Extending Browser Extension Fingerprinting to Mobile Devices

Brian Hyeongseok Kim
brian.hs.kim@usc.edu
University of Southern California
Los Angeles, CA, USA

Shujaat Mirza
shujaat.mirza@nyu.edu
New York University
New York, NY, USA

Christina Pöpper
christina.poepper@nyu.edu
New York University Abu Dhabi
Abu Dhabi, UAE

ABSTRACT

Browser extensions are tools that extend basic browser features to enhance web experience. It has been shown that extensions can be exploited to fingerprint users and even infer personal information about them [8, 10]. However, as browser extensions have been limited to desktops previously, no prior work has explored fingerprintability of extensions on mobile devices, despite the increasing extension support for mobile browsers. This paper aims to fill this gap by extending extension fingerprinting techniques, traditionally performed on desktops, to mobile phones. Out of the 16 chosen extensions, we discover that 6 extensions are uniquely identifiable by their client-side modifications. We present our experimental results through our evaluation of variable interactions between various browsers, devices, and extension lists, and investigate how shifting the attention from the list of installed extensions to the actual modification data can help attackers discriminate users better.

CCS CONCEPTS

• **Security and privacy → Pseudonymity, anonymity and untraceability**; **Browser security**.

KEYWORDS

extension fingerprinting; browser extensions; mobile browsers

ACM Reference Format:
Brian Hyeongseok Kim, Shujaat Mirza, and Christina Pöpper. 2023. Extending Browser Extension Fingerprinting to Mobile Devices. In *Proceedings of the 22nd Workshop on Privacy in the Electronic Society (WPES '23), November 26, 2023, Copenhagen, Denmark.* ACM, Copenhagen, Denmark, 6 pages. https://doi.org/10.1145/3603216.3624955

1 INTRODUCTION

Browser fingerprinting is a technique used to identify users based on attributes collected from web browsers, bypassing the need for stateful identifiers like cookies. One attribute utilized in browser fingerprinting is the inventory of installed extensions, which enhance the default web browsing experience by offering customized functionalities. Various techniques [12, 16, 17, 22] have been devised to enumerate extensions on different browsers on a large scale. Consequently, countermeasures have been developed to obscure information from potential attackers while preserving the functionality of the extensions [9, 23].

As web experience becomes more mobile-driven, it is imperative to investigate the potential risks associated with extension fingerprinting on mobile platforms. Unlike previous works focused on desktop environments, this work explores the effectiveness of these techniques on mobile browsers. Furthermore, we examine the interplay between device, browser, and extension list to derive meaningful insights from our findings.

This paper focuses on two behavioral methods of extension fingerprinting, Document Object Model-based (DOM) [22] and Cascading Style Sheet-based (CSS) [12] techniques. Unlike non-behavioral techniques that rely on extensions' configurations, which could be obfuscated or randomized per session [16], behavioral techniques rely on changes made to page elements, which are inherently tied to the extensions' functionalities.

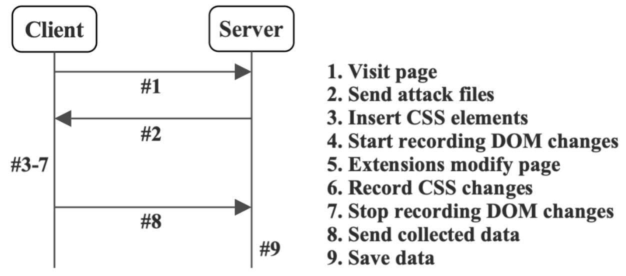
Attacker Model: We consider a malicious individual aiming to fingerprint users' extensions by placing a tracking script on a webpage that potential victims will access. Because extensions only have access to modify pre-specified domains, our attack model assumes that the tracking script is loaded on a domain accessible by the extension, which can modify elements on this malicious page as desired by extension functionalities. This scenario is definitely possible, since an inexperienced extension developer could just request access to all domains and a naive user can grant this extension access to every domain they visit, including the malicious webpage.

