To Extend or not to Extend: on the Uniqueness of Browser Extensions and Web Logins

Gábor György Gulyás
INRIA
gabor.gulyas@inria.fr

Dolière Francis Somé
INRIA
doliere.some@inria.fr

Nataliia Bielova
INRIA
nataliia.bielova@inria.fr

Claude Castelluccia
INRIA
claude.castelluccia@inria.fr

ABSTRACT

Recent works showed that websites can detect browser extensions that users install and websites they are logged into. This poses significant privacy risks, since extensions and Web logins that reflect user's behavior, can be used to uniquely identify users on the Web. This paper reports on the first large-scale behavioral uniqueness study based on 16,393 users who visited our website. We test and detect the presence of 16,743 Chrome extensions, covering 28% of all free Chrome extensions. We also detect whether the user is connected to 60 different websites.

We analyze how unique users are based on their behavior, and find out that 54.86% of users that have installed at least one detectable extension are unique; 19.53% of users are unique among those who have logged into one or more detectable websites; and 89.23% are unique among users with at least one extension and one login.

We use an advanced fingerprinting algorithm and show that it is possible to identify a user in less than 625 milliseconds by selecting the most unique combinations of extensions.

Because privacy extensions contribute to the uniqueness of users, we study the trade-off between the amount of trackers blocked by such extensions and how unique the users of these extensions are. We have found that privacy extensions should be considered more useful than harmful. The paper concludes with possible countermeasures.

CCS CONCEPTS

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

KEYWORDS

web tracking, uniqueness, anonymity, fingerprinting

ACM Reference Format:
Gábor György Gulyás, Dolière Francis Somé, Nataliia Bielova, and Claude Castelluccia. 2018. To Extend or not to Extend: on the Uniqueness of Browser Extensions and Web Logins. In *2018 Workshop on Privacy in the Electronic Society (WPES'18), October 15, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3267323.3268959

1 INTRODUCTION

In the last decades, researchers have been actively studying users' uniqueness in various fields, in particular biometrics and privacy communities hand-in-hand analyze various characteristics of people, their behavior and the systems they are using. Related research showed that a person can be characterized based on her typing behavior [53, 61], mouse dynamics [52], and interaction with websites [33]. Furthermore, Internet and mobile devices provide rich environment where users' habits and preferences can be automatically detected. Prior works showed that users can be uniquely identified based on websites they visit [50], smartphone apps they install [23] and mobile traces they leave behind them [29].

Since the web browser is the tool people use to navigate through the Web, privacy research community has studied various forms of *browser fingerprinting* [22, 27, 30, 32, 34, 49]. Researchers have shown that a user's browser has a number of "physical" characteristics that can be used to uniquely identify her browser and hence to track it across the Web. Fingerprinting of users' devices is similar to physical biometric traits of people, where only physical characteristics are studied.

Similar to previous demonstrations of user uniqueness based on their behavior [23, 50], *behavioral characteristics*, such as browser settings and the way people use their browsers can also help to uniquely identify Web users. For example, a user installs web browser extensions she prefers, such as AdBlock [1], LastPass [14] or Ghostery [8] to enrich her Web experience. Also, while browsing the Web, she logs into her favorite social networks, such as Gmail [13], Facebook [7] or LinkedIn [15]. In this work, we study *users' uniqueness* based on their behavior and preferences on the Web: we analyze how unique are Web users based on their *browser extensions and logins.*

In recent works, Sjösten et al. [56] and Starov and Nikiforakis [58] explored two complementary techniques to detect extensions. Sánchez-Rola et al. [54] then discovered how to detect any extension via a timing side channel attack. These works were focused on the technical mechanisms to detect extensions, but what was not studied is *how browser extensions contribute to uniqueness of users at large scale.* Linus [46] showed that some social websites are vulnerable to the "login-leak" attack that allows an arbitrary script to detect whether a user is logged into a vulnerable website. However, it was not studied *whether Web logins can also contribute to users' uniqueness.*

In this work, we performed the first large-scale study of user uniqueness based on browser extensions and Web logins, collected from more than 16,000 users who visited our website (see the breakdown in Fig. 6). Our experimental website identifies installed Google Chrome [9] extensions via Web Accessible Resources [56], and detects websites where the user is logged into, by methods that rely on URL redirection and CSP violation reports. Our website is able to detect the presence of 13k Chrome extensions on average per month (the number of detected extensions varied monthly between 12, 164 and 13, 931), covering approximately 28% of all free Chrome extensions[1] . We also detect whether the user is connected to one or more of 60 different websites. Our main contributions are:

- A large scale study on *how unique users are based on their browser extensions and website logins*. We discovered that 54.86% of users that have installed at least one detectable extension are unique; 19.53% of users are unique among those who have logged into one or more detectable websites; and 89.23% are unique among users with at least one extension and one login. Moreover, we discover that 22.98% of users could be uniquely identified by Web logins, even if they disable JavaScript.
- We study the privacy dilemma on Adblock and privacy extensions, that is, *how well these extensions protect their users against trackers and how they also contribute to uniqueness*. We evaluate the statement "the more privacy extensions you install, the more unique you are" by analyzing how users' uniqueness increases with the number of privacy extensions they install; and by evaluating the tradeoff between the privacy gain of the blocking extensions such as Ghostery [8] and Privacy Badger [17].

We furthermore show that browser extensions and Web logins can be exploited to fingerprint and track users by only checking a limited number of extensions and Web logins. We have applied an advanced fingerprinting algorithm [38] that carefully selects a limited number of extensions and logins. For example, Figure 1 shows the uniqueness of users we achieve by testing a limited number of extensions. The last column shows that 54.86% of users are unique based on all 16,743 detectable extensions. However, by testing 485 carefully chosen extensions we can identify more than 53.96% of users. Besides, detecting 485 extensions takes only 625ms.

Finally, we give suggestions to the end users as well as website owners and browser vendors on how to protect the users from the fingerprinting based on extensions and logins.

In our study we did not have enough data to make any claims about the stability of the browser extensions and web logins because only few users repeated an experiment on our website (to be precise, only 66 users out of 16,393 users have made more than 4 tests on our website). We leave this as a future work.

2 BACKGROUND

2.1 Detection of browser extensions

A *browser extension* is a program, typically written in JavaScript, HTML and CSS. An extension may alter the context of the webpage, intercept HTTP requests, or provide extra functionality. Examples

[1]The list of detected extensions and websites are available on our website: https://extensions.inrialpes.fr/faq.php

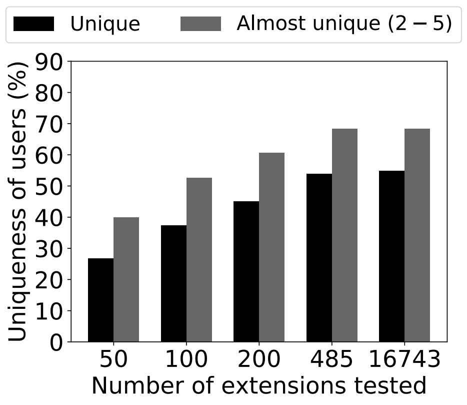


Figure 1: Results of general fingerprinting algorithm. Testing 485 carefully selected extensions provides a very similar uniqueness result to testing all 16,743 extensions. Almost unique means that there are 2–5 users with the same fingerprint.

of browser extensions are ad blockers, such as AdBlock [1] and password managers, such as LastPass [14].

In the Google Chrome web browser, each extension comes with *a manifest file* [11], which contains metadata about the extension. Each extension has a unique and permanent identifier, and the manifest file of an extension with identifier `extID` is located at `chrome-extension://[extID]/manifest.json`. The manifest file has a section *web_accessible_resources* (WARs) that declares which resources of an extension are accessible in the content of any webpage [10]. The WARs section specifies a list of paths to such resources, presented by the following type of URL:
`chrome-extension://[extID]/[path]`, where `path` is the path to the resource in the extension.

Therefore, a script that tries to load such an accessible resource in the context of an arbitrary webpage is able to check whether an extension is installed with a 100% guarantee: if the resource is loaded, an extension is installed, otherwise it is not. Figure 2 shows an example of AdBlock extension detection: the script tries to load an image, which is declared in the *web_accessible_resources* section of AdBlock's manifest file. If the image from AdBlock, located at `chrome-extension://[AdBlockID]/icons/icons24.png` is successfully loaded, then AdBlock is installed in the user's browser.

Sjösten et al. [56] were the first to crawl the Google Chrome Web Store and to discover that 28% of all free Chrome extensions are detectable by WARs. An alternative method to detect extensions that was available at the beginning of our experiment, was a behavioral method from XHOUND [58], but it had a number of false positives and detected only 9.2% of top 10k extensions (see Sec. 9 for more details). Therefore, we decided to reuse the code from Sjösten et al. [56] with their permission to crawl Chrome Web Store and identify detectable extensions based on WARs. During our experiment, we discovered that WARs could be detected in other Chromium-based browsers like Opera [16] and the Brave Browser [3] (we could even detect Brave Browser since it is shipped with several default extensions detectable by WARs). We have chosen to work with Chrome, as it was the most affected.

