification results. Random forest selects 10 trees and 3 features.	The best ownership of the article is 0.985, which means that for long-term tracking of each browser, only 1.5% match errors.
LSTM	[47]	In this paper, fingerprint features are transformed into one-dimensional vectors, and the latest three fingerprints are input into a group each time.	The best accuracy of the training set is 92.4%, and the best accuracy of the test set is 93.3%.
Bi-RNN	[49]	The authors split the UserAgent attributes and then weighted the sum. The Canvas element is CRC replaced.	The best F1-score of this method reached 99.25%.

[47] and adopted the LSTM (Long Short-Term Memory) algorithm to track fingerprint evolutions, achieving better results than reference [8]. Recently, Bird et al. [48] proposed a semisupervised machine learning method for detecting fingerprint scripts. The method is extended to detecting unknown scripts by candidate scripts that may contain fingerprint recognition. This semisupervised learning method is robust to incomplete and small tag sets. Also in 2020, Qixu et al. [49] adopted Bi-RNN (bidirectional cyclic neural network) and added attention mechanism to deal with long-term fingerprint evolutions. And the author tried to use the gradual web application and browser fingerprint combined to improve the stability of recognition. Li and Cao [50] conducted the first large-scale study on millions of fingerprints. The author not only analyzed the algorithm to track the evolution of fingerprints but also analyzed the reasons for the changes of different features of fingerprints and paid attention to the impact of browser fingerprints on information leakage, such as some fonts will reveal whether the user has Office installed, while the time zone and IP may reveal the user's physical address. The research of Pugliese et al. [51] is a fortune. The author conducted a three-year survey on more than 1,300 users. In addition to studying the feasibility of tracking users by tracking the browser fingerprint evolutions, they also studied users' privacy behavior, their perception of browser fingerprinting, the countermeasures they apply, and the impact of their study.

In general, the research on browser fingerprint evolution still has the following problems:

(1) The long-term fingerprint evolution dataset problem: due to the privacy problem of browser fingerprinting, many researchers refuse to disclose their own datasets, and in the article [8], the author only open-sources some of the data. However, new researchers often lack enough time to collect long-term fingerprint evolution data. This supply-demand conflict looks set to continue in the short-term

(2) Matching algorithm performance issues: in the article [50], it is pointed out that for many matching algorithms, once the data set is expanded to the millions level, the time consumption cannot be ignored. Considering the number of daily visits to large commercial websites, this is an unavoidable problem

(3) Long-term tracking of browser fingerprint evolution: track users' browser fingerprinting for as long as possible, and there is always room for reoptimization

3. Browser Fingerprint Application

3.1. Commercial Advertisement Recommendation. When the browser fingerprint was first proposed [5], it was pointed out that it could be used to track users, and its characteristics were similar to cookies. It was pointed out in the article that the browser fingerprint could be used as a unique identification mark alone or in combination with other marks to uniquely locate users. So, browser fingerprints can generally be used to do all the things that require identifying a user. Its workflow is shown in Figure 1. One of the first and largest applications is that commercial companies use to target users for advertising, price discrimination, and to collect users' physical and financial status and other privacy [52]. In 2013, Nikiforakis et al. conducted a large-scale study on the application of browser fingerprint in business [53]. The author captured as many as 20 pages from each of the top 10,000 Alexa sites for analysis, and the final results showed that the research results showed that fingerprint identification has become a part of some of the most popular Internet sites.

