School of Informatics

# Informatics Research Review
# Fingerprinting Security vs Privacy

January 2020

**Abstract**

Browser fingerprinting is an area coming to prominence and the research stands divided between those who are looking to leverage it for security applications and those looking to defend the privacy that it threatens. This review is an investigation into the compatibility of research from both sides functioning together to support first party sites while defending against third party tracking. Security research will be discussed and proposed fingerprinting countermeasures analyzed as to their effectiveness in functioning alongside. Future directions in research are proposed in order to bridge the divide to have safe, private, and effective browsing.

Date: Wednesday 8th January, 2020

**Supervisor:** Tariq Elahi

## 1 Introduction

Numerous and essential aspects of people's lives now are accessed and managed through websites on the internet. Within this context there are two conflicting concerns; a need to identify users for security purposes when interacting with sensitive sites such as banks, and a desire to protect their privacy. Numerous studies have shown that in addition to the behavioural tracking for tailored advertisements, there exist dangers of security exploits and re-identification of users from the tracking of browsers [1].

*Browser fingerprinting* had long been active within industry for these commercial reasons but gained traction in the academic conscious primarily with Eckersley's development of Panopticlick in 2010 [2]. However, this interest is strongly divided with those using it as a tool to supplement authentication or verification schemes [3, 4, 5, 6] and a far larger group looking to block it all together for privacy protection [7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. What appears to be missing as a result of this division and what this review will investigate is whether both security for first party sites and privacy against third party tracking can be supported simultaneously.

This type of stateless tracking is rising to prominence with the more recent studies reporting 68.8% of Alexa Top 10,000 Sites containing some type of fingerprinting [7] increasing from original reports of 0.4% in 2013 [13]. And while stateful tracking such as cookies have long running and popular control mechanisms such as Privacy Badger and legislation such as the GDPR, browser fingerprinting's invisible presence has allowed it to grow unchecked.

Browser fingerprinting is defined consistently in the literature as a stateless action of collecting measurable web accessed attributes, *vectors*, to build an identity for tracking over a period of time [17]. These vectors when proposed and studied are typically measured by Shannon entropy to determine effectiveness of use. For the purposes of this paper analysis will be limited to desktop browser fingerprinting to maintain consistency in literature comparison. Two previous literature surveys exist from recent years, Bujlow [18] and Browser Fingerprinting: A Survey [19], unofficially published by known authors in the field. This review diverges substantially from them in its account of security uses for fingerprinting as well as the focused nature of the analysis regarding cooperation with fingerprinting countermeasures.

Section 1 will introduce the work that has been produced on security uses of fingerprinting. Section 2 will contain countermeasure proposals organized by strategy with each implementation analyzed for it's compatibility with the previously introduced security features.

## 2 Literature Review

### 2.1 Security Uses

Much of the research literature on browser fingerprinting is divided into identifying new or stronger vectors for creating fingerprints or defending against those discoveries. However, beginning with the work on fingerprinting there has been a growing movement towards *multidimensional authentication* where authentication is made on a best effort basis using passwords and implicit signals that, while potentially forgeable, are numerous and increase the computational difficulty of an adversarial attack [20]. The important vector attributes to evaluate a fingerprinting system's accuracy are the *uniqueness*, of particular concern to privacy advocates, and the *stability*, or amount of change it undergoes over time [21].

### 2.1.1 Detecting Session Hijacking

One of the first major academic papers looking into using fingerprinting for security purposes was SHPF [3] by a private research group in Vienna in 2013. Its main goal was to detect sudden changes in a browser's attributes that would indicate the presence of session hijacking at a time that it and related issues dominated OWASP's Top Ten list. The solution was to be used as an additional security feature against an adversary possessing a session token.

The authors built off of the OWASP AppSensor Project[1] but expanded from monitoring only IP and the UserAgent string to a flexible feature monitoring system primarily supporting HTML header tracking and fingerprinting with CSS vectors [3]. Their prototype broke fingerprinting checks down into *static/synchronous*, and *dynamic/asynchronous* checks that could be performed constantly or when requesting a behaviour requiring a higher degree of security [3].