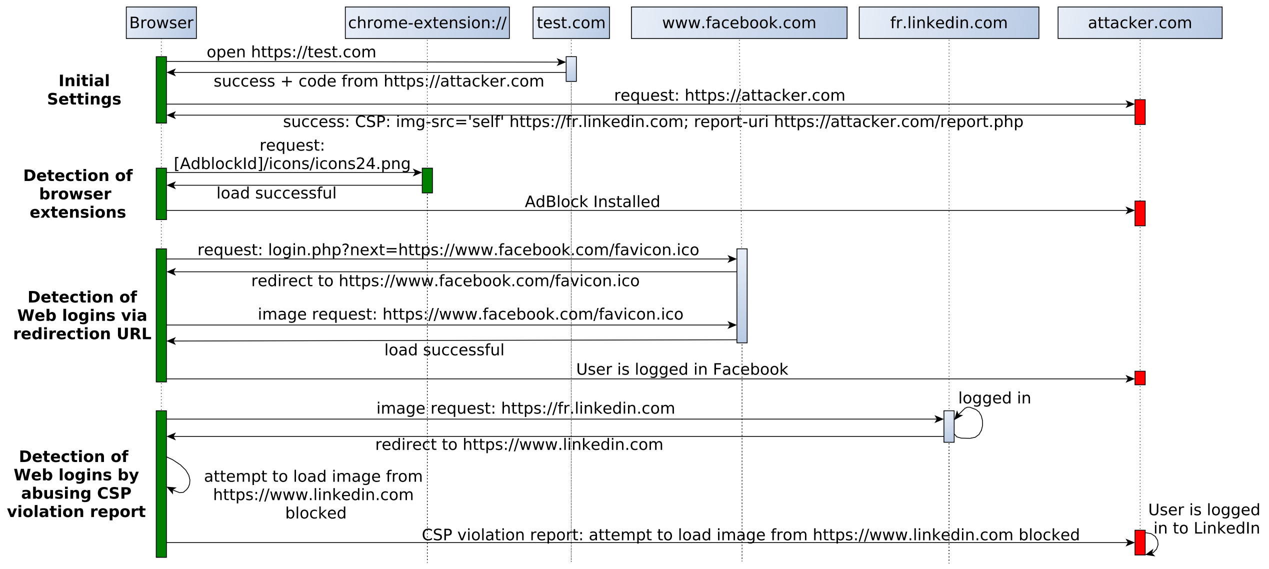

Browser
chrome-extension://
test.com
www.facebook.com
fr.linkedin.com
attacker.com
Initial Settings
open https://test.com
success + code from https://attacker.com
request: https://attacker.com
success: CSP: img-src='self' https://fr.linkedin.com; report-uri https://attacker.com/report.php
Detection of browser extensions
request: [AdblockId]/icons/icons24.png
load successful
AdBlock Installed
Detection of Web logins via redirection URL
request: login.php?next=https://www.facebook.com/favicon.ico
redirect to https://www.facebook.com/favicon.ico
image request: https://www.facebook.com/favicon.ico
load successful
User is logged in Facebook
Detection of Web logins by abusing CSP violation report
image request: https://fr.linkedin.com
logged in
redirect to https://www.linkedin.com
attempt to load image from https://www.linkedin.com blocked
CSP violation report: attempt to load image from https://www.linkedin.com blocked
User is logged in to LinkedIn

Figure 2: Detection of browser extensions and Web logins. A user visits a benign website test.com which embeds third party code (the attacker' script) from attacker.com. The script detects an icon of `Adblock` extension and concludes that `Adblock` is installed. Then the script detects that the user is logged into Facebook when it successfully loads Facebook `favicon.ico`. It also detects that the user is logged into LinkdedIn through a CSP violation report triggered because of a redirection from https://fr.linkedin.com to https://www.linkedin.com. All the detection of extensions and logins are invisible to the user.

2.2 Detection of Web logins

In general, a website cannot detect whether a user is logged into other websites because of Web browser security mechanisms, such as access control and Same-Origin Policy [18]. In this section, we present two advanced methods that, despite browser security mechanisms, allow an attacker to detect the websites where the user is logged into. Figure 2 presents all the detection mechanisms.

Redirection URL hijacking. The first requirement for this method to work is the login redirection mechanism: when a user is not logged into Facebook, and tries to access an internal Facebook resource, she automatically gets redirected to the URL http://www.facebook.com/login.php?next=[path], where `path` is the path to the resource. The second requirement is that the website should have an internal image available to all the users. In the case of Facebook, it is a `favicon.ico` image.

By dynamically embedding an image pointing to https://www.facebook.com/login.php?next=https%3A%2F%2Fwww.facebook.com%2Ffavicon.ico into the webpage, an attacker can detect whether the user is logged into Facebook or not. If the image loads, then the user is logged into Facebook, otherwise she is not. This method has been shown to successfully detect logins on dozens of websites [46].

Abusing CSP violation reporting. Content-Security-Policy (CSP) [5, 57] is a security mechanism that allows programmers to control which client-side resources can be loaded and executed by the browser. CSP (version 2) is an official W3C candidate recommendation [60], and is currently supported by major web browsers.

A CSP delivered with a page controls the resources of the page. For example, CSP can set a rule to allow the browser to load images only from a particular domain. If a webpage tries to load an image from a different domain, CSP will block such request and can send a violation report back to the web server.

An attacker can misuse CSP to detect redirections [41]. We extend this idea to detect logins. For this method to work, a website should redirect its logged in users to a different domain. In the case of LinkedIn, the users, who are not logged in, visit `fr.linkedin.com`, while the users, who are logged in, are automatically redirected to a different domain `www.linkedin.com`. The lowest block of Fig. 2 presents an example of such attack on LinkedIn. Initially, the attacker embeds a hidden iframe from his own domain with the CSP that restricts loading images only from `fr.linkedin.com`. Then, the attacker dynamically embeds a new image on the testing website, pointing to `fr.linkedin.com`. If the user is logged in, LinkedIn will redirect her to the `www.linkedin.com`, and thus the browser will fire a CSP violation report because images can be loaded only from `fr.linkedin.com`. By receiving the CSP report, the attacker deduces that the user is logged in LinkedIn.

3 DATASET

We launched an experiment website in April 2017 to collect browser extensions and Web logins with the goal of studying users' uniqueness at a large scale. We have advertised our experiment by all possible means, including social media and in press. In this section, we first present the set of attributes that we collect in our experiment and the rules we applied to filter out irrelevant records. Then, we provide data statistics and show which extensions and logins are popular among our users.

3.1 Experiment website and data collection

The goal of our website is both to collect browser extensions and Web logins, and to inform users about privacy implications of this particular type of fingerprinting. Using the various detection techniques described in Section 2, we collect the following attributes:

- The list of installed browser extensions, using web accessible resources. For each user we tested around 13k extensions detectable at the moment of testing (see Figure 3).
- The list of Web logins: we test for 44 logins using redirection URL hijacking and 16 logins using CSP violation report.
- Standard fingerprinting attributes [45], such as fonts installed, Canvas fingerprint [21], and WebGL [48]. To collect these attributes, we use FingerprintJS2, which is an open-source browser fingerprinting library [39]. We collected these attributes in order to clean our data and compare entropy with other studies (see Table 3).

To recognize users that perform several tests on our website, we have stored a unique identifier for each user in the HTML5 localStorage. We have communicated our website via forums and social media channels related to science and technology, and got press coverage in 3 newspapers. We have collected 22,904 experiments performed by 19,814 users between April and August 2017.

Ethical concerns. Our study was validated by an IRB-equivalent service at our institution. All visitors are informed of our goal, and are provided with both Privacy Policy and FAQ sections of the website. The visitors have to explicitly click on a button to trigger the collection of their browser attributes. In our Privacy Policy, we explain what data we are collecting, and give a possibility to opt-out of our experiment. The data collected is used only for our own research, will be held until December 2019 and will not be shared with anyone.

Table 1: Users filtered out of the final dataset

Initial users	**19,814**
Mobile browser users	1,042
Chrome browser users with extension detection error	6
Non Chrome users with at least one extension detected	261
Brave browser users	31
Users whose browser has an empty user-agent string, screen resolution, fonts, or canvas fingerprint	2,015
Users with more than 4 experiments	66
Final dataset	**16,393**
Chrome browser users in the final dataset	7,643

Data cleaning. We applied a set of cleaning rules over our initial data, to improve the quality of the data. The final dataset contains 16,393 valid experiments (one per user). Table 1 shows the initial number of users and which users have been removed from our initial dataset. We have removed all 1,042 users with mobile browsers. At the time of writing this paper, browser extensions were not supported on Chrome for mobiles. Since extensions detection were designed for Chrome, we then excluded mobile browsers. Moreover, mobile users tend to prefer native apps rather than their web versions[2]. In fact, the popular logins in our dataset, such as Gmail, Facebook, Youtube, all have a native mobile version.

We have also removed 2,015 users that have deliberately tampered with their browsers: for example, users with empty user-agent string, empty screen resolution or canvas fingerprint. We think that it is reasonable not to trust information received from those users, as they may have tampered with it. We also needed this information to compare our study with previous works on browser fingerprinting. Finally, we have excluded users who have tampered with extension detection. This includes Chrome users for whom extension detection did not successfully complete, and users of other browsers with at least 1 extension detected.

For users who visited our website and performed up to 4 experiments, we kept only one experiment, the one with the biggest number of extensions and logins. We then removed 66 users with more than 4 experiments. We suspect that the goal of such users with numerous experiments was just to use our website in order to test the uniqueness of their browsers with different browser settings.