We also assume that a simple page visit launches the attack. While certain extension functionalities may rely on user interactions (e.g. starting to type for spell check) or specific elements (e.g. a login form), our focus in this paper is on modifications that do not necessitate any user input or preexisting elements. As a result, anyone visiting the malicious page is considered a potential victim.

DOM Technique: Our DOM technique is derived from Starov and Nikiforakis [22], who dynamically create DOM elements queried by extensions and record any changes made to these elements, similar to the idea of honey pages [7]. This technique can also track other types of modifications besides insertion of a new element, such as removal or changes of existing elements. Considering our attacker model, we modify this method and observe all DOM changes that are made to our empty static page without dynamically creating elements that extensions may look for. In other words, we track any additions, removals, or changes to DOM elements that extensions make with no prior requirements (Listing 1 in Appendix A).

CSS Technique: Our CSS technique is derived from Laperdrix et al. [12], who collect unique id or class names of HTML <div> tags that are specific to certain extensions' modifications. These identifiable names are then used to create what we refer to as a "div pair", two <div> elements with identical names (Listing 2 in Appendix A). While one div is styled by the extension based on its functionality, the other remains unchanged. Consequently, any

Figure 1: Our extension fingerprinting attack pipeline



CSS discrepancies within a pair indicate that a certain extension has modified it.

Our paper enhances the understanding of extension fingerprinting by being the first to explore extension fingerprinting in the mobile context, to the best of our knowledge. In addition, we move beyond the binary identification of whether or not a specific extension is installed and instead focus on the granular data obtained from the modifications made by extensions. Finally, we derive additional insights about the effect of variable interaction on the uniqueness of extension fingerprints through our cross-device and cross-browser comparisons.

The remainder of this paper is as follows. Section 2 gives an overview of our experimental design and results. Section 3 presents our main takeaways. Section 4 discusses related work. Section 5 concludes the paper with future work. Appendix A contains example scripts. Appendix B provides additional details on our results.

2 EXPERIMENTS AND RESULTS

2.1 Data Collection & Study Design

We developed a JavaScript fingerprinting attack, hosted on an Ubuntu server. Since mobile browser extensions are still in their early stages, their usage is not widespread yet. Thus, instead of a user study, we opted for a lab experiment with various permutations of device, browser, and extension list to gather comprehensive data points representing different users.

The overarching pipeline of our fingerprinting attack is shown in Figure 1. First, the client first visits our web page hosted on the server and receives our HTML and JavaScript files, which include our attack program code. Our server page, by design, is initially an empty HTML document with reference to our fingerprinting script files. Upon receiving all the files, extensions installed on the client's browser make appropriate modifications based on their functionalities. These changes only affect the page on the client side and not on the server side. Once this is all complete, the window's load event is fired, which is when our attack program stops and sends the gathered data back to the server.

The tested mobile browsers are Kiwi, Yandex, and Firefox Nightly, and the tested mobile devices are Samsung Galaxy Note 10 5G, OnePlus Nord, OnePlus A6000, which are all Android. We do not consider Chrome because it does not support mobile browser extensions. We mitigate this by using Kiwi and Yandex, two popular Chromium-based mobile browsers that support Chrome extensions. Also, instead of using the default mobile Firefox, which has limited list of extensions currently available, we use Firefox Nightly, a pre-release developer version of Firefox, which allows downloading any extensions from the Mozilla Add-on Store through syncing.

Table 1: List of 6 mobile extensions tested in the main study

Extension	ID'able	Technique Used
360 Internet Protection	**Yes**	CSS
AdBlocker Ultimate	**Yes**	CSS
Avast SafePrice	**Yes**	CSS & DOM
Dark Reader	Sometimes	DOM
DuckDuckGo	**Yes**	CSS
Touch VPN	**Yes**	CSS

The initial extension list is comprised of the top 25 extensions from both Firefox and Chrome web stores at the time of the study, with a total of 50 extensions. After excluding overlapping extensions, those not available in either store, and those deemed non-identifiable based on prior research [12, 22], we narrowed down the list to 16 extensions (Table 3 in Appendix B). Following a preliminary study with Galaxy, we concluded that there are 6 extensions uniquely identifiable in at least one of our tested browsers. The remainder of this paper discusses our main study focusing on these 6 extensions (Table 1).