However, their static checks omitted high entropy fingerprinting vectors that were previously established by measurement surveys [2, 22, 13] and put aside new developments in dynamic fingerprinting such as the canvas vector [23] leaving them for future development. Because of these omissions the authors were unable to do more than determine the browser type excluding the version and relied on the IP address monitoring as a stronger defence against spoofing [3]. As the primary vectors chosen, using HTML header ordering and partial support for CSS attributes at the time CSS3 was being standardized appear vulnerable to significant change over time and standards requiring constant maintenance. In a later study it was found that if browsers were to strictly follow the HTTP header standards successful identification using them would drop from 90% to 82% [24].

Despite these limitations further authentication frameworks built upon this fundamental idea of consistent monitoring for identity changes.

### 2.1.2 Enhancing Identity Checks

Preuveneers and Joosen [4] followed this idea by developing SmartAuth in 2015, an authentication framework to reshape identity management. Specifically focusing on reliability and less intrusion to users than passwords in order to to combat widespread data breaches at the time. Following a similar model as before, this involves continuous capture and comparison of a browser fingerprint, but differs substantially in the approach.

SmartAuth is about risk based assessment and uses similarity preserving hash functions to compute the fingerprint in order to allow mutability within certain parameters as calculated by Hoeffding trees bound by a threshold [4]. To prevent replay or phishing attacks addition fields such as a timestamp and checksum are added to the hash. Should there be enough difference in the attributes of the user, it will trigger a secondary check of 2FA[2] or OTP[3] to verify the user.

The authors here do address the issue of privacy concerns from the collection of contextual information about the user. By hashing the fingerprint on the client side they avoid sending user properties over the network as seen in other works [4]. Furthermore, the tool operates on an explicit consent model per property that can be read. As a result the measure of uniqueness of a fingerprint is dependent on the properties allowed by the user. This was not fully tested in their methodology, as they used set test profiles of properties in different scenarios, and so necessitated future work in a negotiation protocol between client and server as well as

[1]https://www.owasp.org/index.php/OWASP_AppSensor_Project
[2]Two factor authentication
[3]One time passcode

identification accuracy[4].

### 2.1.3 Filtering Web Traffic

Stepping back from identifying individuals Bursztein et al. [5] from Google faced the issue of detecting legitimate clients from automated adversaries on a large scale in 2016. Their proposal, Picasso, uses canvas fingerprinting [23] to detect emulated hardware from spoofed devices in order to filter web traffic [5]. Contrary to previous research the graphical challenges generating the fingerprint are desired to minimize entropy between devices of a similar device class and maximize differences across classes preserving privacy [5].

Assuming an adversary with knowledge of their system Picasso requires the browser to draw randomly selected graphical primitives with randomized properties for a number of rounds cumulatively hashing the response before sending it back [5]. By using rounds and a certain size of canvas the researchers aim to both prevent replay attacks and enforce a proof of work by the client [5]. Due to differences in hardware and software versions footprint instability is acceptable to a point that Picasso stores a set of responses for each challenge averaging 10 in the worst case [5].

Picasso claims 100% accuracy from testing against 52 million clients but within their analysis relies on the modifiable User-Agent field for verifying the browser and operating system of challenged clients in their test set [5]. As seen in the strategies of many anti-fingerprinting measures [18, 16, 25], this field is manipulated and while result clustering was used in the measurement to try to combat it, widespread usage of these measures could have introduced skewed results [5].

### 2.1.4 Identical Identification

In 2019 these concepts of fingerprint monitoring for identity management and challenges with canvas fingerprinting were put together in research by Laperdrix et al. [6]. To compensate for non utilization of 2FA they proposed a fingerprint based scheme for authentication that used dynamic vectors such as canvas, audio, and crypto in a challenge based system. It was operated under the threat model of an adversary unaware of the client's software and hardware stack yet able to challenge the client themselves in a computationally bounded way [6]. Their primary goals, to protect against phishing of static browser properties and replay attacks.