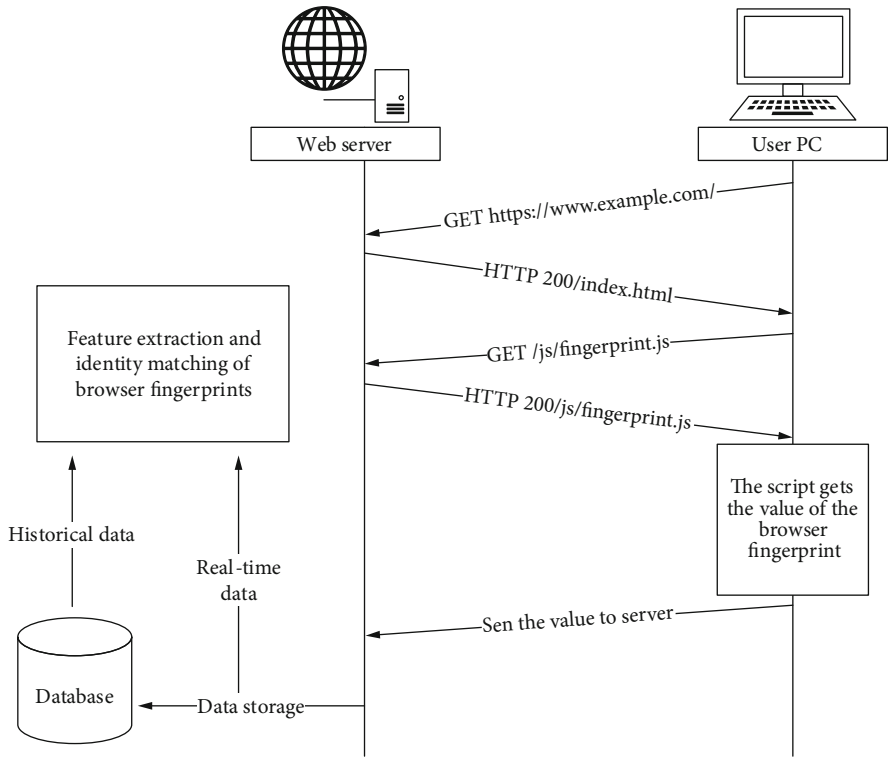

Figure 1: Browser fingerprint workflow.

3.2. Strengthen Safety Certification. Browser fingerprint not only threatens user privacy but can also be used to strengthen security authentication. For example, Unger et al. use browser fingerprint to strengthen HTTP and HTTPS identity authentication [11]. Preuveneers and Joosen's literature [54] propose a protocol that detects various parameters in session authentication and then uses adaptive and dynamic context fingerprints based on Hoeffding trees to continuously determine whether the user's identity is real or not. In 2019, Joosen et al. [55] used Canvas fingerprints made from software and hardware stacks, combined with deep learning technology, to authenticate users and thus protect against replay attacks. The entire authentication process is supported natively by any major browser, client-side stateless, transparent to the user, and very user-friendly to the user experience. In the same year, Laperdrix also adopted Canvas fingerprint to strengthen identity authentication [56]. Unlike Rocket, which uses deep learning to extract features and then compare them, Laperdrix, like Cooke, generates unique, unpredictable, and highly diverse canvas images each time a user logs into the service. The next user link must check that the current Canvas image is a perfect match for each pixel previously generated or reauthenticate. In 2021, Andriamilanto et al. [57] conducted a large-scale experiment on browser fingerprints to strengthen web authentication. Users will verify the fingerprints of the login browser each time they log in. The error rate in the author's experiment is only 0.61%, but browser fingerprint verification is best just a secondary verification; otherwise, users may fall into the river with their web accounts and mobile phones.

3.3. Protection Service Provider. Web service providers can also use browser fingerprint technology to protect themselves. Traditional intrusion detection and other network attack defense methods are relatively passive, whose main purpose is to prevent attackers and protect servers. The addition of browser fingerprint technology can trace the source of network attackers to a certain extent, so as to find out the real identity of the attackers, which, to some extent, gives web service providers the active defense ability, increases the attack cost of the attackers, and can deter the attackers to a certain extent. In 2016, Liu et al. [58] proposed to use enhanced browser fingerprint to track attackers, mainly introducing secondary attributes that are helpful for correlation judgment but are not easy to change and utilizing the storage technology of the browser. Later, Jia et al. further combined browser fingerprint and honeypot [59] and proposed a mini honeypot for browser fingerprint, which is more convenient for users to deploy and use. On the internal network of a service provider, there may be a complex intranet, and the configurations of different devices may be complex and full of vulnerabilities. Browser fingerprint technology can quickly and easily reflect the hardware and software configuration of different devices. Network administrators can perform security configuration and monitoring.

3.4. Browser Fingerprints Prevent Robot Accounts. Many companies have already adopted a variety of methods to detect robots and scripts, such as ThreatMetrix [60], Distil Networks [61], MaxMind [62], which all use browser fingerprints to detect robots and abnormal activities. In the

literature [8] the authors mentioned that their Picasso system can successfully distinguish between the browser series (Chrome, Firefox, etc.) and the operating system series (Windows, iOS, OSX, etc.) more than 52 million clients, 100% of which accuracy. It can be used to combat script abuse in the Play Store or other mobile application markets, and it can also protect user accounts from logging in from unknown devices. In 2016, Quanzhu et al. already [63] aimed at the current hospital's online registration service for popular expert accounts that have been robbed by the scalpers, combined with the characteristics of browser fingerprint technology that can identify the identity of the browser visiting users, and designed an identifiable registration system for the prevention of scalpers by the identity of the registered person. In Qingxuan's article [64], in response to the problem of false evaluation, combined with the characteristics of device fingerprints that can identify the identity of the browser visiting users, an identification system that can identify the identity of false orders is designed.

