Protecting Users Identity Against Browser Fingerprinting

MSc Research Project
MSc in Cybersecurity

Prem Shankar Shingote
Student ID: X20257040

School of Computing
National College of Ireland

Supervisor: Dr. Vanessa Ayala-Rivera

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Prem Shankar Shingote		
Student ID:	x20257040		
Programme:	MSc in Cybersecurity	**Year:**	2022
Module:	MSc Research Project		
Supervisor:	Dr. Vanessa Ayala-Rivera		
Submission Due Date:	15th Dec 2022		
Project Title:	Protecting Users Identity Against Browser Fingerprinting		
Word Count:	6881	**Page Count**	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Prem Shankar Shingote……………………………………………………

Date: 15th Dec 2022………………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Protecting Users Identity Against Browser Fingerprinting

Prem Shankar Shingote
x20257040

Abstract

Nowadays, customer data is the gold mine for advertisers, marketing companies, and hackers. They use every possible method to track users' online activity, and currently they are using a new user-tracking mechanism called "Browser Fingerprinting." This method is different from cookie-based tracking; The browser fingerprinting mechanism collects common attributes of users' devices like OS version, screen resolution, font, and many more without their knowledge and combines them to generate one unique identifier token. This token helps attackers and advertisers spot that user over the internet with 90–99% accuracy. Consequently, the "privacy" of online users is seriously threatened; also, attackers can easily craft the attack based on the users' system configuration. After understanding the seriousness of the issue, modern browsers like Firefox, Brave, and TOR started blocking JavaScript. As a result, they are preventing users from browser fingerprinting, but due to the unavailability of JavaScript, many websites are not functioning properly. That's why it's challenging for users to protect their privacy. To resolve this privacy issue, we developed a browser extension called "Browser Fingerprint Defender," which anonymizes the users' browser by performing an API normalization against passive fingerprinting and object-based JavaScript fingerprinting. It masks the actual system values with generic random values, before sending them to the requested website. After randomizing the parameters, fingerprinting token values also change, and it becomes challenging for advertisers to track users online. To examine the effectiveness of our extension, we tested it on online experimental sites. And as per the test results, it provides appropriate anonymity to users and solves the problem of users' online privacy by making their identity less unique.

1 Introduction

Latest technologies are always being developed to enhance browsing experience of a user over World Wide Web [1]. Current methods and technologies, such as JavaScript, CSS, and HTML5, allow users to modify their web pages more easily and help improve the browsing experience. But on the flip side of the coin, they have various dark sides too, which could expose users to a wide range of risks, and also compromise their privacy [2]. A few years back, online tracking companies tracked users' activity using persistence cookies (zombie cookies), which were then used against the user for targeted marketing and user profiling [3] [4]. When everyone became aware of cookie-based tracking, almost every modern browser-built protection against it, and users also started blocking third party cookies. Subsequently, advertisers have insufficient user data, and their advertising business gets negatively impacted [5]. So, to overcome this issue, advertisers came up with a new passive user tracking method in which people who try to protect their identity by disabling cookies, using an incognito window, changing browsers, or even changing their IP address can still be easily identified using the browser fingerprinting technique [6].

A large amount of high ranked websites performs browser fingerprinting and fetches some most common system details like *System fonts, System Time Zone, whether cookies are enabled, Operating system, OS language, Platform, Keyboard layout, Tor browser or not? Secure browser or not?, Browser permissions, User agent, Sensors (such as gyroscope, proximity, and accelerator), Browser local databases, Navigator properties, HTTP header attributes, Web browser extensions used, Audio context analysis, CPU class, HTML5 canvas fingerprinting (looking at canvas size), Touch support* and much more [7] [8]. After fetching all these details, advertisers combine those small details to create one unique token that helps identify the user on the internet without their previous data or pre-identity, which is called a "fingerprint. Now, whenever that user accesses the same website or any website over the internet by clearing the cookies or from a different IP address, online trackers can still simply recognize a user by their unique fingerprint token [9].

Every system of the user has a different configuration of software and hardware, which makes their Fingerprint unique. According to research conducted by Panopticlick, it was discovered that from the set of 133801.5 browsers, only 1 browser has a duplicate fingerprint, and as per technological development, this uniqueness will increase, making it very easy to spot a specific user from millions [10]. As reported by Krishna.V. Nair and Elizabeth Rose Lalson in their 2018 research, they have proven that the fingerprint of every internet user is 96% unique [5].

Currently, online advertising companies are working together with several websites from different domains to collect user fingerprinting data and create profiles of users based on their system details, interests, and activities. After collecting all this data from websites, advertisers create one unique fingerprinting token as well as one digital profile in their database. Nearly every popular website shares their own database of users' profiles and links this database to other advertisers for different motives, such as targeted advertising [11] [12]. According to analyses conducted in late 2021, on Alexa 's top-100k websites for browser fingerprinting attempts, and as a result of this research, nearly 10.18% of websites performed browser fingerprinting [13]

After understanding the seriousness of fingerprinting, numerous browsers like Firefox and TOR started providing inbuilt protection against them by blocking WebRTC and Canvas APIs which are responsible for fingerprinting. But blocking the JavaScript API can lead to website crashes, and many websites are not working or opening as intended. These crashes are not user friendly, and they have an effect on the performance of the browser; that is why many browsers have not implemented fingerprinting protection [14] [13].

So, in the end, if a user becomes aware of being tracked by websites, he cannot do much to protect himself due to the unavailability of a full proof solution. At this point, our research question comes into play: "How users can protect their privacy and identity over the internet where online advertisers and marketers are tracking users' online activity using browser fingerprinting techniques even though the user is using a VPN and deleting cookies?"

To solve this problem, we have proposed an extension for the Google Chrome browser named "Browser Fingerprint Defender." We select Chrome since it is the most widely used browser in the world [15]. In this extension, we have designed several methodologies based on currently existing measures for preventing browser fingerprinting. Browser Fingerprint Defender intercepts the request for JavaScript, which has the request for system parameters, and replaces the modified copy of the JavaScript file having dummy system parameters with the requested one.

Every time, Browser Fingerprint Defender provides dynamic fingerprints and does not modify any other parts of JavaScript or HTML, reducing its detectability significantly. This method removing user-uniqueness and pretend to be someone else. We observe that after enabling this extension, a user's fingerprint is totally different from the original one, and it won’t affect the performance of the website. This extension safeguards the user’s privacy and contributes towards making the internet safer place.

This approach has some weaknesses as well. The Browser Fingerprint Defender detects fingerprinting scripts based on our pre-defined criteria. which must focus on specific types of fingerprints to prevent over-blocking, have some limitations.

i) First, they could skip fingerprinting scripts that do not fit their current detection criteria.
ii) Second, to identify new or updated fingerprinting scripts, the detection criteria must be continuously maintained.
iii) Third, if every time users change the values of a parameter, this pattern becomes identical enough for the website to recognize that a user is changing the parameter values.

2 Related Work

A browser fingerprint is more than just a collection of information specific to a certain device. It represents the actual computer component and its exact details. Attackers can find possible security loopholes by analysing and comparing the user system components with the vulnerable component database, such as Common Vulnerabilities and Exposures (CVE) [16]. The primary purpose of browser fingerprinting is to validate the identity of a user without any of this interaction. At the start of computer networking, a user can be traced by the IP address of their machine. but now in browser fingerprinting, it includes the IP address as well as several aspects of the user's system [17]. As we can see in this research paper about how to protect from browser fingerprinting with the help of other methods designed to avoid fingerprint tracking.

A. Using Multiple Browsers:

The browsing technology sets up the communication between the browser and the web server very efficiently but fails to restrict the user's sensitive data. In an attempt at browser fingerprinting, the webpages ask the browser for various system details, including hardware configuration and software versions of the system. Being a default functionality of the browser, it forwards all requested system information under the name of compatibility improvement of website.

One of the most basic and easy ways to protect ourselves from fingerprinting is to use multiple browsers. Using different browsers for different purposes gives their individual fingerprint to trackers. But in spite of that, researcher Károly Boda shows in his research about cross-browser fingerprinting. According to that, if a user changes his browser, still some of his system components like screen resolution, time zone remain the same and that is sufficient to spot the user from crowds[18]. By collecting sufficient data of OS-like plugins, list of fonts, and the scripts which identify the system details after bypassing the browser as this system information is static and won't change even after changing the browser. One of the methods identified by researcher Yinzhi Cao is identifying the user very accurately (90.84% to 99.24% across multiple browsers) with the help of the WebGL API [19].

B. Blocking extensions:

According to many researchers and reports, another robust way to prevent browser fingerprinting is by blocking all the scripts before their execution in the browser [20]. To prevent execution of scripts, there are multiple well-known extensions available in the market like Disconnect, Ghostery, and NoScript [21] which perform very well while blocking all the scripts. Despite that, in order to perform their full proof, Woking this extension first requires a list of all fingerprinting scripts that we need to block. As there is a continued development in browsing technology, it is very challenging to keep track of newly evolved fingerprinting scripts. Also, as we discussed in the introduction to this report, some websites won't work properly when a browser is blocking the scripts that are required for loading their CSS content [5].