For each authentication attempt two canvas element images are rendered. One to compare for the current session and the other to be used for comparison for a following attempt [6]. Expanding on the original canvas tests of Mowery and Shacham [23] Laperdrix et al. investigated the features that demonstrated the most entropy with regards to nuances in a user's software and hardware stack. Those uncovered were fonts, emojis, colour gradients, mathematical curves, shadows, and size [6]. Their efficiency showed when testing the same challenge against all participants, 98.7% of responses represented less than 0.1% of the total participants with only one set above 5% [6].

Results were compared on a pixel by pixel basis and did not allow for slight variations from any instability in the canvas vector. From their test data around 4% of their over 2000 participants were considered an outlier for having over eight canvas changes in a year time period [6]. The canvas vector can change from browser, OS, or hardware graphics updates so eight changes in a year appears to be low threshold to discount data at. If there was a canvas difference when running the system where the response was not a match, the user was asked to manually re-authenticate.

### 2.1.5 Summary

From the security proposals presented one can see that the common mechanism of continuous fingerprint monitoring is used not to identify a particular individual but to supplement risk analysis of the main authentication strategy. There is a bias away from vectors such as JavaScript and plugins, previously associated as having high uniqueness or entropy and often targeted by countermeasures [2, 26, 17, 27, 18]. Instead canvas and browser attributes are favoured as more stable over time for re-identification. These vectors are not as widely relied on for commercial cross-browser tracking due to the dependence on the specific browser [26, 28].

## 2.2 Compatibility of Countermeasures

The defensiveness in the fingerprinting research comes out of a long history with stateful tracking using cookies as a way to profile user's web usage for commercial gain with little regard to personal exposure. Even those in favour of augmenting authentication and security with these tools are concerned with the incentive to invade privacy in the pursuit of accuracy. To approach this Torres et al. in their countermeasure paper suggest the separation of first and third party trackers [12]. The former using tracking for authentication or personalization in one location that the user is present and the latter which can present nuanced and aggregated information about a user without consent to a site they've never been to. In the following section fingerprinting countermeasure tools are presented and analyzed as to their ability to support first party tracking such as those detailed in the previous section while maintaining their countering of fingerprinting for the third party trackers.

### 2.2.1 Blocking

Blocking strategies are a response to the ability of fingerprinting to occur without knowledge or consent. Unlike cookies, there is no state on the client side indicating such an action has occurred. Furthermore users cannot destroy identifiers in the way they can remove cookies as the fingerprints are stored remotely on company servers [26].

Merzdovnik et al. [16] in 2017 and more recently Datta et al. [21] in 2019 performed an aggregated study of some of the more popular blocking extensions including AdBlock Plus[4], Ghostery[5], and Privacy Badger[6] among others. The original study categorized blocking rule sets as *community-driven* with open and public modifications, *centralized* where an organization curates rules, and algorithmic where the sets are built from observation and usage [16]. These blocking lists are enacted through DNS blocking, filtering through interception proxies to remove cookies or tracking code, and DOM element removal [16].

A fundamental flaw of block lists is that they will never be complete and therefore will always be less effective [29]. Measurements by Datta et al. indicate that having a AFPET[7] like these is only marginally better than nothing with an entropy decrease from 13 to 11 bits [21, 25].

[4]https://adblockplus.org/
[5]https://www.ghostery.com/
[6]https://www.eff.org/privacybadger
[7]Anti-Fingerprinting Privacy Enhancing Technology

With regards to interference with security tools community-driven rule sets are vulnerable to malicious actors adding the domains of the security fingerprinters to prevent authentication checks. Should this occur it would not prevent a user from accessing the service as the solutions proposed in the first section are all augmentations to existing systems. On the other hand, if using an algorithmic approach, the security tooling would not be blocked as it conforms to the same origin policy. Furthermore, should a blocker extension use an interception proxy the IP address would lose it's uniqueness as a property affecting anti spoofing mechanisms employed by SHPF and SmartAuth.