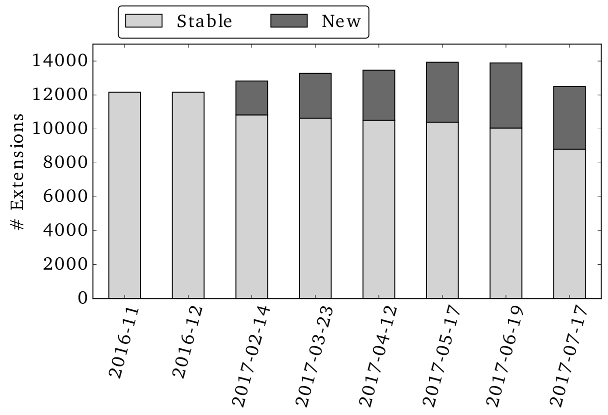


Figure 3: Evolution of detected extensions in Chrome

Evolution of browser extensions. From November 2016 to July 2017, we crawled on a montly basis the free extensions on the Chrome Web Store in order to keep an up-to-date set of extensions for our experiment. Figure 3 presents the evolution of extensions throughout the period of our experiment. Since some extensions got removed from the Chrome Web Store, the number of stable extensions decreased.

Out of 12,164 extensions that were detectable in November 2016, 8,810 extensions (72.4%) remained stable throughout the 9-months-long experiment. In total, 16,743 extensions were detected at some point during these 9 months. Since every month the number of detectable extensions was different, on average we have tested around 13k extensions during each month.

3.2 Data statistics

Our study is the first to analyze uniqueness of users based on their browser extensions and logins at large scale. Only uniqueness based on browser extensions was previously measured, but on very

[2]https://jmango360.com/wiki/mobile-app-vs-mobile-website-statistics/

Table 2: Previous studies on measuring uniqueness based on browser extensions and our estimation of uniqueness.

Study	Fingerprints collected in a study	Extensions targeted in a study	Unique finger-prints in a study	Unique fingerprints in our dataset
Timing leaks [54]	204	2,000	56.86%	55.64%
XHOUND [58]	854	1,656	14.10%	49.60%
Ours	7,643	13k	39.29%	39.29%

small datasets of 204 [54] and 854 [58] participants. We measure uniqueness of 16,393 users for all attributes, and of 7,643 Chrome browser users for browser extensions.

Comparison to previous studies. To compare our findings with the previous works on browser extensions, we randomly pick subsets of 204 (as in [54]) and 854 (as in [58]) Chrome users 100 times (we found that picking 100 times provided a stable result). Table 2 shows uniqueness results from previous works and an estimated uniqueness using our dataset.

The last column in Table 2 shows our evaluation of uniqueness for a given subset of users. Our estimation for 204 random users is 55.64%, which is close to the 56.86% from the original study [54]. For 854 random users, we estimate that 49.60% of them are unique, while in the original XHOUND study [58] the percentage of unique users is only 14.1%. We think that such small percentage of unique users in [58] is due to (1) a smaller number of extensions detected (only 174 extensions were detected for 854 users); (2) a different user base: while our experiments and [54] targeted colleagues, students and other likely privacy-aware experts, XHOUND [58] used Amazon Mechanical Turk, where users probably have different habits to installing extensions. Out of 7,643 users of the Chrome browser, where we detected extensions, 39.29% of users were unique. This number shows a more realistic estimation of users' uniqueness based on browser extensions than previous works because of a significantly larger dataset.

To the best of our knowledge, our study is the first to *analyze uniqueness of users based on their web logins*, and on combination of extensions and logins.

Normalized Shannon's entropy. We compare our dataset with the previous studies on browser fingerprinting: AmIUnique [44, Table B.3] (contains 390,410 fingerprints, collected between November 2014 and June 2017) and Hiding in the Crowd [34] (contains 1,816,776 users collected in 2017). Entropy measures the amount of identifying information in a fingerprint – the higher the entropy is, the more unique and identifiable a fingerprint will be. To compare with previous datasets, which are of different sizes, we compute normalized Shannon's entropy:

$$H_N(X) = \frac{H(X)}{\log_2 N} = -\frac{1}{\log_2 N} \cdot \sum_i P(x_i) \log_2 P(x_i) \qquad (1)$$

where X is a discrete random variable with possible values $\{x_1, ..., x_n\}$, $P(X)$ is a probability mass function and N is the size of the dataset.

Table 3: Normalized entropy of extensions and logins compared to previous studies.

Standard fingerprinting studies			
Attribute	Ours	AmIUnique [44]	Hiding [34] Desktop
User Agent	0.474	0.601	0.304
List of Plugins	0.343	0.523	0.494
Timezone	0.168	0.187	0.005
Screen Resolution	0.271	0.276	0.213
List of Fonts	0.652	0.370	0.335
Canvas	0.611	0.503	0.387
Studies on extensions and logins			
Attribute	Ours	Timing leaks [54]	XHOUND [58]
Extensions	0.641	0.869	0.437
Logins	0.441	N/A	N/A

Table 3 compares the entropy values of well-known attributes for standard fingerprinting and for logins for all 16,393 users in our dataset and for browser extensions for 7,643 Chrome users. All the attributes in standard fingerprinting are similar to previous works, except for fonts and plugins. Unsurprisingly, plugins entropy is very small because of decreasing support of plugins in Firefox [51] and Chrome [55]. Differently from previous studies that detected fonts with Flash, we used JavaScript based font detection, relying on a list of 500 fonts shipped along with the FingerprintJS2 library. As those fonts are selected for fingerprinting, this could explain why our list of fonts provides a very high entropy.

In our dataset, as well as in previous studies, browser extensions are one of the most discriminating attributes of a user's browser. The computed entropy of 0.641, computed for the 7,643 Chrome users, lays between the findings of Timing leaks [54] and XHOUND [58]. One possible explanation is the size of the user base. For instance, users in XHOUND had few and probably often the same extensions detected (out of 1,656 targeted extensions, only 174 were detected for 854 users), making only 14.1% of them unique. This explains why the entropy in XHOUND is smaller. Sánchez-Rola et al. [54] computed a very high entropy, but on a very small dataset of 204 users: 116 of them had a unique set of installed extensions, and thus the computed entropy was very high.

3.3 Usage of extensions and logins

Figure 4 shows the distribution of users in our dataset according to the number of detected extensions and logins (users having between 1 and 13 logins or extensions detected), and the number of unique users as they are grouped by number of detected extensions and logins. The maximum number of extensions we detected for a single user was 33. The number of users decreases with the number of extensions. The largest group of users have only 1 extension detected, followed by users with 2 detected extensions, etc. We notice that the more extensions a user has, the more unique she is. We analyze this phenomenon further in Section 4.2. Among users with exactly 1 extension detected, 7.39% are unique. This percentage rises to 45.35% and 85.89% for groups of users with exactly 2 and 3 detected extensions respectively.

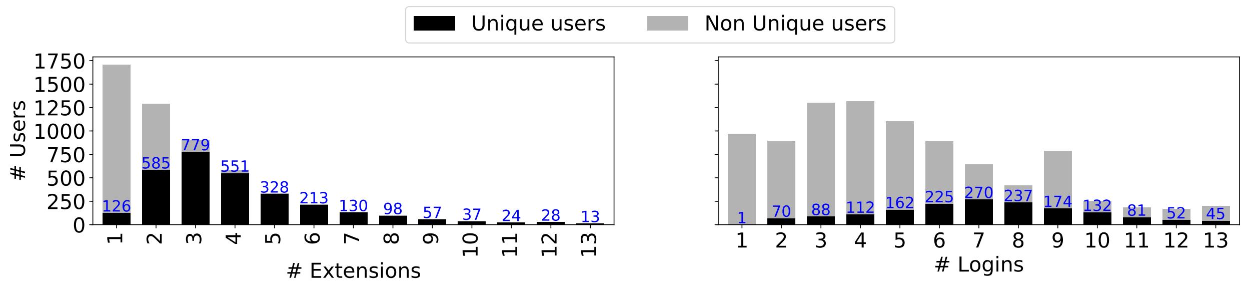


Figure 4: Usage of browser extensions and logins by all users.

Figure 4 also shows the distribution of users per number of detected logins. We found that most users have between 1 and 10 logins, with a maximum number of 40 logins detected for one user. On our website, we were able to detect the presence of 60 logins, which is rather small with respect to the large number of extensions we tested (around 13k per user). This explains why fewer users are unique based on their logins: for example, among users with exactly 1 login detected, 0.10% are unique, and 7.82% are unique among users with exactly 2 logins detected.

Table 4: Top seven most popular extensions in our dataset and their popularity on Chrome Web Store

Extension	Dataset	Chrome
AdBlock	1,557	10,000,000+
LastPass: Free Password Manager	1,081	7,297,730
Ghostery	735	2,665,427
Privacy Badger	594	771,804
Adobe Acrobat	585	10,000,000+
Cisco WebEx Extension	482	10,000,000+
Save to Pocket	428	2,752,642

Table 5: Top seven most popular logins in our dataset and their ranking according to Alexa

Website	Dataset	Alexa Rank
Gmail (subdomain of Google)	6,828	1
Youtube	6,780	2
Facebook	5,493	3
LinkedIn	3,913	13
Blogger	3,393	53
Twitter	3,274	8
eBay.com	2,220	33

What extensions are the most popular among our users? Table 4 presents the seven most detected extensions in our dataset of 16,393 users. The three most popular extensions are AdBlock [1], password manager LastPass [14] and tracker blocker Ghostery [8]. These extensions are also very popular according to their downloads statistics on Chrome Web Store.