3.5. Reverse User Software and Hardware. In the article by Schwarz et al. [19], it is mentioned that reverse thinking is adopted, and the characteristics of browser fingerprints are used to reverse the characteristics of users. The user's software and hardware information can be obtained through browser fingerprints. Many users cannot install security patches or upgrade security in time; so, attackers can use the public CVE [65] vulnerabilities to carry out targeted attacks. Malwarebyte has extensively documented how malicious advertisements use fingerprints to send malware to vulnerable devices in the literature [66]. Attackers use browser fingerprinting technology to check whether users have exploitable vulnerabilities, and if so, jump to contain malicious code. Page. In 2016, Saito et al. proposed [67] to use browser fingerprints to infer the user's CPU characteristics, mainly to determine whether the CPU supports Advanced Encryption Standard New Instructions (AES-NI) and Intel Turbo Boost Technology (Turbo Boost). Later, the author carried out further advancement [68], able to identify more CPUs, and the number of CPU cores with higher precision. After the Spectre and Meltdown vulnerabilities were exposed, it can be said that the leakage of this information poses a significant security threat to users. Concerning results were shown in a 2020 [69] study, and browser extension fingerprinting may lead to personal data leakage, such as religious and medical. issues. Fortunately, these are not direct leaks, but the author's inferences based on the description of the plug-in, but it is still worth alerting.

Although browser fingerprinting technology has been applied in many scenarios, it still has not become an almost necessary technology for the web like cookies, mainly because of the following reasons.

(1) Performance consumption is a problem: compared with cookie technology, the performance consumption of browser fingerprinting technology cannot be ignored. Its main performance consumption is reflected in three aspects. The first is the acquisition of web front-end browser features, which usually requires running a large amount of JS code, which will consume a lot of user resources to run. The second point is that when transmitting fingerprint data, network delay and bandwidth are unavoidable. The third point is the performance consumption of the server for browser fingerprint matching. The consumption of these three stages is unavoidable and can only be optimized according to the needs

(2) Iterative problem of technology update: browser fingerprinting is an accessory of the rapid development of web technology, and many ways of acquiring fingerprint features will change or disappear with technological upgrades. For example, Flash-based acquisition methods no longer exist. If you want to use the browser fingerprint technology for a long time, you need to constantly follow the relevant web technology to upgrade your browser fingerprint related code

4. Evaluation and Processing Methods of Browser Fingerprints

4.1. Browser Fingerprinting Feature Evaluation Tool. Browser fingerprints are complicated, and different components have different occurrence probabilities. How to describe and measure the uncertainty of this information is a common problem faced by all scholars. In 1948, Shannon proposed the concept of "information entropy," which solved the problem of people's quantitative measurement of information. In the literature [5], the information of the browser fingerprint is modeled. The article assumes that there is a browser fingerprint algorithm $F(x)$, which is similar to the Hash function, for each input browser fingerprint x. There is a unique output $\mathrm{F}(x) = f_n, n \in [0, 1, \cdots, N]$. The probability of each result f_n is $P(f_n)$. Then, you can get the corresponding self-information amount:

$$I(F(x) = f_n) = -\log_2(P(f_n)). \tag{1}$$

The result $I(F(x))$ is rounded up to indicate how many bits are needed to represent the information, and the information entropy of the corresponding browser fingerprint is the expectation of information entropy. Here is the following formula:

$$H(f) = -\sum_{n=0}^{N} P(f_n)\log_2(P(f_n)). \tag{2}$$

The browser fingerprint is composed of the hardware and software components of multiple browsers. A similar method can be used to define the definition of a single component of the browser. The fingerprint of a single component is $F_s(\cdot), s \in S$. The self-information amount and information entropy of its individual components are

defined as follows:

$$I_s(f_{n,s}) = -\log_2(P(f_{n,s})),$$
$$H_s(F_s) = -\sum_{n=0}^{N} P(f_{n,s}) \log_2(P(f_{n,s})). \tag{3}$$

If the components are independent of each other, their information entropy can be linearly added, but this is not the case. The fingerprint components of multiple browsers are often related to each other. For example, the Edge browser is mostly related to the Windows system, while the Safari browser is often bounded to the IOS system; so, it is necessary to use conditional self-information to measure multiple components together:

$$I_{s+t}(f_{n,s}, f_{n,t}) = -\log_2(P(f_{n,s}|f_{n,t})). \tag{4}$$

The method of information entropy can be used to demonstrate the feasibility of using browser fingerprints to identify users. Since the actual probability cannot be obtained, a statistical approximation can only be obtained. Therefore, when the browser fingerprint algorithm is actually used for user tracking, it is best to first perform statistical collection of browser fingerprint information and reasonably evaluate the information volume and information entropy of each component of the browser fingerprint. When performing the comparison of browser fingerprints, a certain weight can be selected according to different information entropies.