C. Tor Browser:

There are multiple browsers in the market which have similar functionality like TOR browsers and they have inbuilt protection against browser fingerprinting by blocking all Canvas APIs [14]. The TOR browser is the upgraded version of Firefox browser, and which is particularly developed for TOR network. As mentioned in the design document of TOR, it defends users against "Reimplementation or Subsystem Modification", "Value spoofing", "Functionality or Feature Removal", "Site Permissions" and makes browser fingerprints as unique as possible [22]. Although this approach has various problems when dealing with real words, First, the fingerprint generated by TOR browser is very specific and it is duck soup for advertisers to identify that user is using TOR browser. Second, users have to compromise the GUI of the website while using it because it blocks all graphic API calls. Third, the proposed protection is so weak that small changes in browsers make huge changes in fingerprints. Fourth, remaining personalization effectiveness and customizability are heavily impacted due to its mono-configuration.

D. Spoofing extensions:

Another effective way to prevent browser fingerprinting is by using the spoofing extension technique. This method is more effective than the above three fingerprinting prevention methods. This extension sends dummy information in response to trackers. This type of browser extension is widely available on the market and very effectively works on the Google Chrome browser. With its very easy configuration, the Google Chrome browser can easily pretend to be Firefox, Opera, or Safari, as shown in the research paper published by Nick Nikiforakis [23]. But the drawback of this method is that the JavaScript property will reveal a different value from what the user agent says, and it will be identical to a tracker that the browser intentionally modified its default values. To solve this issue, researcher Christof Ferreira Torres developed one browser extension called FP-Block, which does the separation of web identities [24]. The idea behind fingerprint generation in FP-Block is to generate fingerprints for every website opened by a user, but whenever a browser reconnects or opens the same website, it'll reuse the previously generated identity. Although this concept is good for defences against fingerprinting, the implementation part suffers from the same issues as the previous extensions. Also, the generated fingerprint is inconsistent, and it is very easy for advertisers to find hidden information or modified information.

After reviewing all the above research papers from previous researchers, we come to the conclusion that there are various ways to prevent browser fingerprinting. But all solutions are temporary and don't offer much protection against advanced and modern fingerprinting attacks [25] [26]. The already existing browser prevention technique did not work as expected. In modern browsers, Google Chrome is the most used and most vulnerable browser, which makes it easier to steal browser fingerprints. But there are some modern browsers, such as Brave and TOR, that protect users' privacy by blocking browser fingerprinting attempts from all websites. In study of Brave and TOR browser we found that how can we randomize the requested browser fingerprinting parameters. We develop an extension that helps users protect their identity and prevents them from being tracked. We have developed a Google Chrome browser extension because, as we mentioned previously, Google Chrome is the most vulnerable browser of all. We name our browser extension "Browser Fingerprint Defender." This extension can modify the values of parameters used to fingerprint the user but won’t affect the user's browsing performance.

3 Research Methodology

This section of the report will describe the approaches used to detect the attack of browser fingerprinting and contain a prevention technique against browser fingerprinting. The action plan for implementation as well as the follow-up procedure will be covered in the report section below. Also in Figure 1: the workflow diagram of Browser Fingerprint Defender describes the detailed steps from data collection to final results. This overall process has fourteen stages and partitions them into three stages: identifying the problem, Conducting, Preliminary Experiments and Evaluating Countermeasures. Will see every stage one by one in the below section.

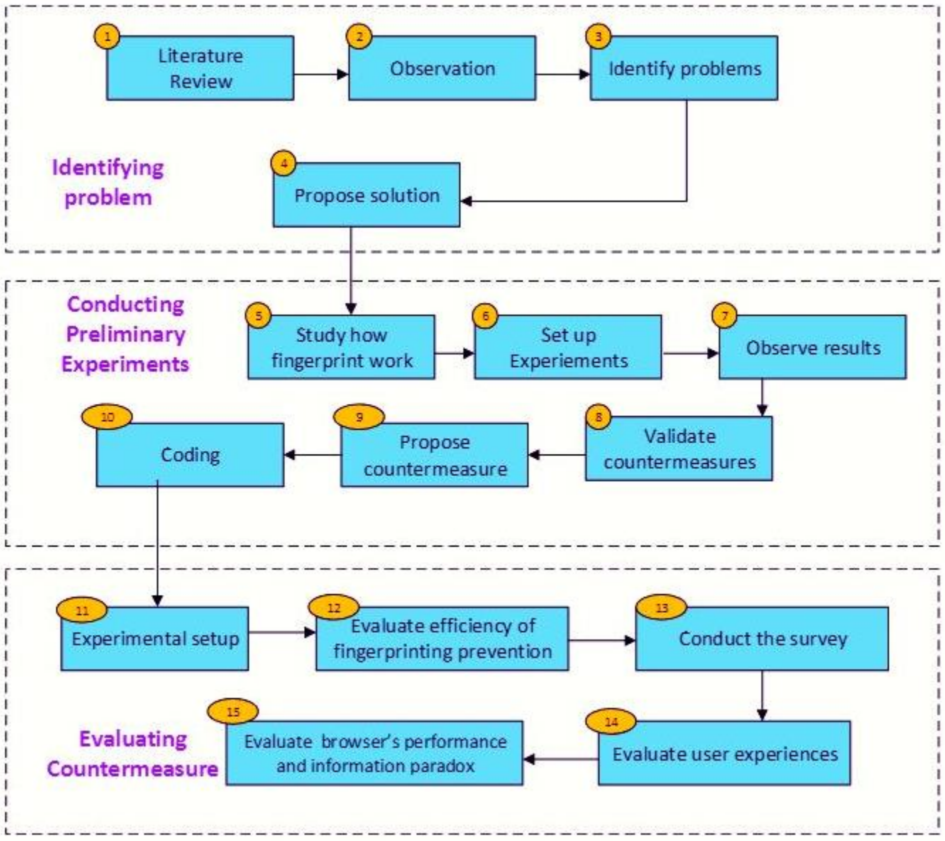

Identifying problem
1 Literature Review
2 Observation
3 Identify problems
4 Propose solution
Conducting Preliminary Experiments
5 Study how fingerprint work
6 Set up Experiments
7 Observe results
8 Validate countermeasures
9 Propose countermeasure
10 Coding
Evaluating Countermeasure
11 Experimental setup
12 Evaluate efficiency of fingerprinting prevention
13 Conduct the survey
14 Evaluate user experiences
15 Evaluate browser's performance and information paradox

Figure 1: The workflow diagram of Browser Fingerprint Defender

A. Identifying Problem:

1. Literature Review: This section is the first stage where we identified the problem of every Internet user being monitored by the browser fingerprinting technique, and we observe that it's a major threat to the privacy of the user.

2. Observation: In this section, we studied and critically analyzed the multiple research papers, then observed the solutions proposed by other researchers to prevent browser fingerprinting.

3. Identify Problem: We observe the overall research procedure, different techniques, and scenarios presented by different authors, and after that, we identify if the user is aware of browser fingerprinting, but there is still no full proof solution to prevent a fingerprinting attack.

4. Propose Solution: After identifying the problem in the time following, we started finding a solution that protects the users' privacy and won't impact their performance. So, we started working and came up with a user friendly, lightweight browser extension called "Browser Fingerprinting Defender."

B. Conducting Preliminary Experiments (solution design)

5. Study how fingerprints work: To develop the "Browser Fingerprint Extension," we first need to understand how the fingerprinting method works, which are the different

methods of fingerprinting, which parameters are used to generate fingerprints, and which are the important parameters in fingerprinting. This is all we learn while studying the workings of fingerprints.

6. Set up Experiments: To test the workings of fingerprinting, we have used websites like browserleaks.com [27], which have experimental setups. On this website, there are several different fingerprint scripts implemented that fetch system parameters and generate the fingerprint.

7. Observe Results: After testing our browser in the experimental setup, we got to know that these websites are collecting too much information about our system when we only visit once, and we have no idea about it..

8. Validate Countermeasures: To solve the answer to this question, we have planned some countermeasures that change the parameters of the system while sending responses to the website and won't affect the performance of the website.

9. Propose Countermeasures: To implement the validated countermeasures, we designed a structure in such a way that a Chrome user can pretend to be a user of Firefox, Edge, or Safari and hide his actual identity successfully.

10. Coding: After all the experiments and successful planning of countermeasures, we developed a frontend for Browser Fingerprint Defender using HTML and a backend using JavaScript.

C. Evaluating Countermeasure

11. Experimental setup: As we don't have our own setup to test our extension, that's why we tested it on browserleaks.com. This proposed countermeasure setup for fingerprints validates the efficacy of fingerprint prevention, user experience, This proposed countermeasure setup for fingerprints validates the efficacy of fingerprint prevention, user experience, overhead performance, and information paradox of the planned fingerprint countermeasure.

12. Evaluate the efficiency of fingerprinting prevention: Here we had a successfully developed extension, and after enabling it, we tested it on browserleaks.com. After testing, we got a very successful result. When we were using Google Chrome browser but after enabling Browser Fingerprint Defender, out fingerprint showed different every time as Firefox, Edge, and Safari.