What websites users are logging into the most? Table 5 shows the seven most detected websites in our experiment. These websites are also highly rated according to Alexa[3]. For instance, Google [12], Facebook [7] and Youtube [20] are regularly ranked as the top 3 most popular websites by Alexa[4]. Being able to detect such popular websites further strengthen our study as they represent websites that are widely used by users in the wild.

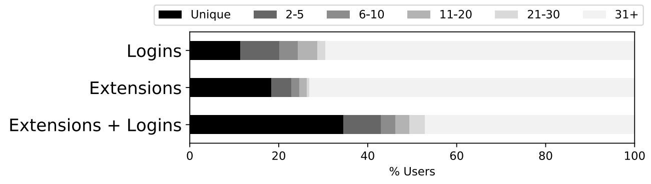


Figure 5: Distribution of anonymity set sizes for 16,393 users based on detected extensions and logins.

4 UNIQUENESS ANALYSIS

In this section we present the results for user's uniqueness based on all 16,743 extensions and 60 logins. We first show uniqueness for the full dataset of 16,393 users, and then present more specific results for various subsets of our dataset.

Uniqueness results for the full dataset. Figure 5 shows the uniqueness of users according to their extensions and logins, and a combination of both attributes. Out of the 16,393 users, 11.30% are unique based on their logins. For 42.1% of users in our dataset, we did not detect any logins. These users either did not log into any of the 60 websites we could detect or blocked third party cookies, that prevented our login detection from working properly.

Considering only detected extensions, 18.38% of users in our dataset are unique. This result is also influenced by the 66.61% of users who did not have any extension detected: these are either Chrome users with no extensions detected, or users of other browsers.

An attacker willing to fingerprint users can also use their detected logins and extensions combined. Interestingly, by combining extensions and logins, we found that 34.51% of users are uniquely identifiable. It is worth mentioning that 32.61% of users have no extensions and no logins detected. This impacts significantly the computed uniqueness.

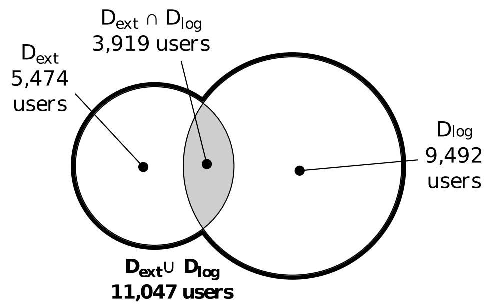


Figure 6: Four final datasets. D_{Ext} contains users, who have installed at least one detected extension and D_{Log} contains users, who have at least one login detected.

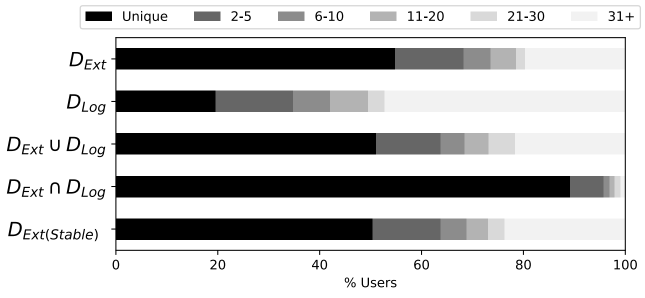


Figure 7: Anonymity sets for different datasets

4.1 Four final datasets

In our full dataset of 16,393 users, we have observed 7,643 users of Chrome browser, for whom testing of browser extensions worked properly. In this subsection we consider various subsets of our full dataset that demonstrate uniqueness results for users who have at least one extension or one login detected. Figure 6 shows four final datasets that we further analyze in this section:

- D_{Ext} contains 5,474 Chrome users, who have installed at least one extension that we can detect.
- D_{Log} contains 9,492 users, who have logged into at least one website that we detect.
- $D_{Ext} \cap D_{Log}$ contains 3,919 Chrome users who have at least one extension and one login detected.
- $D_{Ext} \cup D_{Log}$ contains 11,047 users who have either at least one extension or at least one login detected.

4.2 Uniqueness results for final datasets

Figure 7 presents results for the four datasets. D_{Ext} dataset shows that 54.86% of users are uniquely identifiable among Chrome users, who have at least one detectable extension. This demonstrates that browser extensions detection is a serious privacy threat as a fingerprinting technique.

Among 9,492 users with at least one login detected (D_{Log} dataset) only 19.53% are uniquely identifiable. This result can be explained by a very small diversity of attributes (only 60 websites).

When we analyzed Chrome users who have at least one extension and one login detected ($D_{Ext} \cap D_{Log}$ dataset), we found out that 89.23% of them are uniquely identifiable. This means that without any other fingerprinting attributes, the mere installation of at least one extension, in addition to being logged into at least one website imply that the majority of users in this dataset can be tracked by their fingerprint based solely on extensions and logins!

Furthermore, for dataset $D_{Ext} \cup D_{Log}$ that contains users with at least one extension or at least one login, we compute that 51.15% of users can be uniquely identified. This result becomes particularly interesting when we compare the size of the $D_{Ext} \cup D_{Log}$ dataset, which contains 11,047 users, with the size of the D_{Ext} dataset, that has 5,474 users. The size of $D_{Ext} \cup D_{Log}$ is almost twice as large as D_{Ext}. Nevertheless, the percentage of unique users and the distribution of anonymity set sizes in these datasets are very similar: 54.86% of unique users in D_{Ext} and 51.15% of unique users in $D_{Ext} \cup D_{Log}$. We believe this is due to the fact that extensions and logins are orthogonal properties. We checked the cosine similarity between these attributes as binary vectors, and found that all attribute pairs had a very low similarity score, all below 0.34, with 11 exceptions below 0.2.

The last row $D_{Ext(Stable)}$ shows uniqueness of users in the D_{Ext} dataset, but considering only stable extensions (see more details in Section 3). Interestingly, 50.35% of users are uniquely identifiable with their stable extensions only and the distribution of anonymity set sizes is very similar too. This result shows that browser extensions that were added or removed throughout the 9-months-long experiment do not influence the result of users' uniqueness.

The more extensions you install, the more unique you are. In the beginning of this section, we have shown that 54.86% of users are unique among those who have at least one extension detected (D_{Ext} dataset). Figure 8 shows how uniquely identifiable users are when they have more extensions detected. Among users with at least two extensions detected, 76.25% are uniquely identifiable. This percentage rises quickly to 92.22% and 95.85% when we consider users with at least three and four extensions detected respectively.

We made a similar analysis for logins: likewise, the percentage of unique users grows if we consider users with a higher number of detected logins. 31.58% users with at least 5 logins are uniquely identifiable with their logins only. This percentage rises to 38.98% when we detected at least 8 logins. Intuitively, the more extensions or logins a user has, the more unique he becomes. It is worth mentioning that the subsets of users considered decreases as we increase the number of extensions or logins detected, as shown in Figure 4.

Uniqueness if JavaScript is disabled. Users might decide to protect themselves from fingerprinting by disabling JavaScript in

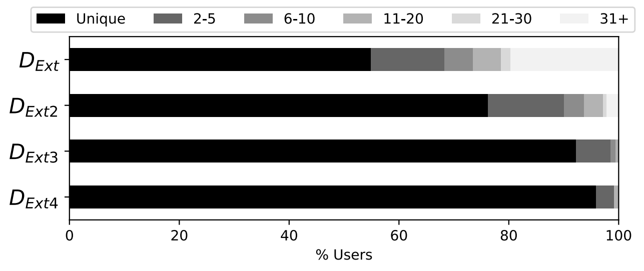


Figure 8: Anonymity sets for users with respect to the number of detected extensions

[3]Alexa ranking extracted on the the 28th of June 2018

[4]Note that Gmail is a subdomain of Google, that is why it is ranked 1 in Table 5.

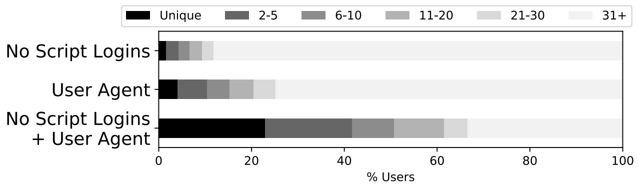


Figure 9: Anonymity sets when JavaScript is disabled

their browsers. However even when JavaScript is disabled, detection of logins via a CSP violation attack still works. Among 60 websites in our experiment, we discovered that such an attack works for 18 websites. Figure 9 shows anonymity sets for 9,492 users of D_{Log} dataset assuming users have disabled JavaScript. By considering only logins detectable with CSP, 1.63% of users are uniquely identifiable, and 4.10% are unique based on a user agent string that is sent with every request by the browser. However, when we combine the user agent string with the list of logins detectable with CSP, 22.98% of users become uniquely identifiable.