In our main study, we conducted tests for each device-browser pair. We started with a baseline data point of no extensions installed, which served as our ground truth. Then, we installed each extension individually, recording a data point for each installation. Finally, we captured a data point with all the extensions installed. In total, we analyzed approximately 70 permutations (i.e. users) for our study.

2.2 DOM Extensions

This subsection presents our results regarding Avast SafePrice and Dark Reader, the two identifiable extensions using the DOM technique. Detecting whether these extensions exist or not required mapping the DOM mutations to the particular extension installed at that data point. This process was done manually, since there were only at most 30 DOM mutations observed at any given instance.

Regarding Avast SafePrice, 15 DOM mutations that consistently appeared with its installation were mapped to Avast SafePrice. One mutation, however, changed the fieldset id to different values for every access. Since this variance was a consistent behavior, and not specific to certain device, browser, or extension list, we consider this behavior as expected and perform no additional analysis.

Regarding Dark Reader, DOM mutations were sometimes observed and sometimes not, despite its seemingly working functionality (i.e. the extension inverts bright colors to dark). There was also inconsistency regarding whether such mutations are observed based on the extension list (i.e. in isolation or when all 6 extensions were installed). In addition, the number of observed DOM mutations varied. Sometimes, one MutationRecord was the entire list of changes, while at other times, an entire <script> or <style> tags of 10+ lines were added and removed. Finally, Dark Reader was the only one out of the 6 tested extensions, where there was unexpected data variance across the three times we accessed our page for a given instance to check for validity. This caused difficulty in treating these three accesses as a singular data point as intended, so we do not make further analysis for Dark Reader either.

To summarize, we were able to detect Avast SafePrice extension consistently and Dark Reader sometimes. We do not make any further analysis involving the cross-device or cross-browser comparisons because 1) for Avast SafePrice, we have checked that all mutations are indeed the same across the board (or random for the attribute "fieldset id"), and 2) for Dark Reader, our data was inconsistent for us to derive meaningful observations.

2.3 CSS Extensions

In the case of the CSS technique and the five related extensions (see Table 1), detecting the presence or absence of an extension is achieved by examining the existence of corresponding records in our JSON data. Our attack program captures instances where there is a disparity between the baseline and trigger values of specific div pairs associated with a given extension. If no such records are found, it can be inferred that the extension is not installed.

To make further analysis beyond detecting if this extension is present, this subsection explores various factors that render values different for these extensions. More specifically, we study the interaction of specific browsers, devices, and installed extension lists that changed specific baseline or trigger values, which the attacker can use to infer more information about the user beyond their extension list. The summarized version of results can be found in Table 2, and the detailed version is provided in Table 4 in Appendix B.

2.3.1 Cross-Device. Cross-device comparison is done across the three devices tested to see if device variation results in any differences in either baseline or trigger values. We made pairwise device comparisons (Nord vs. Galaxy, Nord vs. A6000, Galaxy vs. A6000). The resulting table is shown on the left side of Table 2.

The "All" row stands for the data points when all the extensions were installed and tested at the same time to observe any cross effects between CSS extensions. There were no cross effects observed, as indicated by the numbers found in the "All" row, which is simply the sum of all other rows in that column. The denominator of a given row indicates the total number of queried comparison points for cross-device on some specific attribute's value. The numerator indicates how many of these values are actually different for a given pairwise device comparison. More generally, 0 in the numerator means that all the values recorded are the same across the devices.

Our cross-device comparison shows that Galaxy and A6000 devices produce more similar fingerprints with each other than with Nord, as shown by lower percentage in the Galaxy vs. A6000 column compared to the other two columns regardless of the extension (or browser, as indicated in the detailed table in Appendix B). This was surprising at first since Galaxy is produced by Samsung whereas A6000 and Nord are both produced by OnePlus. But upon closer examination, we concluded that most differences were all size related (i.e. the display size for Galaxy Note 10 5G and OnePlus A6000 is both 160mm, whereas it is 164mm for OnePlus Nord). In other words, even if two users have the same extensions downloaded, an attacker can discriminate the two through the information about its device in use, provided by extension modification data. This is an additional component to the simple extension list detection.