13. Conduct a survey: After successfully testing the effectiveness of our extension, we tested it on a different system. Before enabling the extension, our fingerprint was 27% unique, and after enabling the extension, it shows as 99.52% unique.

14. Evaluate the user experience: As we tested this extension on 5 to 10 different systems, we didn't face any issues while accessing any website. The experience of browsing the web while enabling extensions is as similar as browsing the web without any extensions.

15. Evaluate browser’s performance and information paradox: According to the overall result of this phase, the proposed fingerprint countermeasure must be able to solve the issues with the current countermeasure. The overall performance of the extension meets the requirement and successfully protects the privacy of users over the internet.

4 Design Specification

To develop this extension, we used some basic programming languages, such as JavaScript and HTML. It is essential for the extension to communicate easily with the browser. So, at the start, we developed a simple extension that can easily connect with a browser and run smoothly over it. Then we start developing code to extract data from the browser. After successfully running the extension, we started fetching system attributed information such as browser information, system information, and OS information. In the randomizing script, we can’t randomize all system attributes because randomizing all attributes might impact browser performance and stability, and sometimes it might crash the browser. To avoid this issue, we develop a list of selected attributes which can be changed, and which can’t be changed during the process of randomizing the parameters. When browser shares the system information for randomizing, we modify the value of parameters, and to change those attribute values, we have created a list of dummy values that look similar to real parameter values. So, whenever a browser asks for system parameters, our extension will give the dummy values from this list. We have multiple values for a single parameter, so our extension picks any random value from the list and submits it to the browser.

A. Workflow diagram of Browser Fingerprint Defender

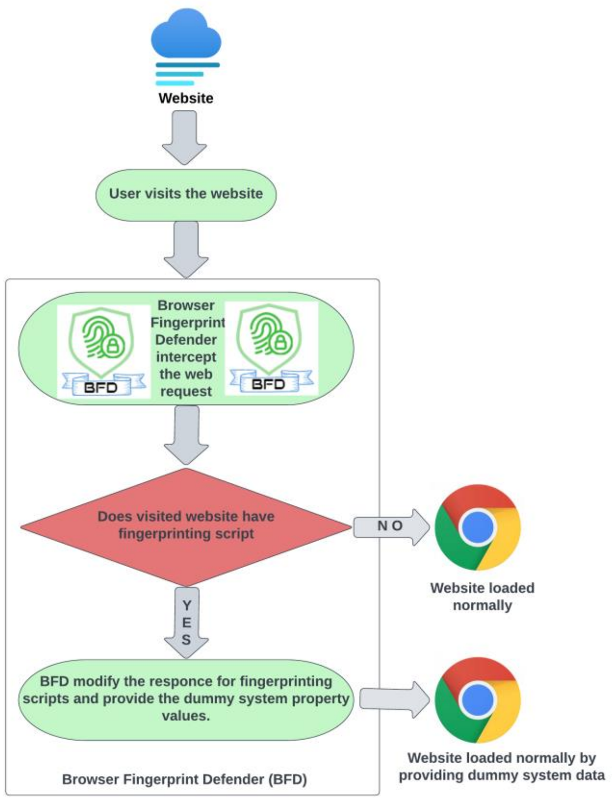

Website
User visits the website
Browser Fingerprint Defender intercept the web request
BFD
BFD
Does visited website have fingerprinting script
NO
Website loaded normally
YES
BFD modify the responce for fingerprinting scripts and provide the dummy system property values.
Website loaded normally by providing dummy system data
Browser Fingerprint Defender (BFD)

Figure 2: Workflow diagram of Browser Fingerprint Defender

The above Figure 2 shows the workflow diagram of the browser fingerprint defender and gives an overview of the internal functionality of the extension. This extension intercepts browser requests and checks for fingerprinting scripts, then reports the dummy values to the website. To understand the design and workflow of Browser Fingerprint Defender, I will divide the extension into three parts, then explain each one by one as follows:

i. **User visits the website**:
After enabling the browser extension, the user visits any random website on the internet. Then all website data gets passed through the extension.

ii. **Browser Fingerprint Defender intercepts the request**:
After visiting the website, our extension starts intercepting each and every JavaScript from the website backend for fingerprinting scripts. If it is successful in finding the fingerprinting script, go to the third point; if not, execute the webpage normally without making any modifications in response.

iii. **Modifies the values in the response:**
When our extension found the fingerprinting script, it immediately changed its parameters and sent a response to the website with dummy values. This modification process is very fast, and it won't impact the speed of the website. After sending this modified data, the Browser Fingerprint Extension loads the website successfully into the browser.

B. Backend Architecture of Browser Fingerprint Defender

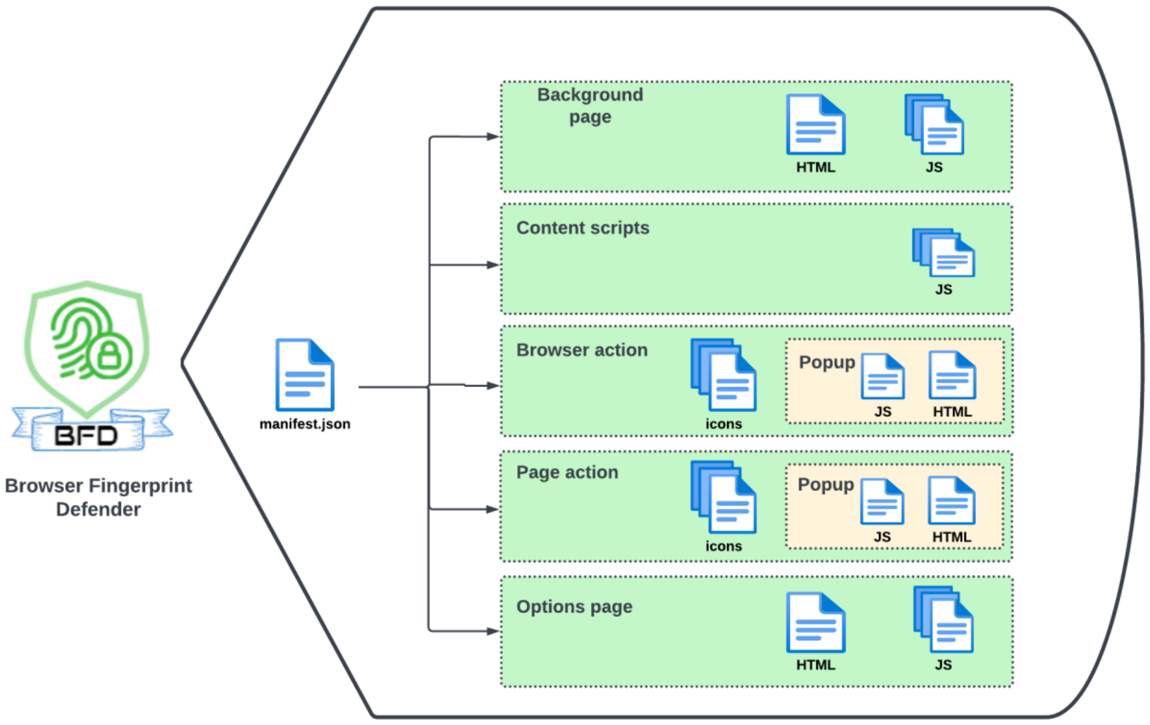


Figure 3: Overall Backend Architecture of Browser Fingerprint Defender

The above Figure 3 shows the programming architecture of our extension. This extension is divided into six sections, which you will see one by one below:

i. manifest.json

This is a JSON file in the browser extension which tell the browser regarding which components need to pick up from which place. Any browser needs a manifest file for the purpose of displaying the extension icon on the Home Screen prompt. The name of extension, all of their logos, and other information regarding our extension is provided by JSON file to the browser. The manifest.json file contains some important information about extensions, such as the name of the extension, which icon should be used for which purpose, which URL or file should be opened after running the extension, and a number of other specific details.

ii. Background Page:

The Background scripts are one of the high secure sections of the any Google Chrome Extension environment when there is a case of logging in and connecting to the API or serve. Background scripts are totally dependent on plugins, and the reason behind that is that if the plugin is installed, then scripts can be run in the background very easily.

iii. Content Scripts:

The Content-scripts.js type of files is most probably used to put additional functionality into browser extensions. If we need to add some extra features to the extension, then it is not possible to add everything in the bagroud.js file; that’s why the content-scripts.js file is used. If we need to add some extra functions to the extension, then we can create multiple content-scripts.js files, but then we just need to mention those files in the manifest.json file with the link. It’s useful for When we add those URLs to the manifest.json file, then connecting with a page’s DOM becomes very easy in every way.

iv. popup.html / popup.js
The popup.html page is the graphical user interface (GUI) that appears when we click on the browser extension icon that appears in the extension menu. This popup shows all available functionality of extension and make it easy to control at single place, rather controlling from options.html.

v. Browser Action:
The Browser action section contains JavaScript files, which are utilized for randomizing the popup attributes. The browser action section is also mentioned as vendor.js in our extension source code. This section of JavaScript also ensures the integrity of extensions. The source code for this is implanted inside the "script" tag and located on the first page of the extension, prior to all HTML tags. This section also contains actionable icons of extensions; for example, when an extension is active, it shows in green, and it shows in gray when it is in inactive mode. Whichever icons are shown in extension bar, those got loaded from this section. This section is also linked to the popup.js file, which handles the frontend GUI of the extension.

vi. Page Action:
This section having the setting page icons and JavaScript ’s for the popup windows menu. Any menu pops up after clicking on extension icon that all scripts are written here. Also, the icons required for all extension menu pages will also get called from here.