5 FINGERPRINTING ATTACKS

According to the uniqueness analysis from Section 4, 54.86% of users that have installed at least one detectable extension are unique; 19.53% of users are unique among those who have logged into one or more detectable websites; and 89.23% are unique among users with at least one extension and one login. Therefore, extensions and logins can be used to track users across websites. In this section we present the threat model, discuss and evaluate two algorithms that optimize fingerprinting based on extensions and logins.

5.1 Threat model

The primary attacker is an entity that wishes to uniquely identify a user's browser across websites. An attacker is recognizing the user by his *browser fingerprint*, a unique set of detected browser extensions and Web logins (we call them *attributes*), without relying on cookies or other stateful information. A single JavaScript library that is embedded on a visited webpage can check what extensions and Web logins are present in the user's browser. By doing so, an attacker is able to uniquely identify the user and track her activities across all websites where the attacker's code is present. We assume that an attacker has a dataset of users' fingerprints, either previously collected by the attacker or bought from data brokers.

5.2 How to choose optimal attributes?

The most straightforward way to track a user via browser fingerprinting is to check all the attributes (browser extensions and logins) of her browser. However, testing all 13k extensions takes around 30 seconds[5] and thus may be unfeasible in practice. Therefore, the *number of tested attributes* is one of the most important property of fingerprinting attacks – the attack is faster when fewer attributes are checked. But testing fewer attributes may lead to worse uniqueness results, because more users will share the same fingerprint.

While it was shown that finding the optimal fingerprint is an NP-hard problem [38], finding approximate solutions is neither a trivial task. For example, choosing the most popular attributes worked in the case of tracking based on Web history, but this strategy is not necessarily the globally optimal case.

Following the theoretical results of Gulyás et al. [38], we consider these two strategies: (1) to target a specific user, and thus to select attributes that makes her unique with high probability – called *targeted fingerprinting* algorithm, and (2) to uniquely identify a majority of users in a dataset, and thus select the same set of attributes for all users – we call it *general fingerprinting* algorithm. Targeted fingerprinting mainly uses popular attributes if they are not detectable (e.g., popular extensions are not installed) or unpopular ones if they are detectable. General fingerprinting instead, considers attributes that are detectable roughly at half of the population (this allows to chose more independent attributes which makes a fingerprint based on these attributes more unique).

Using the algorithms developed in [38], we performed experiments with general and targeted fingerprinting. Our goal is to achieve results close to those in Section 4, but by testing a smaller number of attributes[6].

5.3 Targeted fingerprinting

Attack outline. The attacker aims to identify a specific user with high probability. In order to do this, the attacker needs to have information about the targeted user in her dataset of fingerprints. The attacker generates a *fingerprint pattern* that consists of a list of attributes with a known value, such as f_j = [AdBlock=yes, LastPass=No, ...]. Notice that a fingerprint pattern contains not only extensions that the user installed, but also extensions that are not installed. This information also helps to uniquely identify the user.

Let us denote the user database as D of n users and m attributes, each row i corresponding to user U_i and each column j corresponding to attribute A_j. Let the algorithm target user U_i. First, we need to find her most distinguishing attribute A_j, shared among the smallest number of other users. Let us denote these users as $S_{i,j}$. Then we need to find a second most distinguishing property A_k which separates U_i from $S_{i,j}$. Then the algorithm continues searching for the most distinguishing attributes, until the given pattern makes U_i unique (or there are no more acceptable choices left).

Evaluation. We applied targeted fingerprinting algorithm [38] on our datasets $D_{Ext}, D_{Log}, D_{Ext} \cap D_{Log}$ and $D_{Ext} \cup D_{Log}$, and computed a fingerprint pattern for each user. By using these patterns, we have computed the anonymity sets for all datasets, that are identical to those shown in Figure 7. We therefore omit repeating these results in a new figure.

For each unique user, the fingerprint pattern contains a smaller number of attributes than the number of attributes detected for the user. For example, it is enough to test only 2 extensions for a user who has installed 4 detectable extensions. Figure 10 shows the distribution of fingerprint pattern sizes of unique users (marked with "targeted"), and compares them to the number of attributes detected for each user. The figure clearly shows that fingerprint patterns are typically smaller than the number of detected attributes users have.

For non-unique users, the size of the fingerprint pattern is often bigger than the number of detected attributes the user has. Let us discuss this on our largest dataset $D_{Ext} \cup D_{Log}$, but note that

[5]We evaluate performance in Section 6.

[6]We reused the implementation of Gulyás et al., who shared their code [37].

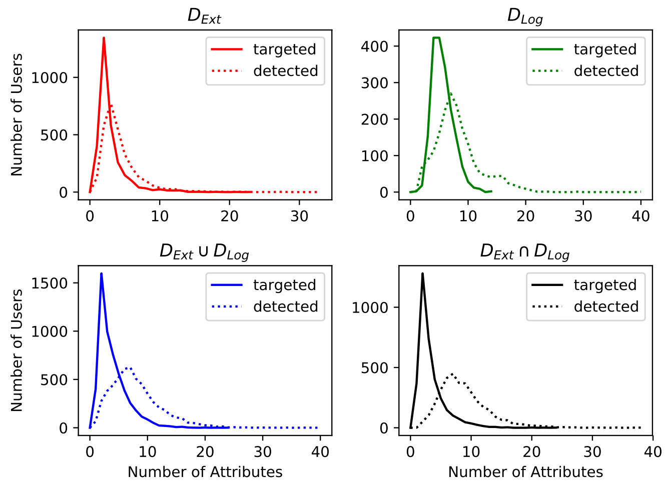


Figure 10: Comparison of fingerprint pattern size (targeted) and the total number of detected attributes (detected) for unique users.

other datasets exhibit the same phenomena. For unique users, on average we have 7.94 attributes detected, while the average size of fingerprint pattern is 3.94 attributes only. For non-unique users, the average number of detected attributes is 5.41, while the average size of fingerprint pattern grew up to 30.17. This result is not surprising: with less information it is more difficult to distinguish users, and the fingerprint pattern may also include negative attributes (i.e., LastPass=No means an extension should not be detected), which can extend the length greatly.

The targeted fingerprint is efficient, as it provides almost maximal uniqueness while reducing the number of attributes. However, it cannot be used for new users, because the attacker does not have any background knowledge about them. To reach a wider usability with a trade-off in the fingerprint pattern size, we also consider general fingerprinting [38].

5.4 General fingerprinting

Attack outline. The purpose of this algorithm is to provide a short list of attributes, called *fingerprint template*. If the attributes in a fingerprint template are tested for a certain user, she will be uniquely identified with high probability. Similarly to the example of targeted fingerprinting, we consider the fingerprint template F = [AdBlock, LastPass, ...], that would yield the fingerprint f_j^F=[yes, no, ...] for the user U_j.

The algorithm first groups all users into a set S. Then it looks for an attribute A_i that will separate S into roughly equally sized subsets S_1 and S_2 if we group users based on their attribute A_i. In the next round, it looks for another $A_j \neq A_i$ that splits S_1, S_2 further into roughly equally sized sets. This step is repeated until we run out of applicable attributes or the remaining sets could not be sliced further.

Evaluation. To apply general fingerprinting, we first measure uniqueness by using all attributes, which will be our target level A. Then, we run the algorithm until either it stops by itself (e.g., fingerprint cannot be extended further), or we terminate it early when the actual level of uniqueness B is less than 1% from level A.

Figure 11 shows the anonymity sets for different fingerprint lengths for our datasets D_{Ext}, D_{Log}, $D_{Ext} \cap D_{Log}$ and $D_{Ext} \cup D_{Log}$, generated by the general fingerprinting algorithm. For D_{Ext} and D_{Log}, the algorithm provided fingerprint templates of 485 extensions and 35 logins. In these cases the algorithm stopped since no more attributes could be used for achieving better uniqueness – hence the final anonymity sets are very close to those in Figure 7. In the cases of $D_{Ext} \cap D_{Log}$ and $D_{Ext} \cup D_{Log}$, we observed slow convergence in uniqueness, thus we could stop the algorithm earlier (shown as white dots in Figure 11). As a result, for $D_{Ext} \cap D_{Log}$, we can obtain 86.19% of unique users by testing 270 extensions and logins. For $D_{Ext} \cup D_{Log}$, we can obtain 48.31% of unique users by testing 419 extensions and logins.

We conclude that the general fingerprint can achieve a significant decrease in the fingerprint length while maintaining the level of uniqueness almost at maximum. In the next section we discuss the performance of these results.

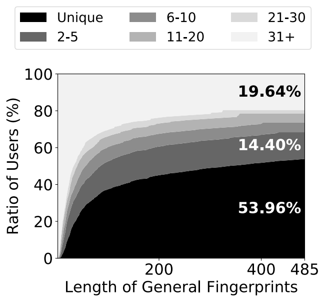
For D_{Ext} dataset, general fingerprinting algorithm provides 485 extensions, but we found out that 20 of these extensions were not stable (see Figure 3) and were not present in the last month of our experiment. Using all extensions, including unstable ones, can be useful to maintain fingerprint comparability with older data or with users having older versions of extensions. However, if we constrain general fingerprinting to stable extensions only, we get a fingerprint template of 465 extensions, leading to 50.33% uniqueness – still very close to the results of baseline uniqueness results, which was 50.35% with stable extensions only.