On a different tact, Al-Fannah et al. [7] developed Fingerprint-Alert to target the 17 attributes they discovered during their measurement of the presence of fingerprinting on the web. Built as a browser add on it operated by monitoring HTTP traffic when a site was loaded and alerting the user if any of the 17 attributes were being relayed back to another server [7]. Users could block these HTTP messages, preventing both the flagged field and any others in the request from being received. While usability was verified there was no testing as to the effectiveness of this strategy for preventing fingerprinters from forming a fingerprint with other vectors. Even so, this approach of asking user consent for each fingerprinting action was shown to be successful by the Tor Browser to combat canvas fingerprinting and could succeed here if the 17 attributes are among the most common of vectors [14].

The 17 targeted attributes do include graphics related information about the WebGL and GPU of the device [7] but do not specifically include one that would interfere with a response to a canvas challenge. Additionally, as admitted by the author, encryption or unknown attributes will easily circumvent the tool [7]. Since the tool alert displays the source and destination urls it would be possible for a user to allow first party tracking with regards to security fingerprinting practices and block third party requests. However, automation would be required for usability with continuous authentication.

For working successfully with security fingerprinting systems the latter system of blocking information and whitelisting acceptable requests shows promise over blacklisting domains and removing page content.

### 2.2.2 Randomness

The randomness strategy for countermeasures aims to break the stability of the vectors used by the fingerprinter.

PriVaricator [8] is a widely analyzed circumvention tool that sought to interrupt the linkability of fingerprints by spoofing browser parameters within a small threshold of the original value. After a certain threshold of accesses had occurred for the variable, the tool hid a randomized portion of a user's plugins and modified the offsetHeight and offsetWidth parameters $\pm 5\%$ noise [8]. The success of this strategy ranged from 37% to 96% of the spoofed values resulting in a new fingerprint mainly dependent on how the features were weighed by which 3rd party fingerprinting tool.

Using PriVaricator while interacting successfully with a security fingerprinting tool would depend on their thresholds on variable change tolerated by the authenticator. Since the security tools work as augmenters to main security systems, if the randomized variable for PriVaricator was too great a difference, the additional authentication mechanism would be triggered for verification [4, 6] whereas smaller random variables might remain acceptable depending on the level of risk involved. Granted that if the difference is too small the tool might fall prey to linking fingerprints. With respect to the canvas challenges of Picasso and Smart Auth, the authors proposed a similar tactic of adding noise to the image pixels which would interfere. Fortunately this technique is later shown to be flawed by the same authors as vulnerable to removing the noise with statistical analysis over multiple tests [13].

FP-Random [9] in 2017 did attempt this technique of adding randomization into the canvas rendering in order to prevent tracking across sites. However, unlike a standard canvas poisoner which manipulates pixels FP-Random takes the elements with non-deterministic specifications such as colour or font and adjusts them slightly, ideally with no detectable difference for the user [9]. They extend this strategy to the audio vector and also randomize the order of JavaScript objects enumerated by the browser to cover vectors of large entropy. For usability they have two strategies, *Random*, which changes the values on every execution and *Session*, where the values are fixed at startup for the entirety of the browsing session [9].

Should the Session strategy be used, most of the security measures as described above would perform as expected for the duration of the session and only fail between browsing periods. This would eliminate the potential user functionality of having a trusted device that, when recognized, would prevent the need for further identity verification but would also prevent third party tracking across browsing sessions. Depending on the instability permitted by the Picasso project, if the specifications were too far from the device class the user could be tagged as non-organic traffic as they are aiming not to identify individuals but clusterings of similar software and hardware stacks.

A year later some of the aspects of FP-Random could be seen in the Disguised Chromium browser from Baumann et al. [10]. These were part of a configuration framework testing different strategies to prevent re-identification, one of larger anonymity sets and one of constant changes. In the implementation, a browser configuration snapshot from the client was stored in a centralized server and replaced with either a *common shared configuration* to hide amongst the many or a *random configuration* from someone else to disengage trackers [10]. This strategy of using configurations from other user's combination of browser and OS had been proposed by Fiore et al. earlier [15] but had lacked significant analysis. Moreover additional protections against canvas fingerprinting enact random changes for colour on a pixel level but only those close to a colour border where the pixels were not uniform [10]. Unfortunately, the larger anonymity option did not perform well and continued to generate unique fingerprints for those using it likely due to a more encompassing vector set used by the fingerprinter.