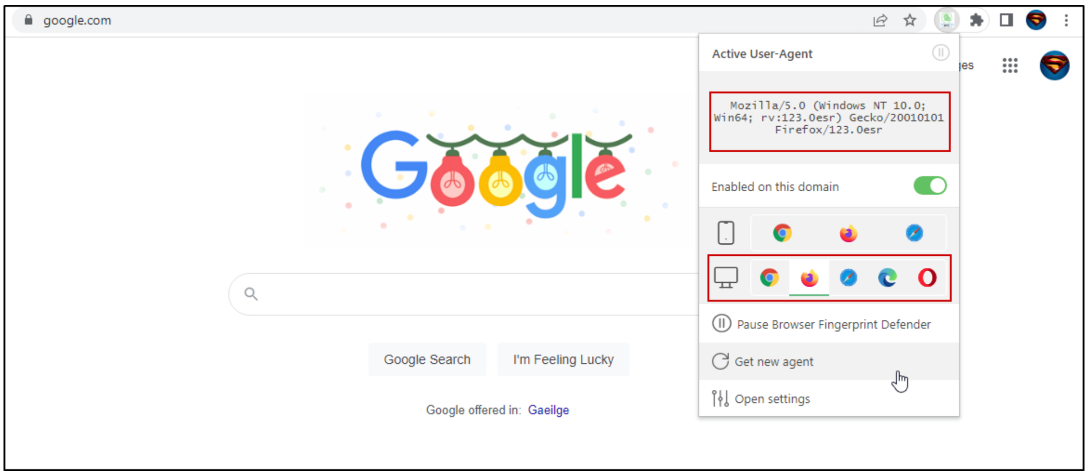
vii. Option Page:
This option page has two files, options.html and options.js, which show the extra setting menu for extensions. When user opens option page then there are lot of extension configuration options available. The options.html file contains a GUI for the option menu, and the option.js file contains the scripts required for the option page.

5 Implementation

The Browser Fingerprint Defender extension was implemented for the Google Chrome browser to protect users' privacy when they are surfing the internet. In the above Figure 2 and Figure 3 we saw the overall functionality as well as backend architecture of Browser Fingerprint Defender. With its help, we understand the workflow and implementation of the extension. Now we will see the actual implementation part of our extension with the help of the following diagrams and source code.

A. Dummy Parameter Generation:

As we can see in Figure 4: Dummy Firefox parameters are generated by Browser Fingerprint Defender. The extension is active, and in the browser selection section, we have selected Firefox only, although all other options are deselected. That’s why, as we can see in the first highlighted area, our extension generates dummy parameters for Firefox. In this case every time when page gets refreshed or user clicked on "Get new agent" button then our extension will only give the parameters of Firefox, until and unless user change it by their own. When this extension is enabled, a dummy fingerprint will be sent to every website that tries to fingerprint the user.



Active User-Agent

```
Mozilla/5.0 (Windows NT 10.0; Win64; rv:123.0esr) Gecko/20100101 Firefox/123.0esr
```

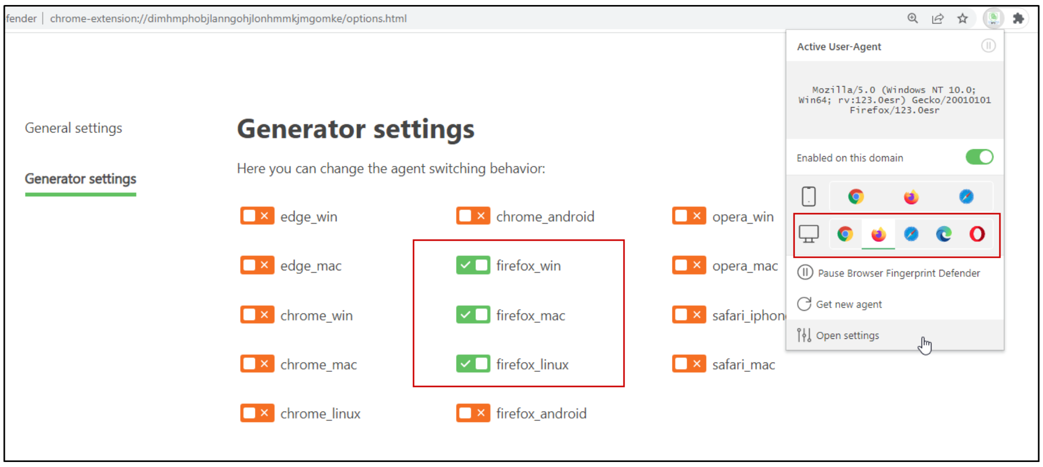
Enabled on this domain ✓

Figure 4: Dummy Firefox parameters is generated by Browser Fingerprint Defender

B. Use fingerprints of multiple browsers:

i. Selecting multiple browser fingerprints from GUI:

Browser fingerprint defender provides the user with the functionality of using multiple browser fingerprints after every page refresh. These multi browser fingerprints provide an extra layer of protection to your privacy and make it almost impossible for a tracker to understand browser patterns. As we can see in Figure 5, in the highlighted area on the right side, which shows the active browsers for dummy parameters, Also, if we click on "open settings menu," we'll get the Generator settings page, which also shows the menu to enable or disable the specific browsers. All these options give concrete uniqueness to the user's fingerprints.


fender | chrome-extension://dimhmphobjlanngohjlonhmmkjmgomke/options.html
General settings
Generator settings
Pause Browser Fingerprint Defender
Get new agent
Open settings

Generator settings

Here you can change the agent switching behavior:

✗ edge_win	✗ chrome_android	✗ opera_win
✗ edge_mac	✓ firefox_win	✗ opera_mac
✗ chrome_win	✓ firefox_mac	✗ safari_iphon
✗ chrome_mac	✓ firefox_linux	✗ safari_mac
✗ chrome_linux	✗ firefox_android	

Active User-Agent

```
Mozilla/5.0 (Windows NT 10.0; Win64; rv:123.0esr) Gecko/20100101 Firefox/123.0esr
```

Enabled on this domain ✓

Figure 5: Manually configuring the of Browser Fingerprint Defender

ii. Backend working Architecture of Browser Fingerprint Defender

To implement multiple browser fingerprint models, we have designed the backend architecture for manipulation of values, which is shown below in Figure 6: Random Agent Assigning Workflow for Browser Fingerprint Defender When the user selects a browser for fingerprinting or chooses multiple browsers like Opera, Firefox, Edge, or Safari for a dummy fingerprint, the workflow will go as follows: As we saw in Figure 4, users have selected Firefox, as we can see in Figure 6. To create a dummy fingerprint for Firefox, have four different operating systems available (Linux, Mac, Windows, and Android), as seen in Figure 7. Our code is designed in such a way that it will randomly choose any operating system from a selection. After selecting an operating system, it randomly fetches data from two tables, Random Profile, and System Configuration, which are relevant to the selected operating system. In this Random Profile table, we have added basic system details like date, language, screen resolution, etc. that are necessary for every request of browser. Secondly, in the System Configuration table, it has operating system parameters. If we have selected Linux in the first place, then the System Configuration method automatically fetches a different value for every parameter.