6 IMPLEMENTATION AND PERFORMANCE

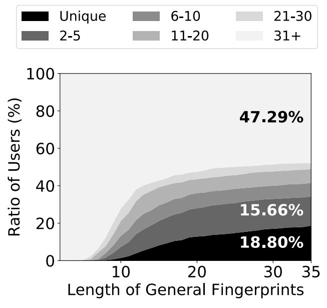
In this section we discuss the design choices we made for our experimental website and analyze whether browser extensions and Web logins fingerprinting is efficient enough to be used by tracking companies.

To collect extensions installed in the user's browser, we first needed to collect the extensions' signatures from the Chrome Web Store. We collected 12,497 extensions in August 2017, using the code shared by Sjösten et al. [56]. To detect whether an extension was installed, we tested only one WAR per extension (see more details on WARs in Section 2). Because the extensions' signatures size was 40Mb and could take a lot of time to load on the client side, we reorganized and compressed them to 600kb. However, testing all the 12,497 extensions at once took 11.3–12.5 seconds and was freezing the UI of a Chrome browser. To avoid freezing, we split all the extensions in batches of 200 extensions, and testing all the 12,497 extensions ran in approximately 30s.

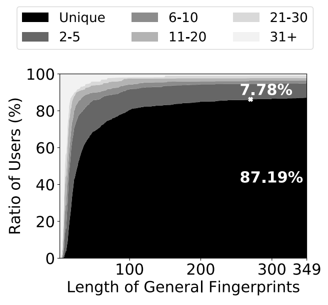
Since testing all the extensions takes too long, trackers may not be using this technique in practice. Therefore, we measured how much time it takes to apply the optimized fingerprinting algorithms from Section 5. Targeted fingerprinting addresses each user separately, hence the number of tested extensions differs a lot from user to user. General fingerprinting instead provides a generic optimization for all users. Based on our results from Section 5, an attacker can test 485 extensions and obtain the same uniqueness results as with testing all 12,497 extensions. Such testing can be run in 625



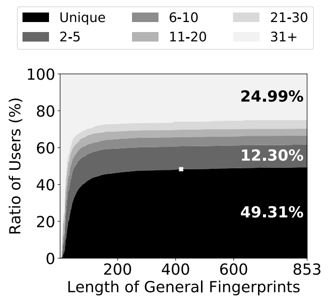
(a) D_{Ext} - 5,474 users



(b) D_{Log} - 9,492 users



(c) $D_{Ext} \cap D_{Log}$ - 3,919 users



(d) $D_{Ext} \cup D_{Log}$ - 11,047 users

Figure 11: Anonymity sets for different numbers of attributes tested by general fingerprinting algorithm.

milliseconds with the signature file size below 25Kb, which make real-life tracking feasible. For websites with limited traffic volumes, extension detection alone could be used for tracking, or for websites with a higher traffic load, it could contribute supplementary information for fingerprinting. Regarding targeted fingerprinting the attacker can do even better, as such short patterns can be detected in less than 10 milliseconds.

Compared to extension detection, Web login detection methods depend on more external factors (such as network speed and how fast websites respond), thus they should be used with caution. For redirection URL hijacking detection, we observed that the majority of Web logins can be detected in 0.9–2.0 seconds, however the timing was much harder to measure for the method based on CSP violation report. We observed that if the network was overloaded and requests were delayed, then the results of login detection were not reliable; however, it is likely that unreliable results can be easily discarded by checking timings of results (e.g., large delays appearing only in few cases).

Moreover, we found a bug in the CSP reporting implementation in the Chrome browser that makes this kind of detection even more difficult. In fact, without a system reboot for more than a couple of days (we observed that this varies between one day to multiple weeks), the browser stopped sending CSP reports. We reported the issue to Chrome developers, as this bug not only makes CSP-based detection unreliable, but more importantly CSP itself.

7 THE DILEMMA OF PRIVACY EXTENSIONS

Various extensions exist that block advertisement content, such as AdBlock [1], or block content that tracks users, such as Disconnect [6]. Such extensions undoubtedly protect users' privacy, but if they are easily detectable on an arbitrary webpage, then they can contribute to users' fingerprint and can be used to track the user across websites. In our experiment based on detecting extensions via WARs, we could detect four privacy extensions: AdBlock [1], Disconnect [6], Ghostery [8] and Privacy Badger [17]. The goal of this section is to analyze the tradeoff between the privacy loss (how fingerprintable users with such extensions are) and the level of protection provided by these extensions.

To understand this tradeoff, we computed (i) how unique are the users who install privacy extensions; (ii) how many third-party cookies are stored in the user's browser when a privacy extension

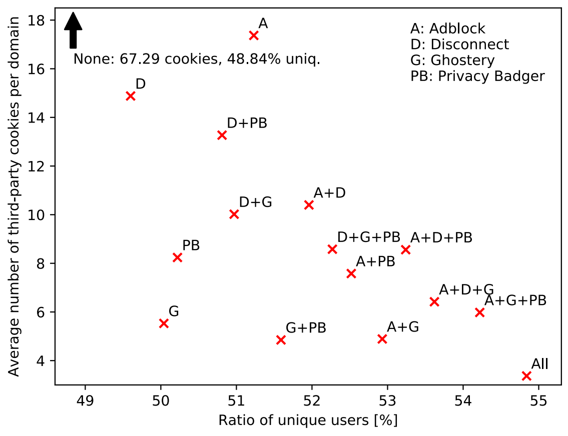


Figure 12: Uniqueness of users vs. number of unblocked third-party cookies

is activated (the smaller the number of third-party cookies, the better the privacy protection). We analyzed four privacy extensions detectable by WARs and 16 combinations of these extensions: AdBlock [1], Disconnect [6], Ghostery [8] and Privacy Badger [17].

First, we measured how a combination of privacy extensions contributes to fingerprinting. To measure uniqueness of users for a combination of extensions (i.e., AdBlock+Ghostery), we removed other privacy extensions from the D_{Ext} dataset[7] (i.e., Disconnect and Privacy Badger), and then evaluated the percentage of unique users for each combination.

Second, we measured how many third-party cookies were set in the browser, even if privacy extensions were enabled. We performed an experiment, where for each combination of extensions, we crawled the top 1,000 Alexa domains, visiting homepage and 4 additional pages in each domain[8]. We kept the browsing profile while visiting pages in the same domain, and used a fresh profile when we visited a new domain. We explicitly activated Ghostery, which is deactivated by default, and trained Privacy Badger on

[7]The total number of users does not change since we simply remove certain extensions from the user's record in our dataset.
[8]We have extracted the first 4 links on the page that refers to the same domain.

homepages of 1,000 domains before performing our experiment. We collected all the third-party cookies that remained in the user's browser for each setting and divided it by the number of domains crawled.

Figure 12 reports on the average number of cookies that remained in the browser for each combination of extensions, and the corresponding percentage of unique users.

Similarly to the results of Merzdovnik et al. [47], Ghostery blocks most of the third-party cookies, and the least blocking extension is AdBlock. Surprisingly, some combinations such as Disconnect + Ghostery resulted in more third-party cookies being set than for Ghostery alone – even after double checking the settings, and re-running the measurements, we do not have an explanation for this phenomena. However, as this can have a serious counter-intuitive effect on user privacy, it would be important to investigate this in future work.

More privacy extensions indeed increase user's unicity. All of these privacy extensions are also part of the general fingerprint we calculated in Section 5.4. However, this has little importance in practice. If we ban the general fingerprint algorithm from using privacy extensions, it will generate a fingerprint template of 531 (instead of 485) extensions, leading to a uniqueness level of 51.27%. While 46 is a significant increase in the number of extensions for fingerprinting, as we have seen it already, this would only contribute very little to the overall timing of the attack.

On the other hand, as this experiment revealed, these extensions are also very useful to block trackers. We could therefore conclude that using Ghostery is a good trade-off between blocking trackers and avoiding extension-based tracking. However, in order to efficiently solve the trade-off dilemma, we believe that such functionality should be included by default in all browsers.

8 COUNTERMEASURES

We provide recommendations for users who want to be protected from extensions- and logins-based fingerprinting. We also provide to developers recommendations to improve browser and extensions architecture in order to reduce the privacy risk for their users.

Countermeasures for extension detection. Extension detection method based on Web Accessible Resources detects 28% of Google Chrome extensions, while for Firefox the number is much smaller: 6.73% of extensions are detectable by WARs [56]. Firefox gives a good example of browser architecture that makes extensions detection difficult. The upcoming Firefox extensions API, WebExtensions, which is compatible with Chrome extensions API [4], is designed to prevent extensions fingerprinting based on WARs: each extension is assigned a new random identifier for each user who installs the extension [19]. To protect the users, developers of Chrome extensions could avoid Web Accessible Resources by hosting them on an external server, however this could lead to potential privacy and security problems [56]. Developers of the Chrome browser could nonetheless improve the privacy of their users by adopting the random identifiers for extensions as in WebExtensions API.

Most of the browsers are vulnerable to extension detection, and websites could also detect extensions by their behavior [58]. Therefore today users cannot protect themselves completely, but they still can minimize the risk by using browsers such as Firefox, where a smaller fraction of extensions are detectable.