2.3.2 Cross-Browser. We made three pairwise cross-browser comparisons (Yandex vs. Kiwi, Yandex vs. Firefox, Kiwi vs. Firefox), as shown on the right side of in Table 2. Since 360 Internet Protection was not compatible on our Firefox Nightly browser, Yandex vs. Kiwi comparison was only done for this extension.

Our cross-browser comparison shows that Yandex and Kiwi browsers produce more similar fingerprints with each other than with Firefox. In fact, the fingerprints were identical in the Yandex vs. Kiwi comparison, as indicated by 0 in all the numerators. This was expected since Yandex and Kiwi are Chromium-based and use the same extensions from the Chrome Web Store, whereas Firefox has its own Firefox Add-On Store.

A more surprising observation is that Yandex vs. Firefox and Kiwi vs. Firefox have the same numerator values but not the same denominator values (e.g. 244 for the numerator but 9032 and 9183 for the denominator in the "All" row, respectively). In other words, the number of values that were different in the two comparisons are the same, but the number of all recorded changes are in fact different. Combined with our previous observation, this indicates that although Yandex and Kiwi may have the same values for all the changed attributes that they share, the actual list of attributes that are being changed in one browser might not be identical to the list in the other. For example, an attribute that was changed in Kiwi might not have changed and thus not recorded in Yandex. This insight allows us to conclude that even with the same extensions that share the same modification values across same-family browsers, we can still discriminate users based on the browsers used, by observing the full list of attributes that were changed by the extensions.

3 DISCUSSION

3.1 Extension Fingerprintability & Cross-Effects

Through our DOM technique, we show that an attacker can utilize the MutationObserver in a mobile context to record and map a set of mutations and consequently extract the list of installed extensions from users. We also show that the existing CSS technique is also applicable in a mobile context, in that by using the same elements from the original dataset tested in a desktop setting [12], we could fingerprint some of the corresponding mobile extensions as well.

Regarding the cross effects of extensions, we indicated that we observed none when we examined CSS extensions and DOM extensions separately. When we installed all the extensions together without distinguishing the relevant technique, however, we actually discovered that the presence of an extension that is not tracked can still be inferred by the values for other extensions. For example, Dark Reader, which was tracked by our DOM technique, affected the baseline values of numerous div pairs for CSS related extensions by inverting the colors (e.g. baseline value is rgb(0,0,0) without Dark Reader and rgb(232, 230, 227) with Dark Reader). Through this, we observed a cross-effect between a DOM extension and other CSS extensions, where we could detect the existence of Dark Reader by examining the baseline values from CSS extensions, even without a specific script that detects Dark Reader. This demonstrates that even if a certain extension is not detected explicitly, it could be inferred by closely examining the collected data, which would provide more discrimination between two users who might seemingly have the same extension list at a first glance.

Table 2: An overview of percentage differences of Pairwise Comparisons (Cross-Device and Cross-Browser) based on CSS technique for 5 extensions under consideration. The "All" row represents the case with all five extensions installed.

	Cross-Device			Cross-Browser		
Per Extension	Nord vs. Galaxy	Nord vs. A6000	Galaxy vs. A6000	Yandex vs. Kiwi	Yandex vs. Firefox	Kiwi vs. Firefox
AdBlocker	0/114	0/114	0/114	0/105	0/98	0/105
DuckDuckGo	0/6	0/6	0/6	0/6	0/6	0/6
Avast SafePrice	48/8498 (0.56%)	48/8498 (0.56%)	0/8498	0/8298	219/8004 (2.74%)	219/8130 (2.69%)
360 Internet	0/560	0/560	0/560	0/816	-	-
Touch VPN	9/958 (0.93%)	9/958 (0.93%)	1/958 (0.1%)	0/942	25/924 (2.7%)	25/942 (2.65%)
All	57/10136 (0.56%)	57/10136 (0.56%)	1/10136 (0.01%)	0/10167	244/9032 (2.7%)	244/9183 (2.66%)

3.2 Interaction of Factors and Values

Percentages of differences in cross-browser and cross-device comparisons are low, but if we use them to represent different users (e.g. two users with the same extension list and browser but on different devices), we can see how one instance of difference in values can help an attacker discriminate users.