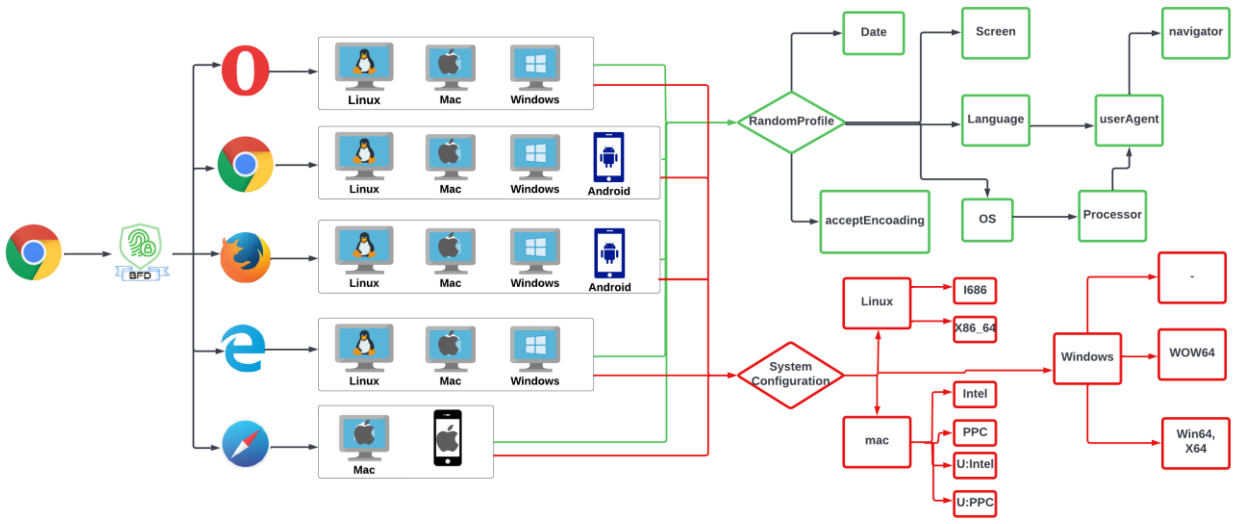
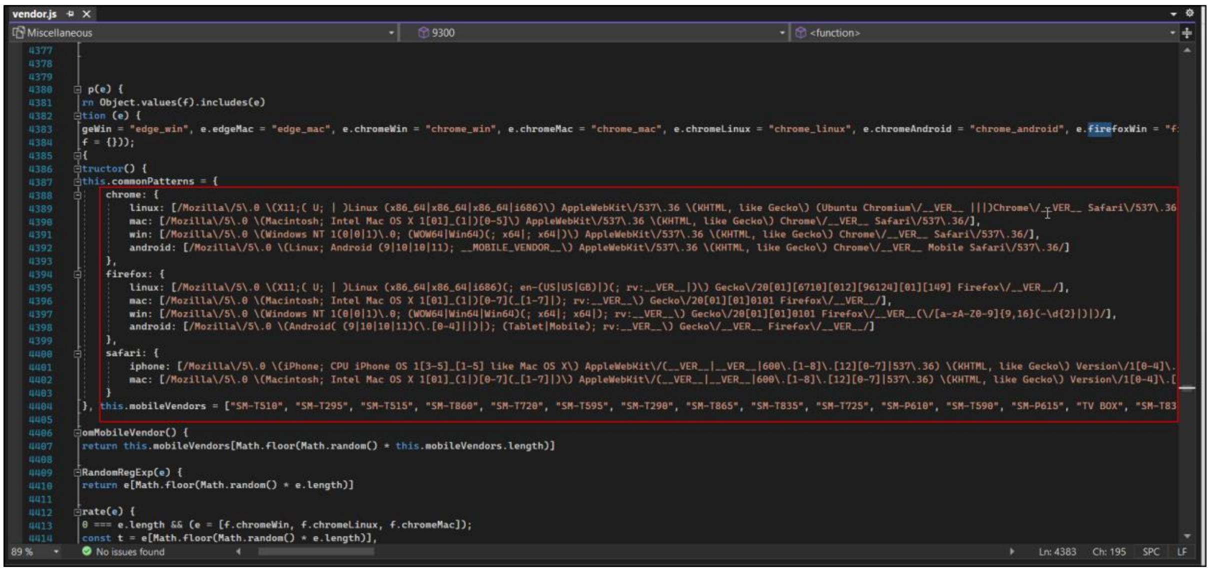


Figure 6: Random Agent Assigning workflow of Browser Fingerprint Defender

iii. Source code for manually selecting required fingerprints of browsers.

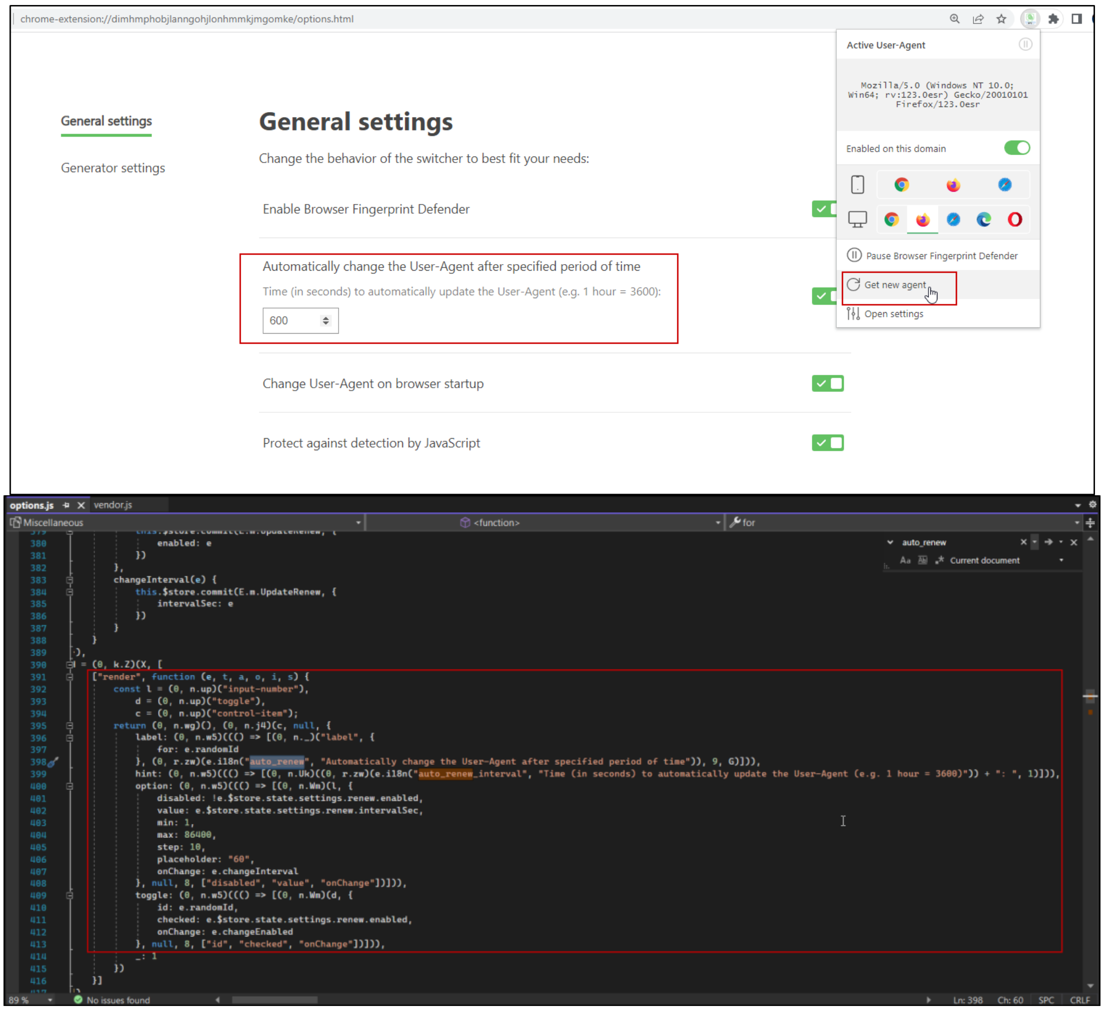


```



 p(e) {
rn Object.values(f).includes(e)
tion (e) {
geWin = "edge_win", e.edgeMac = "edge_mac", e.chromeWin = "chrome_win", e.chromeMac = "chrome_mac", e.chromeLinux = "chrome_linux", e.chromeAndroid = "chrome_android", e.firefoxWin = "f
f = {}));
{
tructor() {
this.commonPatterns = {
    chrome: {
        linux: [/Mozilla\/5\.0 \(X11;( U; | )Linux (x86_64|x86_64|x86_64|i686)\) AppleWebKit\/537\.36 \(KHTML, like Gecko\) (Ubuntu Chromium\/__VER__ |||)Chrome\/__VER__ Safari\/537\.36
        mac: [/Mozilla\/5\.0 \(Macintosh; Intel Mac OS X 1[01]_(1|)[0-5]\) AppleWebKit\/537\.36 \(KHTML, like Gecko\) Chrome\/__VER__ Safari\/537\.36/],
        win: [/Mozilla\/5\.0 \(Windows NT 1(0|0|1)\.0; (WOW64|Win64)(; x64|; x64|)\) AppleWebKit\/537\.36 \(KHTML, like Gecko\) Chrome\/__VER__ Safari\/537\.36/],
        android: [/Mozilla\/5\.0 \(Linux; Android (9|10|10|11); __MOBILE_VENDOR__\) AppleWebKit\/537\.36 \(KHTML, like Gecko\) Chrome\/__VER__ Mobile Safari\/537\.36/]
    },
    firefox: {
        linux: [/Mozilla\/5\.0 \(X11;( U; | )Linux (x86_64|x86_64|i686)(; en-(US|US|GB)|)(; rv:__VER__|)\) Gecko\/20[01][6710][012][96124][01][149] Firefox\/__VER__/],
        mac: [/Mozilla\/5\.0 \(Macintosh; Intel Mac OS X 1[01]_(1|)[0-7](_[1-7]|); rv:__VER__\) Gecko\/20[01][01]0101 Firefox\/__VER__/],
        win: [/Mozilla\/5\.0 \(Windows NT 1(0|0|1)\.0; (WOW64|Win64|Win64)(; x64|; x64|); rv:__VER__\) Gecko\/20[01][01]0101 Firefox\/__VER__(\/[a-zA-Z0-9]{9,16}(-\d{2}|)|)/],
        android: [/Mozilla\/5\.0 \(Android( (9|10|10|11)(\.[0-4]||)|); (Tablet|Mobile); rv:__VER__\) Gecko\/__VER__ Firefox\/__VER__/]
    },
    safari: {
        iphone: [/Mozilla\/5\.0 \(iPhone; CPU iPhone OS 1[3-5]_[1-5] like Mac OS X\) AppleWebKit\/(__VER__|__VER__|600\.[1-8]\.[12][0-7]|537\.36) \(KHTML, like Gecko\) Version\/1[0-4]\.
        mac: [/Mozilla\/5\.0 \(Macintosh; Intel Mac OS X 1[01]_(1|)[0-7](_[1-7]|)\) AppleWebKit\/(__VER__|__VER__|600\.[1-8]\.[12][0-7]|537\.36) \(KHTML, like Gecko\) Version\/1[0-4]\.[
    }
}, this.mobileVendors = ["SM-T510", "SM-T295", "SM-T515", "SM-T860", "SM-T720", "SM-T595", "SM-T290", "SM-T865", "SM-T835", "SM-T725", "SM-P610", "SM-T590", "SM-P615", "TV BOX", "SM-T83

omMobileVendor() {
return this.mobileVendors[Math.floor(Math.random() * this.mobileVendors.length)]

RandomRegExp(e) {
return e[Math.floor(Math.random() * e.length)]

rate(e) {
0 === e.length && (e = [f.chromeWin, f.chromeLinux, f.chromeMac]);
const t = e[Math.floor(Math.random() * e.length)],
```