Countermeasures for login. Users may opt for tracker-blocking and adblocking extensions, such as Ghostery [8], Disconnect [6] or AdBlockPlus [2]. But these extensions block requests to well-known trackers, while Web logins detection sends requests to completely legitimate websites, where the user has logged into anyway. Another option is to install extensions that block cookies arriving from unknown or undesirable domains. These extensions do not protect users for the same reason: cookies that belong to websites that the user visits (and treated as first-party cookies) are the same cookies used for login detection (with the only difference that the same cookies are treated as third-party cookies). For example social websites, such as Facebook or Twitter, use first-party cookies. Their social button widgets with third-party cookies may still be allowed by the browser extensions in the context of other websites. Therefore, users can protect themselves from Web logins detection, only by *disabling third-party cookies* in their browsers.

Website owners could also react to such potential privacy risk for their users. In our case, this would simply mean filtering login URL redirection, and sanity checking other redirection mechanisms against the CSP-based attack. Unfortunately, this issue has been known for a while, but website owners do not patch it because they do not consider this as a serious privacy risk [46].

Browser vendors could help avoid login detection by blocking third-party cookies by default. The new intelligent tracking protection of the Safari browser takes a step in the right direction, as it blocks access to third-party cookies and deletes them after a while.

9 RELATED WORK

Since 2006, there have been multiple proposals to detect and enumerate user's browser extensions [26, 28, 35, 43]. Most of them were blog posts that were meant to raise awareness in the security community, but they did not aim either to scientifically evaluate extension detection at large scale, nor to perform user studies, that could explain how extensions contribute to browser fingerprinting. Similarly, there has been an ongoing discussion on Web login detection in the security community [24, 31, 36, 41, 42, 46], but no quantitative studies have been made until this work.

Sjösten et al. [56] provided the first large scale study on enumerating all free browser extensions that were available to Chrome and Firefox. While their work lacked the evaluation of user uniqueness or fingerprintability, it disclosed the fact that 28 of the Alexa top 100k sites already used extensions detection. This result made it clear that extension detection is more than a theoretical privacy threat, thus deserving further studying.

Starov and Nikiforakis [58] were the first to analyze fingerprintability of browser extensions and evaluating how unique users are based on their extensions. Differently from our method, they detected extensions based on the changes extensions make to the webpages. They examined top 10,000 Chrome extensions and found that 9.2% of them were detectable on any website, and 16,6% made detectable changes on specific domains with 90% accuracy. In contrast, we used Web Accessible Resources [56] to detect extensions, and analyzed all free Chrome Web Store extensions. In our measurement period, we observed that 27–28% of all free Chrome extensions were detectable on any website with 100% accuracy. While we did not measure this, in the study of [56] the authors found that 38.96% of top 10k extensions in the Chrome Web Store are detectable with WARs. Sánchez-Rola et al. [54] detected browser extensions through a timing side-channel attack, and were able to detect all extensions in Firefox and Chrome that use access control settings, regardless of the visited site.

Both us and Starov and Nikiforakis analyzed the stability of the proposed detection method. For a sample of 1,000 extensions, Starov and Nikiforakis concluded that 88% of extensions were still detectable after 4 months. In our study, we analyzed 12,164 extensions, and conclude that 72.4% of them are detectable in every month during 9-months period.

To evaluate uniqueness of users based on their browser extensions, Starov and Nikiforakis have collected installed extensions for 854 users. In total, their users had 174 extensions that were fingerprintable. Sánchez-Rola et al.[54] collected fingerprints from only 204 users and tested for 2,000 Chrome and Firefox extensions. In our study, we have 7,643 Chrome users, for whom we tested 16,743 extensions, among them 1,110 extensions were installed by our users.

Regarding performance, Starov and Nikiforakis [58] reported that to detect 5 extensions, their testing website needed roughly 250 ms. Sánchez-Rola et al.[54] are using timing attack, which works in a very similar way as detecting WARs. When querying a non-exist (fake) WAR of an extension, the authors observed a difference in the time the browser takes to respond to the query, depending on whether the extension is installed in the user's browser or not. The difference is caused by the access control mechanism of the browser when the concerned extension is installed or not in the browser. Because of this timing method, Sánchez-Rola et al. had to make 10 calls per extension, while in our work we made only one single call per extension. We measured that the checking time of non-existing resources and loading existing WARs are very close to each other (around 1 ms), thus we argue that our approach is significantly faster.

10 DISCUSSION AND FUTURE WORK

Realistic datasets. To compare our study with previous works on fingerprinting by browser extensions, we analyzed different random subsets of 7,643 users, who run Chrome web browser (where browser extension detection is possible in our experiment). Figure 13 shows how user uniqueness based on extensions changes with respect to the various subsets of our dataset. It clearly demonstrates an intuition that the smaller the user set is, the smaller is the diversity of users, and the easier it is to uniquely identify them.

Figure 13 compares our results to previous studies on browser extensions fingerprinting: we have 7,643 Chrome users, while previous studies had 204 [54] and 854 [58] users, and therefore draw different conclusions about uniqueness of users based on browser extensions.

We reported on the number of unique users in subsets of 204 and 854 users in Section 3.2 (see Table 2). By exploring this comparison, we raise a fundamental question: *What is the "right" size for the dataset?*

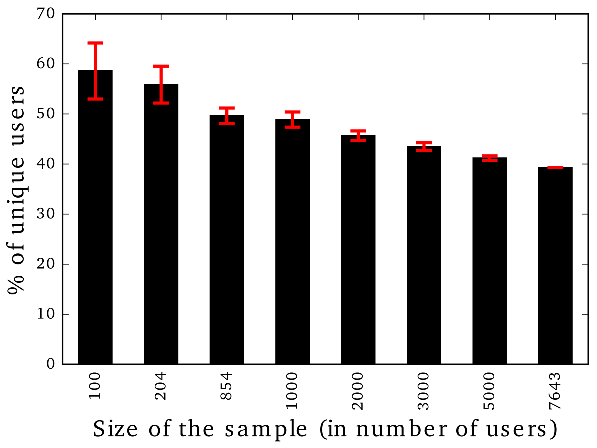


Figure 13: Uniqueness of Chrome users based on their extensions only vs. number of users - 204 is the number of users used in [54] and 854 the number of users considered in [58]

Taking a look at research on standard fingerprinting, in 2010 Eckersley showed that 95% of browsers were unique based on their properties [30], which was backed by several papers since then [25, 45]. However, a recent study states that by looking at 2 million fingerprints in 2018, the authors only found 33.6% of those fingerprints to be unique [34].

It is extremely difficult for computer scientists to get access to such large datasets – in our experience, we advertised our experiment website through all possible channels, including Twitter, Reddit, and press coverage. We experienced that having larger, high quality datasets is a highly nontrivial research task. It is important to re-evaluate our results over time while also aiming to obtain larger dataset sizes.

Stability of fingerprints While studying uniqueness based on various behavioural features, it is very important to know how stable these features are, as the ability to use some of this information as part of a fingerprint does not solely depend on its anonymity set of overall entropy, but also on the information stability (i.e., how frequently it changes over time). Vastel et al. [59] recently analyzed the evolution of fingerprints of 1,905 browsers over two years. They concluded that fingerprints' evolution strongly depends on the type of the device (laptop vs mobile) and how it is used. Overall, they observed that 50% of browsers changed their fingerprints in less than 5 days.

In our study we did not have enough data to make any claims about the stability of the browser extensions and web logins because only few users repeated an experiment on our website (to be precise, only 66 users out of 16,393 users have made more than 4 tests on our website). We would expect that browser extensions are more stable than logins since users do not seem to change extensions very often, while they may log in and log out of various websites during the day. However, studying the stability of extensions and logins would require all our users to install a tool (probably a browser extension) in their browsers that would monitor the extensions they install and logins they perform. This kind of experiment would be even harder to perform at large scale since users do not easily trust to install new browser extensions. In AmIUnique experiment, Laperdrix [44] was trying to measure stability of browser fingerprints – he collected data from 3,528 devices over a twenty-month-long experiment. We managed to have 16,393 users testing our website in 9 months. This shows that users have more trust in testing their browser on a website than installing new extensions.

We therefore keep the study of fingerprints stability for future work and raise an important question in privacy measurement community: *How can we ensure a large scale coverage of users for our privacy measurement experiments?*

11 CONCLUSION

This paper reports on a large-scale study of a new form of browser fingerprinting technique based on browser extensions and website logins. The results show that 18.38% of users are unique because of the extensions they install (54.86% of users that have installed at least one detectable extension are unique); 11.30% of users are unique because of the websites they are logged into (19.53% are unique among those who have logged into one or more detectable websites); and 34.51% of users are unique when combining their detected extensions and logins (89.23% are unique among users with at least one extension and one login). It also shows that the fingerprinting techniques can be optimized and performed in 625 ms.

This paper illustrates, one more time, that user anonymity is very challenging on the Web. Users are unique in many different ways in the real life and on the Web. For example, it has been shown that users are unique in the way they browse the Web, the way they move their mouse or by the applications they install on their device [40]. This paper shows that users are also unique in the way they configure and augment their browser, and by the sites they connect to. Unfortunately, although uniqueness is valuable in society because it increases diversity, it can be misused by malicious websites to fingerprint users and can therefore hurt privacy.