For example, we can see from the "Per Browser" section in Table 4 in Appendix B that all the cross-device differences are only visible in Firefox only. To simply put, this tells us that the browser is a crucial interacting factor in whether the attacker can get identifiable information about the device. This specific insight calls attention to how variable interaction can make user information visible.

From the "Per Device" section in the same table, we also see similar observations we made in Section 2.3.1, where the tested device impacts the results of the cross-browser comparison of interest. Here, we see again that Galaxy and A6000 report similar percentages, despite that they are manufactured by different companies.

Overall, by studying the interplay of variables for any pairwise comparisons, we are able to further distinguish user data points beyond extension lists and show that we can obtain more information about the unique device or browser they are using directly from their extension modification data.

4 RELATED WORK

Multiple studies demonstrate the threat of browser fingerprinting and extension fingerprinting. Laperdrix et al. [11] and Vatsel et al. [24] have shown the uniqueness and long-term trackability of browser fingerprints. Pugliese et al [15] conducted a 3-year longitudinal user study on browser fingerprinting. Van Goethem and Joosen [4] and Cao et al. [2] demonstrate cross-session or cross-browser fingerprinting capabilities. Acar et al. [1] and Nikiforakis et al. [14] reveal the exploitation of browser extensions for fingerprinting. Chen and Kapravelos [3] identify privacy-leaking extensions with over 60 million users combined. Gulyas et al [5] uncover users' behavioral uniqueness with detectable extensions. Sjösten et al. [17] utilize Web Accessible Resources to determine extension presence non-behaviorally. Sanchez-Rola et al. [16] enumerate extensions through a timing side-channel attack against access control settings. Weissbacher et al. [25] introduce a tool to detect extensions that steal browser history, utilizing network traffic analysis.

More recently, Solomos et al. [20] introduce continuous fingerprinting, a technique that observes extension modifications throughout their lifecycle, increasing the coverage of identifiable extensions by 66.9%. In a separate work, Solomos et al. [19] emphasize the importance of extension fingerprinting for tracking functionalities triggered by user inputs, successfully fingerprinting around 5,000 unique extensions, including previously undetectable ones. Lin et al. [13] demonstrate that extension fingerprinting is possible without relying on JavaScript APIs, achieving comparable results to previous approaches. These works highlight the evolving nature of extension fingerprinting and the persistent threat it poses.

As for countermeasures, Sjösten et al. [18] propose a whitelist-based mechanism to control extension access for probing and content injection. Starov et al. [21] investigate the concept of extension bloat, referring to page modifications unrelated to extension functionality. They demonstrate that 5.7% of studied extensions were unnecessarily identifiable and propose an access control mechanism to tackle the issue.

5 CONCLUSIONS AND FUTURE WORK

This study presents the results of our extension fingerprinting attack, combining two behavioral techniques and applying them to mobile devices. In doing so, we try to both address a research direction that has not been explored before and examine the extent of data granularity that can be observed through our method. This paper shows that performing extension fingerprinting on mobile devices is not only a viable but also an important research direction in the growing realm of mobile web experiences.

Whereas our selection of configurations is satisfactory for the purpose of demonstrating feasibility of extension fingerprinting on mobile browsers, future research can expand the scope of our study by including a wider range of browsers, devices, operating systems, and extensions. This would provide a more comprehensive understanding of extension fingerprinting in diverse environments. In this regard, conducting a user study within a mobile context would be a valuable next step. Such a study would not only allow for the collection of data from varied configurations, but also facilitate testing the efficacy of the attack in a real-world setting.

Furthermore, testing our attack program against the state-of-the-art countermeasures that can protect susceptible extensions from being fingerprinted [9, 23] or can detect fingerprinting scripts [6] will further validate our experimental results. This is an interesting direction for future work, to pioneer in investigating how fingerprinting countermeasures hold in a mobile context.