4.2. Browser Fingerprint Evolution Evaluation Tool. But more often, we hope to quantify the degree of difference between the two browsers. Yamada uses the edit distance to describe the degree of difference [46]. Edit distance, also known as Levenshtein distance, is a quantitative measurement of the degree of difference between two-character strings (such as English characters): another string. The formula is as follows:

$$lev_{\{a,b\}(i,j)} = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(\mathrm{i}, j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{a_i \neq b_j} \end{cases} & \text{ohterwise.} \end{cases} \tag{5}$$

In its original edit distance, the unit of comparison is each character. In Yamada's paper, the original version was not used directly. Since the format of each browser plug-in or version is fixed, such as Firefox 50.0 and Chrome 60.5, the author regards each plug-in version description as a single character when comparing. Then, take the entire list of plug-ins as a string and compare them according to the edit distance. The author calls this method YIKS distance.

There is also a time-based difference characterization called disagreement decay. Disagreement decay is the probability that an entity changes the value of an attribute s within the time Δt. This probability is denoted by $d^{\neq}(s,\Delta t)$. We can characterize the probability distribution function of this probability through mathematical statistics, specifically expressed as follows:

$$d^{\neq}(A,\Delta t) = \frac{|f_{s\neq s,n,\Delta t}|}{|f_{s,n,\Delta t}|}. \tag{6}$$

The absolute value represents the number of samples, which corresponds to an agreement decay. As the name suggests, agreement decay is the probability that an entity remains the same value of an attribute s within the time Δt . Similar to the above, it is not explained.

4.3. Matching Performance Evaluation Tool. No matter what feature is used, in order to judge whether the incoming visit comes from the previous user, the website must match the browser fingerprints one by one. When the amount of browser fingerprint data is large, the positive and negative samples in the matching process are unbalanced, because the new visit must originate from a certain user, which means that the rest are negative samples. The traditional accuracy rate can no longer measure the performance of the matching algorithm. Usually, selected indicators are F1-score, G-mean, MCC ,and AUCPRC. Among them, AUCPRC stands for area under precision-recall curve, and the other three formulas are as follows:

$$\mathrm{F1-score} = 2 \cdot \frac{\mathrm{Recall} \times \mathrm{Precision}}{\mathrm{Recall} + \mathrm{Precision}},$$
$$G-\mathrm{mean} = \sqrt{\mathrm{Recall} \times \mathrm{Precision}},$$
$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}. \tag{7}$$

The measurement standard F1-score is based on the harmonic average of precision and recall, which means that in fingerprint recognition, the cost of misclassification of positive and negative samples is the same. While G-mean represents the geometric mean of classifier precision and recall, F1-score and G-mean give more importance to smaller values. MCC is the Matthews correlation coefficient, which is a relatively balanced indicator, which essentially describes the correlation coefficient between the predicted results and the actual results. When the gap between the F1-score of a classification and the MCC is large, it means that a single indicator cannot measure all the advantages and disadvantages of classifiers.

5. Discussion

The rise of browser fingerprint technology conforms to the general trend that people pay more and more attention to privacy. Traditional cookie-based user tracking technology has exposed more and more limitations. For example, cookies may be hijacked [70], modified [2], forged, and even injected from cookies [71]. More and more users choose to ban cookies or install privacy protection plug-ins, and even

recently, Google announced that it would ban third-party cookies [72]. Browser fingerprint technology will become an important way for future user tracking due to its statelessness, no storage, and wide feature sources. We have comprehensively analyzed the results of previous studies and believe that future research will have the following trends:

(1) The application of machine learning technology: this part is mainly applied to the method of browser fingerprint matching. If it is only to match whether the fingerprints are the same as in the article [11, 56], then it is only necessary to match whether the fingerprints are the same. If the fingerprint evolution is considered [5, 8, 45–49], a corresponding matching algorithm is required. Early literatures [5, 8, 58, 73] built efficient matching algorithms based on rules. With the development of machine learning and deep learning, more authors choose to use machine learning methods to analyze browser fingerprint features. For example, the literature [4, 7] uses clustering algorithm and further in order to automatically extract fingerprint signs. Some literatures [8, 9] started to use machine learning algorithms such as neural networks for fingerprint matching. It is believed that there will be more research on the combination of browser fingerprinting and machine learning technology in the future