Evaluating the more successful random configuration option the protections hold true for the entirety of a browsing session and are not modified per request. In particular, the implementation of the canvas manipulation involves tracking the pixel changes made and replicating them for additional requests during the same session [10]. For the authentication framework by Laperdrix et al. this would uphold as a single user even with the enforcement of exact stability. However for the other tool based on canvas challenges, the spoofed browser configurations and randomness in canvas rendering would prevent Picasso's device class identification.

Despite the overall goal of breaking the linkage between multiple fingerprints, randomness should not prevent the operation of security fingerprinting tools in the case when it is static for the duration of the session. For any other mode of operation, success would be dependent on the tolerance of instability by the security tool and the bounds of randomness produced.

### 2.2.3 Profiles

The Blink [11] tool developed in 2015 sought to solve the inherent issues found in early randomization solutions by having sessions run in different virtualized configurations of OS, browser, installed fonts and plugins. This solved the inconsistencies seen with random values but proved to be unusable from a user perspective despite attempts at porting personalized data such as bookmarks between the various browsers [11]. Two strategies, *leery* and *coffee break* were provided, the latter only manipulating installed fonts and plugins for a less disruptive experience [11]. However, using leery each browsing session would result in a separate combination of platform components from a pool resulting in statistically uncommon combinations such as an IE browser on a Linux OS [11, 30].

The coffee break strategy presented in Blink of a manipulation of the static browser configurations could work with both Laperdrix's later authentication proposal and Picasso. A canvas fingerprinting challenge would surface the same result from the browser type and hardware but an encompassing footprint relying on the relative stability of those changed values, fonts being a field with particularly high entropy [17], would fail. However, by maintaining the stack who's characteristics would show in dynamic fingerprinting efforts using canvas or audio the user would be vulnerable to third party tracking across both sites and browsing sessions.

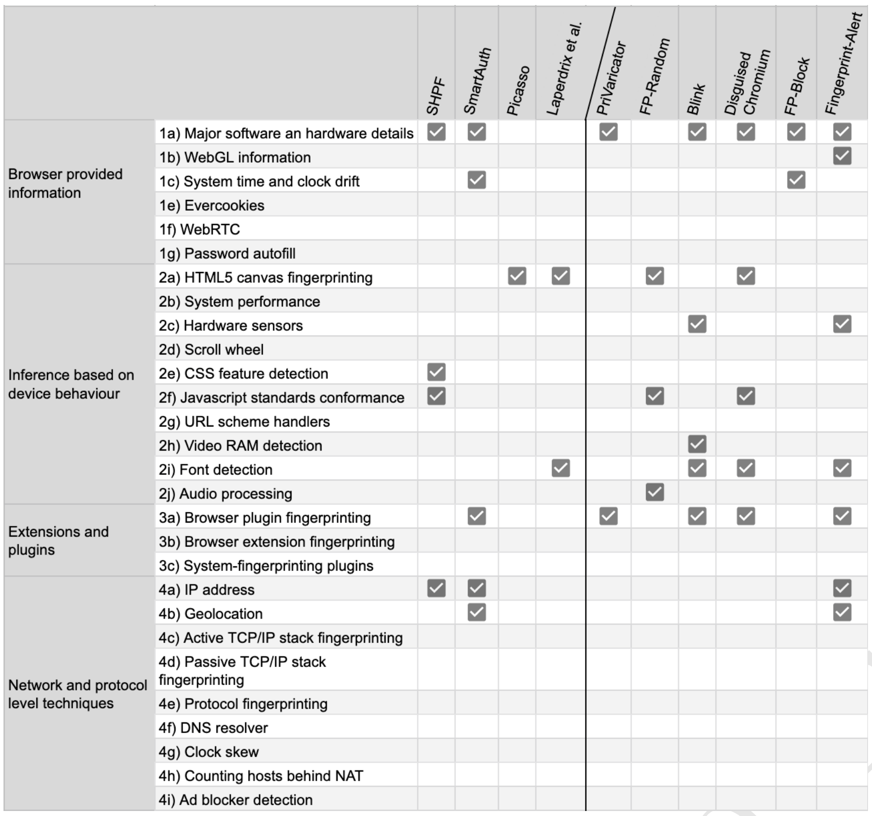
FP-Block presented by Torres et al. [12] extends the idea of presenting a consistent profile to individual sites but each site receives a unique fingerprint. In order to avoid the property inconsistencies noted in many spoofed tools [30] profiles are built off of Markov chains built with usage statistics for the combinations amongst four different categories of fingerprinting vectors [12]. FP-Block operates as a Firefox plugin and presents the profiles by modifying all HTTP traffic and blocking JavaScript access to spoofed variables by defining variable getters [12]. In addition to address canvas fingerprinting, the tool adds noise and the extension's logo to a random location on the canvas and stores it for continuity throughout the session [12].