REFERENCES

[1] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. 2013. FPDetective: Dusting the Web for Fingerprinters. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security* (Berlin, Germany) *(CCS '13)*. Association for Computing Machinery, New York, NY, USA, 1129–1140. https://doi.org/10.1145/2508859.2516674

[2] Yinzhi Cao, Song Li, and Erik Wijmans. 2017. (Cross-)Browser Fingerprinting via OS and Hardware Level Features. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26-March 1, 2017*. The Internet Society.

[3] Quan Chen and Alexandros Kapravelos. 2018. Mystique: Uncovering Information Leakage from Browser Extensions. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (Toronto, Canada) *(CCS '18)*. Association for Computing Machinery, New York, NY, USA, 1687–1700. https://doi.org/10.1145/3243734.3243823

[4] Tom Van Goethem and Wouter Joosen. 2017. One Side-Channel to Bring Them All and in the Darkness Bind Them: Associating Isolated Browsing Sessions. In *11th USENIX Workshop on Offensive Technologies (WOOT 17)*. USENIX Association, Vancouver, BC. https://www.usenix.org/conference/woot17/workshop-program/presentation/van-goethem

[5] Gabor Gyorgy Gulyas, Doliere Francis Some, Nataliia Bielova, and Claude Castelluccia. 2018. To Extend or Not to Extend: On the Uniqueness of Browser Extensions and Web Logins. In *Proceedings of the 2018 Workshop on Privacy in the Electronic Society* (Toronto, Canada) *(WPES'18)*. Association for Computing Machinery, New York, NY, USA, 14–27. https://doi.org/10.1145/3267323.3268959

[6] Umar Iqbal, Steven Englehardt, and Zubair Shafiq. 2021. Fingerprinting the Fingerprinters: Learning to Detect Browser Fingerprinting Behaviors. In *2021 IEEE Symposium on Security and Privacy (SP)*.

[7] Alexandros Kapravelos, Chris Grier, Neha Chachra, Christopher Kruegel, Giovanni Vigna, and Vern Paxson. 2014. Hulk: Eliciting Malicious Behavior in Browser Extensions. In *23rd USENIX Security Symposium (USENIX Security 14)*. USENIX Association, San Diego, CA, 641–654. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/kapravelos

[8] Soroush Karami, Panagiotis Ilia, Konstantinos Solomos, and Jason Polakis. 2020. Carnus: Exploring the Privacy Threats of Browser Extension Fingerprinting. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society.

[9] Soroush Karami, Faezeh Kalantari, Mehrnoosh Zaeifi, Xavier J. Maso, Erik Trickel, Panagiotis Ilia, Yan Shoshitaishvili, Adam Doupé, and Jason Polakis. 2022. Unleash the Simulacrum: Shifting Browser Realities for Robust Extension-Fingerprinting Prevention. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 735–752. https://www.usenix.org/conference/usenixsecurity22/presentation/karami

[10] Pierre Laperdrix, Nataliia Bielova, Benoit Baudry, and Gildas Avoine. 2020. Browser Fingerprinting: A Survey. *ACM Trans. Web* 14, 2, Article 8 (April 2020), 33 pages. https://doi.org/10.1145/3386040

[11] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. 2016. Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints. In *2016 IEEE Symposium on Security and Privacy (SP)*. 878–894. https://doi.org/10.1109/SP.2016.57

[12] Pierre Laperdrix, Oleksii Starov, Quan Chen, Alexandros Kapravelos, and Nick Nikiforakis. 2021. Fingerprinting in Style: Detecting Browser Extensions via Injected Style Sheets. In *30th USENIX Security Symposium*. Online, France. https://hal.archives-ouvertes.fr/hal-03152176

[13] X. Lin, F. Araujo, T. Taylor, J. Jang, and J. Polakis. 2023. Fashion Faux Pas: Implicit Stylistic Fingerprints for Bypassing Browsers' Anti-Fingerprinting Defenses. In *2023 2023 IEEE Symposium on Security and Privacy (SP) (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 1640–1657. https://doi.org/10.1109/SP46215.2023.00094

[14] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. 2013. Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting. In *2013 IEEE Symposium on Security and Privacy*. 541–555. https://doi.org/10.1109/SP.2013.43