(2) Browser fingerprint application research: although many applications of browser fingerprinting have been listed above [52–55, 58, 59, 63, 64], these applications mainly take advantage of two aspects: one is the immutability of fingerprints, and the other is the use of fingerprints. Get feature information. However, with the advancement of related research on software and hardware fingerprinting [15, 16] and related research on browser fingerprinting evolution [8, 45–49], there are more potential applications of browser fingerprinting that can be tapped, such as crossbrowser fingerprinting. Tracking, crossdomain tracking, and user portrait characterization

(3) Research on fingerprint characteristics of modern browsers: as browsers and related network technologies are constantly iterating, many technologies will be discontinued. For example, Microsoft [74], Google [75], and even Adobe [76] themselves have announced the discontinuation of flash technical support. With the rapid development of technologies such as HTML5 and CSS3, the fingerprints of the browser in the metropolis have different characteristics in different eras

The hidden worries of browser technology development are as follows: although browser fingerprinting technology is relatively mature, relevant laws, regulations, and technical specifications have long lagged behind practice [77]. If the information leakage of the previous studies [66–68] is concerned, it is still about the technical security of device information. The research of the article [69] shows the social harm of browser fingerprinting technology to personal privacy. For related solutions, in the short-term, manufacturers should continuously update versions and prohibit the acquisition of some features. In the long run, the fundamental solution still requires governments to establish relevant laws and regulations to constrain and guide relevant technology development.

6. Summary

The current research on browser fingerprints has made certain achievements, which can be used as an important part of user identity tracking technology. Although browser fingerprints are used alone as a sign of user identity, there are still many problems, but the combination of browser fingerprints and traditional user identity tracking technology can be applied in many directions, such as identity tracking, user authentication, and security defense. This article summarizes the relevant research status from three aspects of browser fingerprint acquisition, defense, and long-term tracking, proposes to further discuss the application of this technology in various aspects, and finally summarizes the related research theoretical methods of browser fingerprints.

Data Availability

The experimental data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62176264).

References

[1] K. Mathews-Hunt, "CookieConsumer: tracking online behavioural advertising in Australia," *Computer Law and Security Review*, vol. 32, no. 1, pp. 55–90, 2016.

[2] H. Kwon, H. Nam, S. Lee, C. Hahn, and J. Hur, "(In-)security of cookies in HTTPS: cookie theft by removing cookie flags," *IEEE Transactions on Information Forensics and Security*, vol. 15, p. 1204, 2020.

[3] L. F. Cranor, "Cookie monster," *Communications of the ACM*, vol. 65, no. 7, pp. 30–32, 2022.

[4] J. R. Mayer, *Any person… a pamphleteer: Internet Anonymity in the Age of Web 2.0*, Princeton University, 2009, Undergraduate Senior Thesis.

[5] P. Eckersley, "How unique is your web browser?," in *International Symposium on Privacy Enhancing Technologies Symposium*, pp. 1–18, Springer, 2010.

[6] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in HTML5," in *Proceedings of W2SP*, pp. 1–12, San Francisco, CA, USA, 2012.

[7] A. Faiz Khademi, M. Zulkernine, and K. Weldemariam, "FPGuard: detection and prevention of browser fingerprinting," in *DBSec 2015: Data and Applications Security and Privacy XXIX*, pp. 293–308, Springer, Cham, 2015.

[8] A. Vastel, P. Laperdrix, W. Rudametkin, and R. Rouvoy, "FP-STALKER: tracking browser fingerprint evolutions," in *2018 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, 2018.

[9] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham, "Fingerprinting information in JavaScript implementations," in *Proceedings of W2SP*, vol. 2, Krakow, Poland, 2011.

[10] M. Mulazzani, P. Reschl, M. Huber et al., "Fast and reliable browser identification with javascript engine fingerprinting," in *Web 2.0 Workshop on Security and Privacy (W2SP)*, vol. 5, Citeseer, 2013.

[11] T. Unger, M. Mulazzani, D. Fruhwirt, M. Huber, S. Schrittwieser, and E. R. Weippl, "SHPF: enhancing HTTP(S) session security with browser fingerprinting," in *2013 International Conference on Availability, Reliability and Security*, Regensburg, Germany, 2013.

[12] 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)*, pp. 57–63, Krakow, Poland, 2015.