Figure 7: Source code for manually selecting required fingerprints of browsers.

As you can see in Figure 7. We have implemented the code for automatically selecting the fingerprint list of devices. That’s why we created a devices list and its exact configuration.

C. Change the fingerprint after specific period



chrome-extension://dimhmphobjlanngohjlonhmmkjmgomke/options.html

General settings

Generator settings

General settings

Change the behavior of the switcher to best fit your needs:

Enable Browser Fingerprint Defender

Automatically change the User-Agent after specified period of time

Time (in seconds) to automatically update the User-Agent (e.g. 1 hour = 3600): 600

Change User-Agent on browser startup

Protect against detection by JavaScript

Active User-Agent

Mozilla/5.0 (Windows NT 10.0; Win64; rv:123.0esr) Gecko/20010101 Firefox/123.0esr

Enabled on this domain

Pause Browser Fingerprint Defender

Get new agent

Open settings

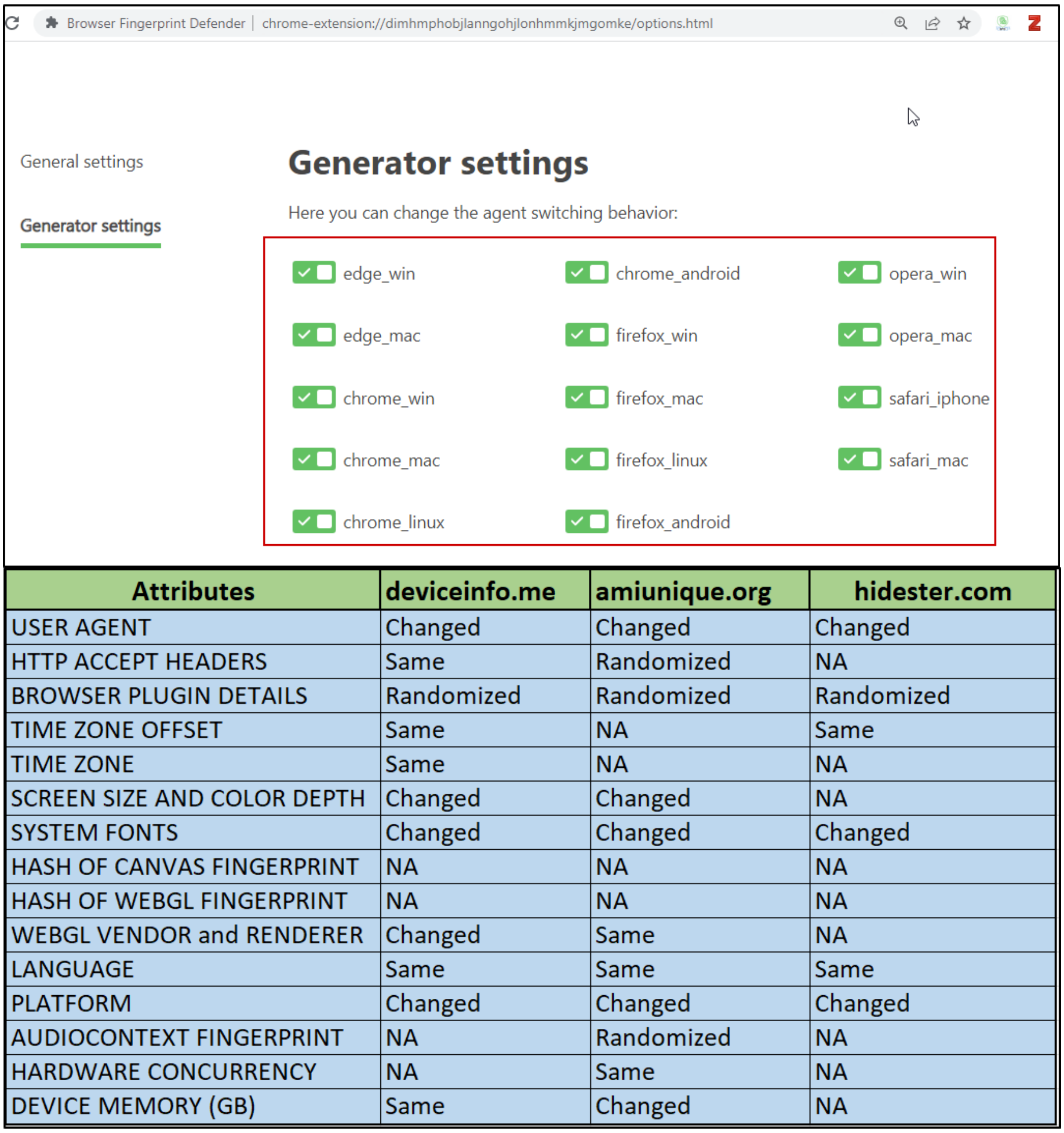
```
            enabled: e
        })
    },
    changeInterval(e) {
        this.$store.commit(E.m.UpdateRenew, {
            intervalSec: e
        })
    }
  }
}),
l = (0, k.Z)(X, [
  ["render", function (e, t, a, o, i, s) {
      const l = (0, n.up)("input-number"),
          d = (0, n.up)("toggle"),
          c = (0, n.up)("control-item");
      return (0, n.wg)(), (0, n.j4)(c, null, {
          label: (0, n.w5)((() => [(0, n._)("label", {
              for: e.randomId
          }, (0, r.zw)(e.i18n("auto_renew", "Automatically change the User-Agent after specified period of time")), 9, G)])),
          hint: (0, n.w5)((() => [(0, n.Uk)((0, r.zw)(e.i18n("auto_renew_interval", "Time (in seconds) to automatically update the User-Agent (e.g. 1 hour = 3600)")) + ": ", 1)])),
          option: (0, n.w5)((() => [(0, n.Wm)(l, {
              disabled: !e.$store.state.settings.renew.enabled,
              value: e.$store.state.settings.renew.intervalSec,
              min: 1,
              max: 86400,
              step: 10,
              placeholder: "60",
              onChange: e.changeInterval
          }, null, 8, ["disabled", "value", "onChange"])])),
          toggle: (0, n.w5)((() => [(0, n.Wm)(d, {
              id: e.randomId,
              checked: e.$store.state.settings.renew.enabled,
              onChange: e.changeEnabled
          }, null, 8, ["id", "checked", "onChange"])])),
          _: 1
      })
  }]
```

Figure 8: GUI and Source code for generating unique fingerprint after specific time.

The best way to stay anonymous from the trackers is to change your identity after a specific period. As we can see in Figure 8 of the GUI, we have provided the option to automatically change the identity after a decided time. Similarly, we can see the code for assigning the time.