Torres et al. are the only authors discussed here that make a substantial effort to separate 1st party and 3rd party tracking, only targeting the former. By maintaining a consistent profile with the 1st party site, authentication is able to proceed unhindered whereas cross site tracking is encumbered from unlinkable fingerprints. Their fingerprint remains constant and re-identifiable to the site its shared with through browser and component upgrades and is noted to cooperate with other popular privacy extensions such as AdBlock Plus and Privacy Badger [12]. Though reliant on a specific rendering of a canvas based on the client's stack, if usage of FP-Block grew it might be feasible for Picasso to model an instability interval for their canvas result that encompassed the modifications made here.

### 2.2.4 Compatibility Overview

Figure 1 demonstrates the collision of vectors between the security fingerprinting tools and the countermeasures analysed. It uses the vector classification introduced by Alaca and Van Oorshot [31] which is expansive in it's coverage but does not represent smaller details. Many of the tools from both sides manipulate or measure browser supplied attributes such as screen size, user agent, or language. Regardless from the figure one can see there is no security tool that does not clash with at least two countermeasures regarding a vector.

Surprisingly as the security feature least invasive into individually identifying vectors Picasso fared the worst for interference by countermeasures. Whether values were adjusted or completely lied about they would have an effect on the bucketing of groups required to identify standardized



| | | SHPF | SmartAuth | Picasso | Laperdrix et al. | PriVaricator | FP-Random | Blink | Disguised Chromium | FP-Block | Fingerprint-Alert |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Browser provided information | 1a) Major software an hardware details | ✓ | ✓ | | | ✓ | | ✓ | ✓ | ✓ | ✓ |
| | 1b) WebGL information | | | | | | | | | | ✓ |
| | 1c) System time and clock drift | | ✓ | | | | | | | ✓ | |
| | 1e) Evercookies | | | | | | | | | | |
| | 1f) WebRTC | | | | | | | | | | |
| | 1g) Password autofill | | | | | | | | | | |
| Inference based on device behaviour | 2a) HTML5 canvas fingerprinting | | | ✓ | ✓ | | ✓ | | ✓ | | |
| | 2b) System performance | | | | | | | | | | |
| | 2c) Hardware sensors | | | | | | | ✓ | | | ✓ |
| | 2d) Scroll wheel | | | | | | | | | | |
| | 2e) CSS feature detection | ✓ | | | | | | | | | |
| | 2f) Javascript standards conformance | ✓ | | | | | ✓ | | ✓ | | |
| | 2g) URL scheme handlers | | | | | | | | | | |
| | 2h) Video RAM detection | | | | | | | ✓ | | | |
| | 2i) Font detection | | | | ✓ | | | ✓ | ✓ | | ✓ |
| | 2j) Audio processing | | | | | | ✓ | | | | |
| Extensions and plugins | 3a) Browser plugin fingerprinting | | ✓ | | | ✓ | | ✓ | ✓ | | ✓ |
| | 3b) Browser extension fingerprinting | | | | | | | | | | |
| | 3c) System-fingerprinting plugins | | | | | | | | | | |
| Network and protocol level techniques | 4a) IP address | ✓ | ✓ | | | | | | | | ✓ |
| | 4b) Geolocation | | ✓ | | | | | | | | ✓ |
| | 4c) Active TCP/IP stack fingerprinting | | | | | | | | | | |
| | 4d) Passive TCP/IP stack fingerprinting | | | | | | | | | | |
| | 4e) Protocol fingerprinting | | | | | | | | | | |
| | 4f) DNS resolver | | | | | | | | | | |
| | 4g) Clock skew | | | | | | | | | | |
| | 4h) Counting hosts behind NAT | | | | | | | | | | |
| | 4i) Ad blocker detection | | | | | | | | | | |