[13] P. Laperdrix, O. Starov, Q. Chen, A. Kapravelos, and N. Nikiforakis, *Fingerprinting in style: detecting browser extensions via injected style sheets*, 2021, https://www.usenix.org/conference/usenixsecurity21/presentation/laperdrix.

[14] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The web never forgets: persistent tracking mechanisms in the wild," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 674–689, Vienna, Austria, 2014.

[15] G. Nakibly, G. Shelef, and S. Yudilevich, "Hardware fingerprinting using HTML5," 2015, http://arxiv.org/abs/1503.01408.

[16] 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)*, pp. 878–894, San Jose, CA, USA, 2016.

[17] E. Bursztein, A. Malyshev, T. Pietraszek, and K. Thomas, "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.

[18] Y. Cao, S. Li, and E. Wijmans, "(Cross-) browser fingerprinting via os and hardware level features," in *24th Annual Network and Distributed System Security Symposium*, Scottsdale, Arizona, USA, 2017.

[19] M. Schwarz, F. Lackner, and D. Gruss, "JavaScript template attacks: automatically inferring host information for targeted exploits," 2019, https://www.ndss-symposium.org/ndss-paper/javascript-template-attacks-automatically-inferring-host-information-for-targeted-exploits/.

[20] 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*, Vienna, Austria, 2016.

[21] J. S. Queiroz, E. L. Feitosa, and C. Mitchell, "A web browser fingerprinting method based on the web audio API," *The Computer Journal*, vol. 62, no. 8, pp. 1106–1120, 2019.

[22] A. Sjösten, S. V. Acker, and A. Sabelfeld, "Discovering browser extensions via web accessible resources," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, Scottsdale, Arizona, USA, 2017.

[23] O. Starov and N. Nikiforakis, "XHOUND: quantifying the fingerprintability of browser extensions," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 941–956, San Jose, CA, USA, May 2017.

[24] I. Sánchez-Rola, I. Santos, and D. Balzarotti, "Extension breakdown: security analysis of browsers extension resources control policies," 2017, https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/sanchez-rola.

[25] O. Starov, P. Laperdrix, A. Kapravelos, and N. Nikiforakis, "Unnecessarily identifiable: quantifying the fingerprintability of browser extensions due to bloat," in *Presented at the the World Wide Web Conference*, San Francisco, CA, USA, 2019.

[26] W. Fuhl, N. Sanamrad, and E. Kasneci, "The gaze and mouse signal as additional source for user fingerprints in browser applications," 2021, https://arxiv.org/abs/2101.03793.

[27] E. Abgrall, Y. L. Traon, M. Monperrus, S. Gombault, M. Heiderich, and A. Ribault, "XSS-FP: browser fingerprinting using HTML parser quirks," 2012, https://arxiv.org/abs/1211.4812.

[28] D. Fifield and S. Egelman, "Fingerprinting Web Users Through Font Metrics," in *Financial Cryptography and Data Security*, pp. 107–124, Berlin, Heidelberg, 2015.

[29] Ł. Olejnik, G. Acar, C. Castelluccia, and C. Diaz, "The leaking battery - a privacy analysis of the HTML5 Battery Status API," in *Data Privacy Management, and Security Assurance*, pp. 254–263, Springer International Publishing, Cham, 2016.

[30] I. Sanchez-Rola, I. Santos, and D. Balzarotti, "Clock around the clock: time-based device fingerprinting," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, Toronto, Canada, 2018.

[31] T. Wu, Y. Song, F. Zhang, S. Gao, and B. Chen, "My site knows where you are: a novel browser fingerprint to track user position," in *ICC 2021- IEEE International Conference on Communications*, Montreal, QC, Canada, June 2021.

[32] K. Boda, "Firegloves," https://fingerprint.pet-portal.eu/?menu=6.

[33] C. F. Torres, H. Jonker, and S. Mauw, "FP-Block: usable web privacy by controlling browser fingerprinting," in *Computer Security-ESORICS 2015-20th European Symposium on Research in Computer Security*, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2015.

[34] F. Besson, N. Bielova, and T. Jensen, "Browser randomisation against fingerprinting: a quantitative information flow approach," in *Secure IT Systems -19th Nordic Conference, NordSec 2014*, pp. 181–196, Springer International Publishing, Cham, 2014.

[35] N. Nikiforakis, W. Joosen, and B. Livshits, "PriVaricator: deceiving fingerprinters with little white lies," in *Proceedings of the 24th International Conference on World Wide Web*, Florence, Italy, 2015.