6 Evaluation

A. Extension result on different websites.


Browser Fingerprint Defender | chrome-extension://dimhmphobjlanngohjlonhmmkjmgomke/options.html

General settings

Generator settings

Generator settings

Here you can change the agent switching behavior:

✓ edge_win	✓ chrome_android	✓ opera_win
✓ edge_mac	✓ firefox_win	✓ opera_mac
✓ chrome_win	✓ firefox_mac	✓ safari_iphone
✓ chrome_mac	✓ firefox_linux	✓ safari_mac
✓ chrome_linux	✓ firefox_android	

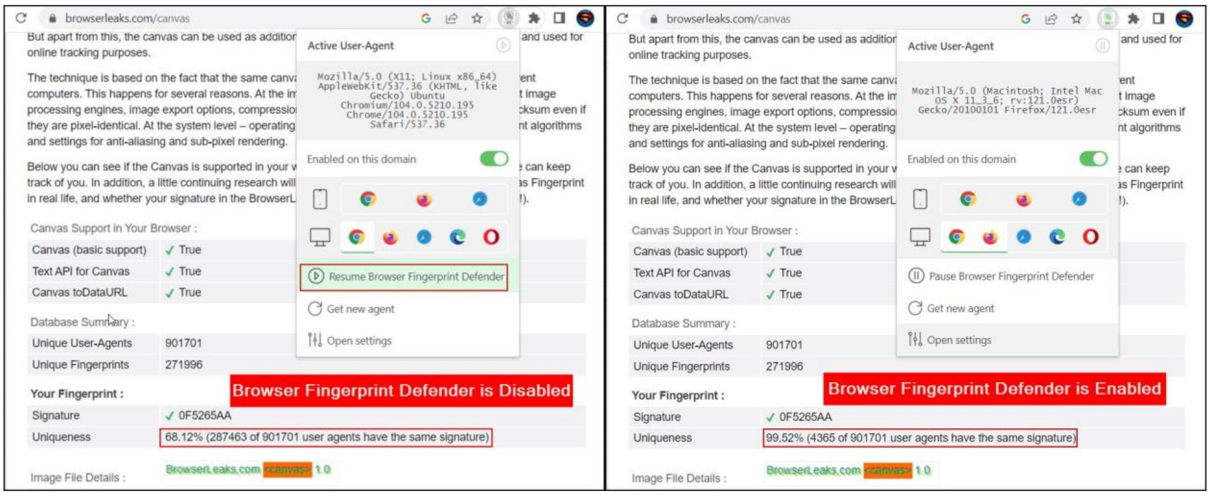
Attributes	deviceinfo.me	amiunique.org	hidester.com
USER AGENT	Changed	Changed	Changed
HTTP ACCEPT HEADERS	Same	Randomized	NA
BROWSER PLUGIN DETAILS	Randomized	Randomized	Randomized
TIME ZONE OFFSET	Same	NA	Same
TIME ZONE	Same	NA	NA
SCREEN SIZE AND COLOR DEPTH	Changed	Changed	NA
SYSTEM FONTS	Changed	Changed	Changed
HASH OF CANVAS FINGERPRINT	NA	NA	NA
HASH OF WEBGL FINGERPRINT	NA	NA	NA
WEBGL VENDOR and RENDERER	Changed	Same	NA
LANGUAGE	Same	Same	Same
PLATFORM	Changed	Changed	Changed
AUDIOCONTEXT FINGERPRINT	NA	Randomized	NA
HARDWARE CONCURRENCY	NA	Same	NA
DEVICE MEMORY (GB)	Same	Changed	NA

Figure 9: Extension result on different websites

On multiple platforms, we tested the performance of our Browser Fingerprint Defender extension and found that it worked excellently with every website. In the above Figure 9: Extension results on different websites, we can see the comparison between different websites like deviceinfo.me, amiunique.org, and hidester.com. We have carefully analyzed and noted the results before and after enabling the extension, as you can see in Figure 9, which shows the three different results from different domains. In the table we can see list of attributes then result got on different domains for attribute. We can observe in the below figure that every website has their own different method for fingerprinting, but still, system attributes on every domain are changed. After analyzing the results of all domains, we can confidently say that the Browser Fingerprint Defender extension is working correctly and providing protection against online trackers.

The negative side of the extension result is that some of the important parameters, such as device memory and hardware concurrency, remain the same every time. If a user uses the fingerprint of the same browser for a long time, for example, only Firefox or only Chrome, then the advertisers can identify the pattern of data and can identify you. To avoid this chance of getting spotted, a user has to change his dummy fingerprinting browser after a specific period of time. For example, a user must change dummy data from the Chrome browser to Firefox and vice versa. This method can make the tracker difficult to identify the user.

B. Checking the efficiency of Browser Fingerprint Defender



But apart from this, the canvas can be used as addition … and used for online tracking purposes.

The technique is based on the fact that the same canv… ent computers. This happens for several reasons. At the im… t image processing engines, image export options, compressio… cksum even if they are pixel-identical. At the system level – operating … nt algorithms and settings for anti-aliasing and sub-pixel rendering.

Below you can see if the Canvas is supported in your w… can keep track of you. In addition, a little continuing research will … as Fingerprint in real life, and whether your signature in the BrowserL… !).

```
Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like
Gecko) Ubuntu
Chromium/104.0.5210.195
Chrome/104.0.5210.195
Safari/537.36
```

Canvas Support in Your Browser :

Canvas (basic support)	✓ True
Text API for Canvas	✓ True
Canvas toDataURL	✓ True

Database Summary :

Unique User-Agents	901701
Unique Fingerprints	271996

Your Fingerprint :

Signature	✓ 0F5265AA
Uniqueness	68.12% (287463 of 901701 user agents have the same signature)

Image File Details : BrowserLeaks.com <canvas> 1.0

```
Mozilla/5.0 (Macintosh; Intel Mac
OS X 11_3_6; rv:121.0esr)
Gecko/20100101 Firefox/121.0esr
```

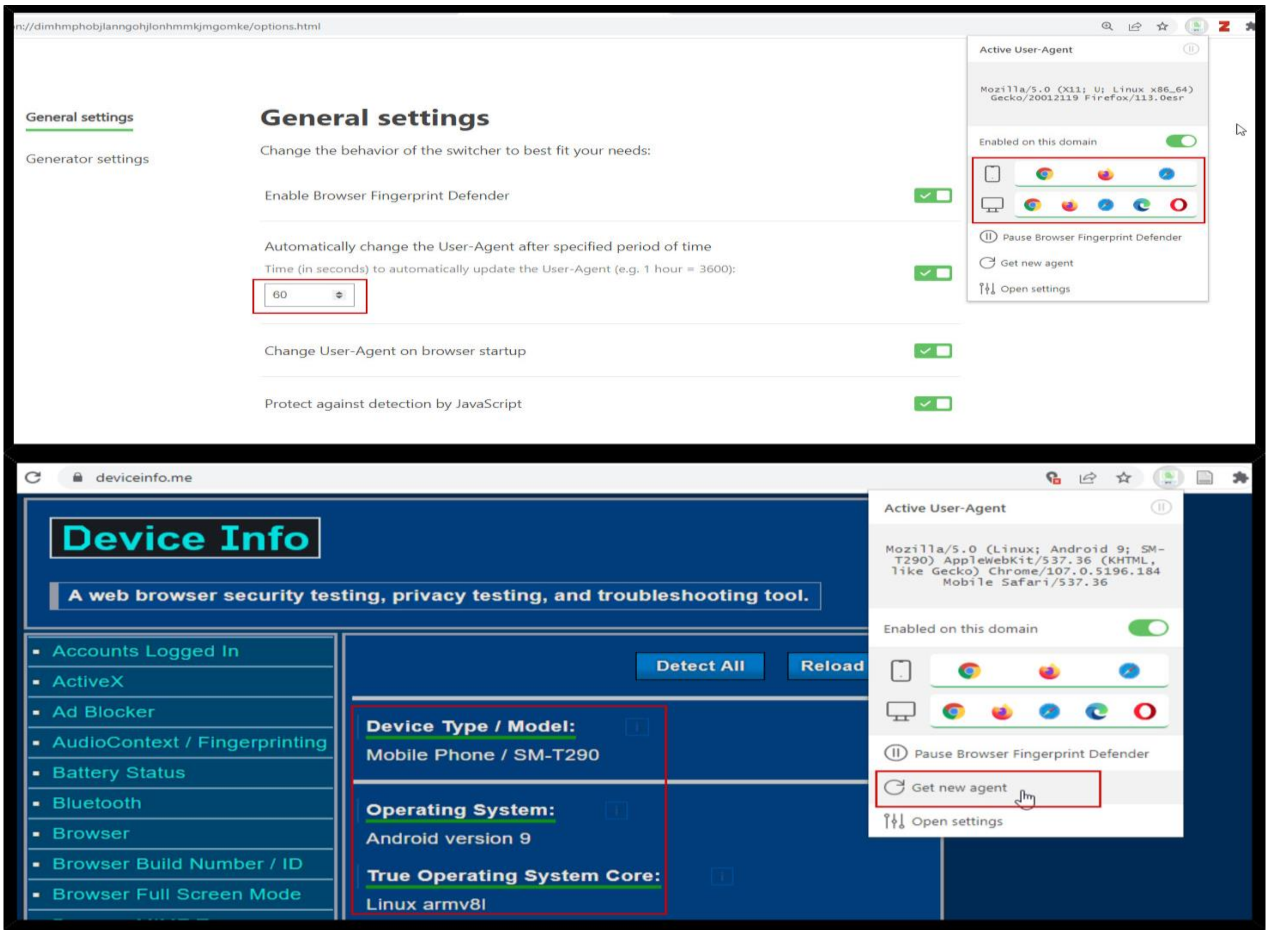
Signature	✓ 0F5265AA
Uniqueness	99.52% (4365 of 901701 user agents have the same signature)

Figure 10: Efficiency of Browser Fingerprint Defender

To check the efficiency of our extension, first we completely disable the extension then open the https://browserleaks.com/canvas [27] to check the original fingerprint of browser. As we can see in Figure 10, we have merged the two images. On the left side of the image, the browser extension is disabled, and we get the uniqueness of our fingerprint as 287463 of 901701 user agents have the same signature as us, and our signature is 68.12% unique. which is easily identifiable. Then as we can see on the right side of the image, browser extension is enabled, and we got uniqueness of our fingerprint as 4365 of 901701 user agent have same signature as us and our signature is 99.52% unique. which is not easily identifiable. We can easily spot the result. When the extension is off, we are 68.12% unique, and after enabling the extension, we become 99.52% unique. Hence, it is proven that after enabling the extension, we get a unique as well as unidentifiable fingerprint, and our privacy is successfully protected.