Another important contribution of this paper is the definition and the study of the trade-off that exists when a user decides to install a "privacy" extension, for example, an extension that blocks trackers. This paper shows that some of these extensions increase user's unicity and can therefore contribute to fingerprinting, which is counter-productive. We argue that these "privacy" extensions are very useful, but they should be included by default in all browsers. "Privacy by default", as advocated by the new EU privacy regulation, should be enforced to improve privacy of all Web users.

ACKNOWLEDGMENT

First of all, we would like to thank the valuable comments and suggestions of the anonymous reviewers of our paper. This research has been partially supported by the ANR projects AJACS ANR-14-CE28-0008 and CISC ANR- 17-CE25-0014-01. We are grateful for Alexander Sjösten, Steven Van Acker, Andrei Sabelfeld, who shared their code and signature database for Chrome browser extension detection [56], and also for Robin Linus, who allowed us to build on his script on social media presence detection. We thank Imane Fouad and Natasa Sarafijanovic-Djukic for their help in evaluating the effect of browser extensions on third-party cookies.

REFERENCES

[1] AdBlock Official website. https://getadblock.com/.
[2] Adblockplus official website. https://adblockplus.org/.
[3] Brave browser. https://brave.com/.
[4] Chrome Extensions API. https://developer.chrome.com/extensions.
[5] Content security policy (csp).
[6] Disconnect Official website. https://disconnect.me/.
[7] Faceboook website. https://www.facebook.com/.
[8] Ghostery Official website. https://www.ghostery.com/.
[9] Google Chrome browser. https://www.google.com/chrome/.
[10] Google. manifest - web accessible resources.
[11] Google. manifest file format.
[12] Google website. https://www.google.com/.
[13] Google's Gmail. https://gmail.com.
[14] Lastpass official website. https://www.lastpass.com/business.
[15] Linkedin website. https://www.linkedin.com/.
[16] Opera browser. http://www.opera.com/.
[17] Privacy Badger - Electronic Frontier Foundation. https://www.eff.org/fr/privacybadger.
[18] Same Origin Policy. https://www.w3.org/Security/wiki/Same_Origin_Policy.
[19] WebExtensions web_accessible_resources. https://developer.mozilla.org/en-US/Add-ons/WebExtensions/manifest.json/web_accessible_resources.
[20] Youtube website. https://www.youtube.com/.
[21] G. Acar, C. Eubank, S. Englehardt, M. Juárez, A. Narayanan, and C. Díaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 674–689, 2014.
[22] G. Acar, M. Juárez, N. Nikiforakis, C. Díaz, S. F. Gürses, F. Piessens, and B. Preneel. Fpdetective: dusting the web for fingerprinters. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 1129–1140, 2013.
[23] J. P. Achara, G. Ács, and C. Castelluccia. On the unicity of smartphone applications. *CoRR*, abs/1507.07851, 2015.
[24] T. Anthony. Detect if visitors are logged into twitter, facebook or google+. http://www.tomanthony.co.uk/blog/detect-visitor-social-networks/, 2012.
[25] K. Boda, Á. M. Földes, G. G. Gulyás, and S. Imre. User tracking on the web via cross-browser fingerprinting. In *Information Security Technology for Applications - 16th Nordic Conference on Secure IT Systems, NordSec 2011, Tallinn, Estonia, October 26-28, 2011, Revised Selected Papers*, pages 31–46, 2011.
[26] M. Bryant. Dirty browser enumeration tricks - using chrome:// and about: to detect firefox and addons. https://thehackerblog.com/dirty-browser-enumeration-tricks-using-chrome-and-about-to-detect-firefox-plugins/index.html, 2014.
[27] 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, NDSS 2017, San Diego, California, USA, 26 February - 1 March, 2017*, 2017. To Appear.
[28] G. Cattani. The evolution of chrome extensions detection. http://blog.beefproject.com/2013/04/the-evolution-of-chrome-extensions.html, 2013.
[29] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific Reports*, 3:1376 EP –, 2013.
[30] P. Eckersley. How unique is your web browser? In *Privacy Enhancing Technologies, 10th International Symposium, PETS 2010, Berlin, Germany, July 21-23, 2010. Proceedings*, pages 1–18, 2010.
[31] A. Elsobky. Novel techniques for user deanonymization attacks. https://0xsobky.github.io/novel-deanonymization-techniques/, 2016.
[32] 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, October 24-28, 2016*, pages 1388–1401, 2016.
[33] H. Gamboa, A. L. N. Fred, and A. K. Jain. Webbiometrics: User verification via web interaction. In *2007 Biometrics Symposium*, pages 1–6, 2007.
[34] A. Gómez-Boix, P. Laperdrix, and B. Baudry. Hiding in the Crowd: an Analysis of the Effectiveness of Browser Fingerprinting at Large Scale. In *Web Conference (WWW 2018)*, Lyon, France, 2018.
[35] J. Grossman. I know what you've got (firefox extensions). http://blog.jeremiahgrossman.com/2006/08/i-know-what-youve-got-firefox.html, 2006.
[36] J. Grossman. Login detection, whose problem is it? http://blog.jeremiahgrossman.com/2008/03/login-detection-whose-problem-is-it.html, 2008.
[37] G. G. Gulyás, G. Acs, and C. Castelluccia. Code repository for paper titled 'near-optimal fingerprinting with constraints'. https://github.com/gaborgulyas/constrainted_fingerprinting, 2016.
[38] G. G. Gulyás, G. Acs, and C. Castelluccia. Near-optimal fingerprinting with constraints. *Proceedings on Privacy Enhancing Technologies*, 2016(4):470–487, 2016.
[39] J. Haag. Modern and flexible browser fingerprinting library. https://github.com/Valve/fingerprintjs2.

[40] B. Hayes. Uniquely me! how much information does it take to single out one person among billions? 102:106–109, 2014.

[41] E. Homakov. Using content-security-policy for evil. http://homakov.blogspot.fr/2014/01/using-content-security-policy-for-evil.html, 2014.

[42] E. Homakov. Profilejacking - legal tricks to detect user profile. https://sakurity.com/blog/2015/03/10/Profilejacking.html, 2015.

[43] K. Kotowitz. Intro to chrome addons hacking: fingerprinting. http://blog.kotowicz.net/2012/02/intro-to-chrome-addons-hacking.html, 2012.

[44] P. Laperdrix. *Browser Fingerprinting: Exploring Device Diversity to Augment Authentication and Build Client-Side Countermeasures*. PhD thesis, INSA Rennes, 2017.

[45] P. Laperdrix, W. Rudametkin, and B. Baudry. Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 878–894, 2016.

[46] R. Linus. Your social media fingerprint. https://robinlinus.github.io/socialmedia-leak/, 2016.

[47] G. Merzdovnik, M. Huber, D. Buhov, N. Nikiforakis, S. Neuner, M. Schmiedecker, and E. Weippl. Block me if you can: A large-scale study of tracker-blocking tools. In *2nd IEEE European Symposium on Security and Privacy*, Paris, France, 2017. To appear.

[48] K. Mowery and H. Shacham. Pixel perfect: Fingerprinting canvas in HTML5. In M. Fredrikson, editor, *Proceedings of W2SP 2012*. IEEE Computer Society, May 2012.

[49] 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, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 541–555, 2013.

[50] Ł. Olejnik, C. Castelluccia, and A. Janc. Why johnny can't browse in peace: On the uniqueness of web browsing history patterns. In *Hot Topics in Privacy Enhancing Technologies (HotPETs 2012)*, 07 2012.

[51] I. Paul. Firefox will stop supporting plugins by end of 2016, following chrome's lead. https://www.pcworld.com/article/2990991/browsers/firefox-will-stop-supporting-npapi-plugins-by-end-of-2016-following-chromes-lead.html.

[52] M. Pusara and C. Brodley. User re-authentication via mouse movements. In *ACM Workshop Visualizat. Data Mining Comput. Security*, page 1–8, 2004.

[53] J. Roth, X. Liu, and D. Metaxas. On continuous user authentication via typing behavior. 23(10):4611–4624, 2014.

[54] I. Sánchez-Rola, I. Santos, and D. Balzarotti. Extension breakdown: Security analysis of browsers extension resources control policies. In *26th USENIX Security Symposium*, pages 679–694, 2017.

[55] J. Schuh. Canvas DefendeSaying Goodbye to Our Old Friend NPAPI, September 2013. https://blog.chromium.org/2013/09/saying-goodbye-to-our-old-friend-npapi.html.

[56] A. Sjösten, S. Van 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*, CODASPY '17, pages 329–336, New York, NY, USA, 2017. ACM.

[57] S. Stamm, B. Sterne, and G. Markham. Reining in the web with content security policy. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 921–930, 2010.

[58] O. Starov and N. Nikiforakis. Xhound: Quantifying the fingerprintability of browser extensions. In *Proceedings of the 38th IEEE Symposium on Security and Privacy*, pages 941–956, 2017.

[59] A. Vastel, P. Laperdrix, W. Rudametkin, and R. Rouvoy. FP-STALKER: Tracking Browser Fingerprint Evolutions. In *39th IEEE Symposium on Security and Privacy (S&P 2018)*, 2018.

[60] M. West, A. Barth, and D. Veditz. Content Security Policy Level 2. W3C Candidate Recommendation, 2015.

[61] Y. Zhong, Y. Deng, and A. K. Jain. Keystroke dynamics for user authentication. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Providence, RI, USA, June 16-21, 2012*, pages 117–123, 2012.