[15] Gaston Pugliese, Christian Riess, Freya Gassmann, and Zinaida Benenson. 2020. Long-Term Observation on Browser Fingerprinting: Users' Trackability and Perspective. *Proceedings on Privacy Enhancing Technologies* 2020 (05 2020), 558–577. https://doi.org/10.2478/popets-2020-0041

[16] Iskander Sanchez-Rola, Igor Santos, and Davide Balzarotti. 2017. Extension Breakdown: Security Analysis of Browsers Extension Resources Control Policies. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 679–694. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/sanchez-rola

[17] Alexander Sjösten, Steven Van Acker, and Andrei Sabelfeld. 2017. 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) *(CODASPY '17)*. Association for Computing Machinery, New York, NY, USA, 329–336. https://doi.org/10.1145/3029806.3029820

[18] Alexander Sjösten, Steven Acker, Pablo Picazo-Sanchez, and Andrei Sabelfeld. 2019. Latex Gloves: Protecting Browser Extensions from Probing and Revelation Attacks. https://doi.org/10.14722/ndss.2019.23309

[19] Konstantinos Solomos, Panagiotis Ilia, Soroush Karami, Nick Nikiforakis, and Jason Polakis. 2022. The Dangers of Human Touch: Fingerprinting Browser Extensions through User Actions. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 717–733. https://www.usenix.org/conference/usenixsecurity22/presentation/solomos

[20] Konstantinos Solomos, Panagiotis Ilia, Nick Nikiforakis, and Jason Polakis. 2022. Escaping the Confines of Time: Continuous Browser Extension Fingerprinting Through Ephemeral Modifications. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (Los Angeles, CA, USA) *(CCS '22)*. Association for Computing Machinery, New York, NY, USA, 2675–2688. https://doi.org/10.1145/3548606.3560576

[21] Oleksii Starov, Pierre Laperdrix, Alexandros Kapravelos, and Nick Nikiforakis. 2019. Unnecessarily Identifiable: Quantifying the Fingerprintability of Browser Extensions Due to Bloat. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 3244–3250. https://doi.org/10.1145/3308558.3313458

[22] O. Starov and N. Nikiforakis. 2017. XHOUND: Quantifying the Fingerprintability of Browser Extensions. In *2017 IEEE Symposium on Security and Privacy (SP)*. 941–956. https://doi.org/10.1109/SP.2017.18

[23] Erik Trickel, Oleksii Starov, Alexandros Kapravelos, Nick Nikiforakis, and Adam Doupé. 2019. Everyone is Different: Client-side Diversification for Defending Against Extension Fingerprinting. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 1679–1696. https://www.usenix.org/conference/usenixsecurity19/presentation/trickel

[24] Antoine Vastel, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. 2018. FP-STALKER: Tracking Browser Fingerprint Evolutions. In *2018 IEEE Symposium on Security and Privacy (SP)*. 728–741. https://doi.org/10.1109/SP.2018.00008

[25] Michael Weissbacher, Enrico Mariconti, Guillermo Suarez-Tangil, Gianluca Stringhini, William Robertson, and Engin Kirda. 2017. Ex-Ray: Detection of History-Leaking Browser Extensions. In *Annual Computer Security Applications Conference (ACSAC)*. event-place: San Juan, Puerto Rico.

APPENDIX

A SAMPLE SCRIPTS

```
{"record1" : "ADD <style>.qc-cmp-showing { visibility:
     hidden !important; } body.didomi-popup-open {
     overflow: auto !important; } #didomi-host {
     visibility: hidden !important; }</style> TO BODY",
 "record2" : "REMOVE <fieldset><div></div></fieldset>
     FROM HTML",
 "record3" : "CHANGE id AT FIELDSET FROM null TO sizzle"}
```

Listing 1: Sample JSON record of DOM mutations by MutationObserver. First is an addition of a style tag to its parent body tag. Second is a removal of a fieldset tag from its parent HTML tag. Third is a change of attribute id on a fieldset tag from the value of null to sizzle.

```
<div class="trigger" id=28913>
  <div trig="no" orig_class="adguard-alert"></div>
  <div trig="yes" class="adguard-alert"></div>
</div>
```

Listing 2: Sample CSS div pair. The top div element with $trig$ = "*no*" is the baseline element that will not be modified by the extension. Its replica on the bottom with $trig$ = "*yes*" serves as the trigger element and can be modified by the extension. If it is modified, we can record the disparity and detect this CSS extension.