[36] P. Laperdrix, W. Rudametkin, and B. Baudry, "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*, pp. 98–108, Florence, Italy, May 2015.

[37] E. Trickel, O. Starov, A. Kapravelos, N. Nikiforakis, and A. Doupé, "Everyone is Different: Client-side Diversification for Defending Against Extension Fingerprinting," 2019, https://www.usenix.org/conference/usenixsecurity19/presentation/trickel.

[38] S. Wu, S. Li, Y. Cao, and N. Wang, "Rendered private: making GLSL execution uniform to prevent WebGL-based Browser fingerprinting," *28th USENIX Security Symposium 2019*, 2019, https://www.usenix.org/conference/usenixsecurity19/presentation/wu.

[39] U. Fiore, A. Castiglione, A. D. Santis, and F. Palmieri, "Countering browser fingerprinting techniques: constructing a fake profile with Google Chrome," in *2014 17th International Conference on Network-Based Information Systems*, pp. 355–360, Salerno, Italy, September 2014.

[40] S. Yokoyama and R. Uda, "A proposal of preventive measure of pursuit using a browser fingerprint," in *Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication*, Bali, Indonesia, 2015.

[41] P. Baumann, S. Katzenbeisser, M. Stopczynski, and E. Tews, "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.

[42] P. Laperdrix, B. Baudry, and V. Mishra, "FPRandom: randomizing core browser objects to break advanced device fingerprinting techniques," in *Engineering Secure Software and Systems -9th International Symposium, ESSoS 2017*, pp. 97–114, Springer International Publishing, Cham, 2017.

[43] D. Mitropoulos, K. Stroggylos, D. Spinellis, and A. D. Keromytis, "How to train your browser: preventing XSS attacks using contextual script fingerprints," *ACM Transactions on Privacy and Security*, vol. 19, no. 1, pp. 1–31, 2016.

[44] A. ElBanna and N. Abdelbaki, "NONYM!ZER: mitigation framework for browser fingerprinting," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 158–163, Sofia, Bulgaria, July 2019.

[45] Ó. Muñoz-García, J. Monterrubio-Martín, and D. García-Aubert, "Detecting browser fingerprint evolution for identifying unique users," *International Journal of Electronic Business*, vol. 10, no. 2, pp. 120–141, 2012.

[46] T. Yamada, T. Saito, K. Takasu, and N. Takei, "Robust identification of browser fingerprint comparison using edit distance," in *2015 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*, pp. 107–113, Krakow, Poland, November 2015.

[47] X. Li, X. Cui, L. Shi, C. Liu, and X. Wang, "Constructing browser fingerprint tracking chain based on LSTM model," in *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pp. 213–218, Guangzhou, China, June 2018.

[48] S. Bird, V. Mishra, S. Englehardt et al., "Actions speak louder than words: semi-supervised learning for browser fingerprinting detection," 2020, https://arxiv.org/abs/2003.04463.

[49] L. Qixu, L. Xinyu, L. Cheng, W. Junnan, C. Langping, and L. Jiaxi, "An android browser fingerprint recognition method based on bidirectional recurrent neural network," *Computer Research and Development*, vol. 57, no. 11, pp. 2294–2311, 2020, (In Chinese).

[50] S. Li and Y. Cao, "Who Touched My Browser Fingerprint? A Large-scale Measurement Study and Classification of Fingerprint Dynamics," in *Proceedings of the ACM Internet Measurement Conference*, Alsace, Colmar, France, 2020.

[51] 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, no. 2, pp. 558–577, 2020.

[52] A. ElBanna and N. Abdelbaki, "Browsers fingerprinting motives, methods, and countermeasures," in *2018 International Conference on Computer, Information and Telecommunication Systems (CITS)*, Alsace, Colmar, France, July 2018.

[53] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless Monster: exploring the ecosystem of web-based device fingerprinting," in *2013 IEEE Symposium on Security and Privacy*, pp. 541–555, Berkeley, CA, USA, May 2013.

[54] D. Preuveneers and W. Joosen, "SmartAuth: dynamic context fingerprinting for continuous user authentication," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, Salamanca, Spain, 2015.

[55] F. Rochet, K. Efthymiadis, F. O. Koeune, and O. Pereira, "SWAT: seamless web authentication technology," in *Presented at the World Wide Web Conference*, San Francisco, CA, USA, 2019.

[56] P. Laperdrix, G. Avoine, B. Baudry, and N. Nikiforakis, "Morellian analysis for browsers: making web authentication stronger with Canvas fingerprinting," in *Detection of Intrusions and Malware, and Vulnerability Assessment -16th International Conference, DIMVA 2019*, pp. 43–66, Springer International Publishing, Cham, 2019.