Figure 1: Security and countermeasure usage of vectors classified by Alaca and Van Oorschot [31]

device classes. Furthermore it is the standardization of device stacks and skewed usage rates in software combinations that weakened the effectiveness of the countermeasure and revealed the spoofing of values.

For the strategies that showed promise the common theme was that as long as a consistent fingerprint was available to first parties for the length of a session, risk based authentication could perform relatively unhindered. Nevertheless there was no clear strategy that allowed for continued risk based analysis over sessions while still preventing third party tracking. The line of investigation with the most potential for a solution is expanding on Torres et al.'s work using different profiles per domain in order to preserve continuity and separation of the fingerprints.

## 3 Summary & Conclusion

The aim of this review was to introduce the research done around two divided portions of the browser fingerprinting literature and investigate paths forward that had potential to bridge the security vs privacy conflict. There are far fewer published papers on security uses of browser fingerprinting than countermeasures despite a similar timeline for the first publication in that subject matter. Countermeasures are being used by a growing subset of security minded individuals but industries like finance have been using fingerprinting to augment security since 2000 [20] ten years before Eckersley published his well known paper about the concept in general.

While the two sides do present conflict as discussed in 2.2.4 there are possible points of coexistence where effort is made to preserve consistent first party tracking and security tools allow users to consent to the information used. Future work in the area of countermeasures should consider following a profile approach to maintain the stability of vectors while keeping information in isolation. On the other side future security fingerprinting tools should explore functioning on consistency of fingerprints rather than exact identifying attributes.

As stated at the beginning of this paper, fingerprinting has grown from an insignificant number to over half of the top most visited websites. These are but two of the research areas required for further development but are of strong importance for safe and private browsing.

## References

[1] Jonathan R. Mayer and John C. Mitchell. Third-Party Web Tracking: Policy and Technology. In *2012 IEEE Symposium on Security and Privacy*, pages 413–427, May 2012. ISSN: 2375-1207, 1081-6011, 1081-6011.

[2] P. Eckersley. How unique is your web browser? *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6205 LNCS:1–18, 2010.

[3] T. Unger, M. Mulazzani, D. Fruhwirt, M. Huber, S. Schrittwieser, and E. Weippl. SHPF: Enhancing HTTP(S) session security with browser fingerprinting. pages 255–261, 2013.

[4] Davy Preuveneers. SmartAuth: dynamic context fingerprinting for continuous user authentication. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing 2015*, volume 13-17-April-2015, pages 2185–2191. ACM; New York, USA, 2015.

[5] E. Bursztein, A. Malyshev, T. Pietraszek, and K. Thomas. Picasso: Lightweight device class fingerprinting for web clients. pages 93–102, 2016.

[6] P. Laperdrix, G. Avoine, B. Baudry, and N. Nikiforakis. Morellian analysis for browsers: Making web authentication stronger with canvas fingerprinting. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11543 LNCS:43–66, 2019.

[7] N.M. Al-Fannah, W. Li, and C.J. Mitchell. Beyond Cookie Monster Amnesia: Real World Persistent Online Tracking. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11060 LNCS:481–501, 2018.

[8] N. Nikiforakis, W. Joosen, and B. Livshits. PriVaricator: Deceiving Fingerprinters with little white lies. pages 820–830, 2015.

[9] P. Laperdrix, B. Baudry, and V. Mishra. FPRandom: Randomizing core browser objects to break advanced device fingerprinting techniques. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10379 LNCS:97–114, 2017.