B DETAILED TABLES

Table 3: List of 16 mobile extensions tested in the preliminary study. We note here that the two extensions that were incompatible on Firefox Nightly were originally selected as part of the 16 extensions because they are indeed available on the Firefox Web Store. However, in our experiment, we failed to download them on our physical device due to incompatibility. Since our criteria was to test all extensions that were identifiable in at least one browser (rightmost column), we decided to include 360 Internet Protection in that list of 6 extensions (indicated in bold).

Extension	Yandex	Kiwi	Firefox Nightly	ID'able in at least one browser
360 Internet Protection	Available	Available	Not Downloadable	**Yes**
AdBlock	~~Not Available~~	Available	Available	No
Adblock Plus	~~Not Available~~	Available	Available	No
AdBlocker Ultimate	Available	Available	Available	**Yes**
AdGuard AdBlocker	~~Not Available~~	Available	Available	No
Avast SafePrice	Available	Available	Available	**Yes**
Dark Reader	Available	Available	Available	**Yes**
DuckDuckGo Privacy Essentials	Available	Available	Available	**Yes**
Ghostery	Available	Available	Available	No
Grammarly	Available	Available	Available	No
Honey: Automatic Coupons & Cash Back	Available	Available	Available	No
LastPass Password Manager	Available	Available	Not Downloadable	No
Pinterest Save Button	Available	Available	Available	No
Privacy Badger	~~Not Available~~	Available	Available	No
Touch VPN	Available	Available	Available	**Yes**
uBlock Origin	~~Not Available~~	Available	Available	No
Total	**11**	**16**	**14**	**6**

Table 4: A detailed overview of percentage differences of Pairwise Comparisons based on CSS technique. Each of the three sections (i.e. Extension, Browser, and Device) indicates through which variable we want to analyze the cross-device and cross-browser pairwise comparisons. Table 2 in the paper above is equivalent to the Extension section shown here, and the significance of the numbers can be understood in a similar fashion.

		Cross-Device			**Cross-Browser**		
Per		**Nord vs. Galaxy**	**Nord vs. A6000**	**Galaxy vs. A6000**	**Yandex vs. Kiwi**	**Yandex vs. Firefox**	**Kiwi vs. Firefox**
Extension	AdBlocker	0/114	0/114	0/114	0/105	0/98	0/105
	DuckDuckGo	0/6	0/6	0/6	0/6	0/6	0/6
	Avast SafePrice	48/8498 (0.56%)	48/8498 (0.56%)	0/8498	0/8298	219/8004 (2.74%)	219/8130 (2.69%)
	360 Internet	0/560	0/560	0/560	0/816	-	-
	Touch VPN	9/958 (0.93%)	9/958 (0.93%)	1/958 (0.1%)	0/942	25/924 (2.7%)	25/942 (2.65%)
	All	57/10136 (0.56%)	57/10136 (0.56%)	1/10136 (0.01%)	0/10167	244/9032 (2.7%)	244/9183 (2.66%)
Browser	Yandex	0/6892	0/6892	0/6892	-	-	-
	Kiwi	0/6896	0/6896	0/6896	-	-	-
	Firefox	114/6484 (1.76%)	114/6484 (1.76%)	2/6484 (0.03%)	-	-	-
Device	Nord	-	-	-	0/6778	88/6024 (1.46%)	88/6122 (1.44%)
	Galaxy	-	-	-	0/6778	200/6020 (3.32%)	200/6122 (3.27%)
	A6000	-	-	-	0/6778	200/6020 (3.32%)	200/6122 (3.27%)
	Total	114/20272 (0.56%)	114/20272 (0.56%)	2/20272 (0.01%)	0/20334	488/18064 (2.7%)	488/18366 (2.66%)