[57] N. Andriamilanto, T. Allard, G. L. Guelvouit, and A. Garel, "A large-scale empirical analysis of browser fingerprints properties for web authentication," *ACM Transactions on the Web*, vol. 16, no. 1, p. 4, 2021.

[58] X. Liu, Q. Liu, X. Wang, and Z. Jia, "Fingerprinting web browser for tracing anonymous web attackers," in *2016 IEEE First International Conference on Data Science in Cyberspace (DSC)*, pp. 222–229, Changsha, China, June 2016.

[59] Z. Jia, X. Cui, Q. Liu, X. Wang, and C. Liu, "Micro-honeypot: using browser fingerprinting to track attackers," in *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pp. 197–204, Guangzhou, China, June 2018.

[60] ThreatMetrix, *ThreatMetrix Announces Cookieless Device Identification to Prevent Online Fraud While Protecting Customer Privacy*https://www.threatmetrix.com/press-releases/threatmetrix-announces-cookieless-device-identification-to-prevent-online-fraud-while-protecting-customer-privacy/.

[61] D. Networks, *The Evolution of Hi-Def Fingerprinting in Bot Mitigation*https://resources.distilnetworks.com/all-blog-posts/device-fingerprinting-solution-bot-mitigation.

[62] MaxMind, *Device Tracking Add-on for Minfraud Services*https://dev.maxmind.com/minfraud/device/.

[63] Y. Quanzhu, J. Pengfei, Y. Lijing, and Z. Hongfang, "An anti-scalper registration system based on browser fingerprinting technology," *Computer Applications*, vol. 36, no. S2, p. 276, 2016, (In Chinese).

[64] X. Qingxuan, "Fake order recognition system based on browser fingerprint," *Electronic Production*, vol. 2, p. 3, 2019, (In Chinese).

[65] CVE, *Common Vulnerabilities and Exposures-The Standard for Information Security Vulnerability Names*https://cve.mitre.org/.

[66] Malwarebytes, *Operation Fingerprint-A Look into Several Angler Exploit Kit Malvertising Campaigns*https://malwarebytes.app.box.com/v/operation-fingerprint.

[67] T. Saito, K. Yasuda, T. Ishikawa et al., "Estimating CPU features by browser fingerprinting," in *2016 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, pp. 587–592, Fukuoka, Japan, July 2016.

[68] T. Saito, K. Yasuda, K. Tanabe, and K. Takahashi, "Web Browser Tampering: Inspecting CPU Features from Side-Channel Information," in *Advances on Broad-Band Wireless Computing, Communication and Applications*, pp. 392–403, Springer International Publishing, Cham, 2018.

[69] S. Karami, P. Ilia, K. Solomos, and J. Polakis, "Carnus: exploring the privacy threats of browser extension fingerprinting," 2020, https://www.ndss-symposium.org/ndss-paper/carnus-exploring-the-privacy-threats-of-browser-extension-fingerprinting/.

[70] S. Sivakorn, I. Polakis, and A. D. Keromytis, "The Cracked Cookie Jar: HTTP cookie hijacking and the exposure of private information," in *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 724–742, San Jose, CA, USA, May 2016.

[71] F. Chen, H. Duan, X. Zheng, J. Jiang, and J. Chen, "Path leaks of HTTPS Side-Channel by cookie injection," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pp. 189–203, Springer, 2018.

[72] Google, "Charting a course towards a more privacy-first web," https://blog.google/products/ads-commerce/a-more-privacy-first-web/.

[73] Z. Liangfeng, W. Yi, W. Yuanyi, and K. Rui, "Statistics-based browser fingerprinting technology," *Information Network Security*, vol. 11, pp. 49–55, 2019, (In Chinese).

[74] Microsoft, *Adobe Flash end of support on December 31, 2020*https://docs.microsoft.com/en-us/lifecycle/announcements/adobe-flash-end-of-support.

[75] Google, *Saying goodbye to Flash in Chrome*https://www.blog.google/products/chrome/saying-goodbye-flash-chrome/.

[76] Adobe, "Adobe Flash Player EOL General Information Page," https://www.adobe.com/products/flashplayer/end-of-life.html.

[77] P. Laperdrix, N. Bielova, B. Baudry, and G. Avoine, "Browser fingerprinting: a survey," *ACM Transactions on the Web*, vol. 14, no. 2, pp. 8: 1–8: 33, 2020.