[10] Peter Baumann, Stefan Katzenbeisser, Martin Stopczynski, and Erik 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 - WPES'16*, pages 37–46, Vienna, Austria, 2016. ACM Press.

[11] P. Laperdrix, W. Rudametkin, and B. Baudry. Mitigating Browser Fingerprint Tracking: Multi-level Reconfiguration and Diversification. pages 98–108, 2015.

[12] C.F. Torres, H. Jonker, and S. Mauw. FP-block: Usable web privacy by controlling browser fingerprinting. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9327:3–19, 2015.

[13] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting. In *2013 IEEE Symposium on Security and Privacy*, pages 541–555, May 2013. ISSN: 1081-6011, 1081-6011.

[14] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. The Web Never Forgets: Persistent Tracking Mechanisms in the Wild. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 674–689, New York, NY, USA, 2014. ACM. event-place: Scottsdale, Arizona, USA.

[15] U. Fiore, A. Castiglione, A.D. Santis, and F. Palmieri. Countering browser fingerprinting techniques: Constructing a fake profile with google chrome. pages 355–360, 2014.

[16] Georg Merzdovnik, Markus Huber, Damjan Buhov, Nick Nikiforakis, Sebastian Neuner, Martin Schmiedecker, and Edgar Weippl. Block Me If You Can: A Large-Scale Study of Tracker-Blocking Tools. In *2017 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 319–333, April 2017. ISSN: null.

[17] D. Fifield and S. Egelman. Fingerprinting web users through font metrics. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8975:107–124, 2015.

[18] T. Bujlow, V. Carela-Espanol, B.-R. Lee, and P. Barlet-Ros. A Survey on Web Tracking: Mechanisms, Implications, and Defenses. *Proceedings of the IEEE*, 105(8):1476–1510, 2017.

[19] Pierre Laperdrix, Nataliia Bielova, Benoit Baudry, and Gildas Avoine. Browser Fingerprinting: A survey. *arXiv:1905.01051 [cs]*, November 2019. arXiv: 1905.01051.

[20] BonneauJoseph, HerleyCormac, van OorschotPaul C, and StajanoFrank. Passwords and the evolution of imperfect authentication. *Communications of the ACM*, June 2015.

[21] A. Datta, J. Lu, and M.C. Tschantz. Evaluating anti-fingerprinting privacy enhancing technologies. pages 351–362, 2019.

[22] K. Boda, Á.M. Földes, G.G. Gulyás, and S. Imre. User tracking on the web via cross-browser fingerprinting. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7161 LNCS:31–46, 2012.

[23] Keaton Mowery and Hovav Shacham. Pixel Perfect: Fingerprinting Canvas in HTML5. page 12, 2012.

[24] P. Laperdrix, W. Rudametkin, and B. Baudry. Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints. pages 878–894, 2016.

[25] Johan Mazel, Richard Garnier, and Kensuke Fukuda. A comparison of web privacy protection techniques. *Computer Communications*, 144:162–174, August 2019.

[26] R. Upathilake, Y. Li, and A. Matrawy. A classification of web browser fingerprinting techniques. 2015.

[27] Keaton Mowery, Dillon Bogenreif, Scott Yilek, and Hovav Shacham. Fingerprinting Information in JavaScript Implementations. page 11, 2011.

[28] Steven Englehardt and Arvind Narayanan. Online Tracking: A 1-million-site Measurement and Analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1388–1401, New York, NY, USA, 2016. ACM. event-place: Vienna, Austria.

[29] Balachander Krishnamurthy and Craig E. Wills. Generating a privacy footprint on the internet. In *Proceedings of the 6th ACM SIGCOMM on Internet measurement - IMC '06*, page 65, Rio de Janeriro, Brazil, 2006. ACM Press.

[30] Vafa Andalibi, Francois Christophe, and Tommi Mikkonen. Analysis of Paradoxes in Fingerprint Countermeasures. page 4.

[31] F. Alaca and P.C. Van Oorschot. Device fingerprinting for augmenting web authentication: Classification and analysis of methods. volume 5-9-December-2016, pages 289–301, 2016.