C. Performance Testing:

We have implemented many features in Browser Fingerprint Defender like automatically change the user agent after specific time, protect against detection by JavaScript, auto start on browser start up and as we can see in right side highlighted area, we have enabled all types of devices fingerprint, keep automatic user agent time 60 seconds and many more. The reason behind enabling all possible features is to do performance testing.



n://dimhmphobjlanngohjlonhmmkjmgomke/options.html

General settings

Generator settings

General settings

Change the behavior of the switcher to best fit your needs:

Enable Browser Fingerprint Defender ✓

Automatically change the User-Agent after specified period of time ✓

Time (in seconds) to automatically update the User-Agent (e.g. 1 hour = 3600):

60

Change User-Agent on browser startup ✓

Protect against detection by JavaScript ✓

Active User-Agent

Mozilla/5.0 (X11; U; Linux x86_64) Gecko/20012119 Firefox/113.0esr

Enabled on this domain

Pause Browser Fingerprint Defender

Get new agent

Open settings

deviceinfo.me

Device Info

A web browser security testing, privacy testing, and troubleshooting tool.

- Accounts Logged In
- ActiveX
- Ad Blocker
- AudioContext / Fingerprinting
- Battery Status
- Bluetooth
- Browser
- Browser Build Number / ID
- Browser Full Screen Mode

Detect All

Reload

Device Type / Model:

Mobile Phone / SM-T290

Operating System:

Android version 9

True Operating System Core:

Linux armv8l

Active User-Agent

Mozilla/5.0 (Linux; Android 9; SM-T290) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5196.184 Mobile Safari/537.36

Enabled on this domain

Pause Browser Fingerprint Defender

Get new agent

Open settings

Figure 11: Performance Testing

As we can see in the above figure 11, there are two sections. In the top section, we can see that we have enabled all parameters, which will force extension to perform under pressure and determine extension capacity. Many times, when we increase the load on the extension, it won't work properly and reveal the user's identity. As we can see in the bottom figure, we have continuously clicked on "Get a new agent," which forces Extension to use its full capability. After the continued load on extension, we observe that it is giving accurate results, as we can see in the bottom of figure 11. As a conclusion, our extension won't reveal user identity while working under pressure.

7 Conclusion and Future Work

The research reported in this paper was inspired by the increasing privacy risk posed by browser fingerprinting. As we studied previous research papers on browser fingerprint, we understand that user don't have direct control over the browser fingerprinting and even if user aware about browser fingerprint still he can't do much to protect himself, because there is no full proof solution available in the market. We found that because consumers have no direct control over it, it represented the biggest risk to consumer privacy than cookie-based user tracking.

Additionally, we have observed that this user tracking method is rapidly being used for online user monitoring, even in the absence of a static IP address or cookie, by online advertising companies. To overcome this issue, we developed the "Browser Fingerprint Defender" extension, which takes full control by identifying fingerprinting attempts and protecting users' privacy from online tracking websites. The primary goal of the extension was to educate browser users about the existence of browser fingerprinting and to provide them with some level of protection against it.

Browser Fingerprint Defender intercepts the JavaScript request which contains the call of system parameters for fingerprinting and replaces those values with dummy parameters. On every fingerprinting request, Browser Fingerprint Defender does not modify any other parts of JavaScript or HTML but provides dynamic values to make fingerprints more unique. This parameter modification method removes the uniqueness of the user and pretends to be someone else. After enabling this extension, the fingerprint of the user is totally different from the real one, and it won't have any effect on the performance of the browser. This extension defends the user’s privacy and makes browsing more private.

In our research, we measured only the passive fingerprinting and JavaScript-based object fingerprinting performed by tracking websites. Despite that, there are a variety of distinct fingerprinting methods available in the market that we haven’t resolved in our research, like audio fingerprinting and media device fingerprinting. Due to a lack of resources and time, it was not feasible to address each of these fingerprinting methods in a single research project.

The majority of the values or parameters we have used in our extension are taken from online fingerprint detection sites like deviceinfo.me, amiunique.org, hidester.com, or browserleaks.com. We considered those parameters because we only saw them on individual sites. However, there is a chance that tracking websites are using more advanced techniques and using parameters that we don’t know.

Furthermore, there are some parameters which we are aware about, but still, we are unable to include them into the user agent. Case-in-point, parameters like Hardware Concurrency, Device Memory and time zone will make fingerprint extremely unique. We have tried to implement those into this extension, but due to extreme technicality and time constraints, we are unable to do so.

8 References

[1] P. Laperdrix, W. Rudametkin, and B. Baudry, “Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints,” in *2016 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, May 2016, pp. 878–894. doi: 10.1109/SP.2016.57.

[2] G. Pugliese, C. Riess, F. Gassmann, and Z. Benenson, “Long-Term Observation on Browser Fingerprinting: Users’ Trackability and Perspective,” *Proc. Priv. Enhancing Technol.*, vol. 2020, pp. 558–577, May 2020, doi: 10.2478/popets-2020-0041.

[3] T. Hupperich, D. Tatang, N. Wilkop, and T. Holz, “An Empirical Study on Online Price Differentiation,” in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, New York, NY, USA, Mar. 2018, pp. 76–83. doi: 10.1145/3176258.3176338.

[4] W. Jiang, X. Wang, X. Song, Q. Liu, and X. Liu, “Tracking your browser with high-performance browser fingerprint recognition model,” *China Commun.*, vol. 17, no. 3, pp. 168–175, Mar. 2020, doi: 10.23919/JCC.2020.03.014.

[5] Krishna. V. Nair and E. RoseLalson, “The Unique Id’s you Can’t Delete: Browser Fingerprints,” in *2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research (ICETIETR)*, Jul. 2018, pp. 1–5. doi: 10.1109/ICETIETR.2018.8529040.

[6] P. N. Bahrami, U. Iqbal, and Z. Shafiq, “FP-Radar: Longitudinal Measurement and Early Detection of Browser Fingerprinting.” arXiv, Dec. 14, 2021. Accessed: Aug. 01, 2022. [Online]. Available: http://arxiv.org/abs/2112.01662

[7] S. Luangmaneerote, E. Zaluska, and L. Carr, “Inhibiting Browser Fingerprinting and Tracking,” in *2017 ieee 3rd international conference on big data security on cloud (bigdatasecurity), ieee international*

conference on high performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids), May 2017, pp. 63–68. doi: 10.1109/BigDataSecurity.2017.40.

[8] "What is Browser Fingerprinting & How Does it Work?," *SEON*, May 11, 2020. https://seon.io/resources/browser-fingerprinting/ (accessed Jul. 29, 2022).

[9] N. Kaur, S. Azam, K. Kannoorpatti, K. C. Yeo, and B. Shanmugam, "Browser Fingerprinting as user tracking technology," in *2017 11th International Conference on Intelligent Systems and Control (ISCO)*, Jan. 2017, pp. 103–111. doi: 10.1109/ISCO.2017.7855963.

[10] O. Starov and N. Nikiforakis, "XHOUND: Quantifying the Fingerprintability of Browser Extensions," in *2017 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA, May 2017, pp. 941–956. doi: 10.1109/SP.2017.18.

[11] J. P. Johnson, "Targeted advertising and advertising avoidance," *RAND J. Econ.*, vol. 44, no. 1, pp. 128–144, 2013, doi: 10.1111/1756-2171.12014.

[12] N. M. Al-Fannah and C. Mitchell, "Too little too late: can we control browser fingerprinting?," *J. Intellect. Cap.*, vol. 21, no. 2, pp. 165–180, Jan. 2020, doi: 10.1108/JIC-04-2019-0067.

[13] U. Iqbal, S. Englehardt, and Z. Shafiq, "Fingerprinting the Fingerprinters: Learning to Detect Browser Fingerprinting Behaviors," presented at the 2021 IEEE Symposium on Security and Privacy (SP), May 2021, pp. 1143–1161. doi: 10.1109/SP40001.2021.00017.

[14] "Mitigating Browser Fingerprinting in Web Specifications." https://w3c.github.io/fingerprinting-guidance/ (accessed Jul. 30, 2022).

[15] A. Rasool, Z. Jalil, and R. J. O. Computing, "A Review of Web Browser Forensic Analysis Tools and Techniques," pp. 15–21, Jun. 2020, doi: 10.1111/RpJC.2020.DOI.

[16] P. Laperdrix, N. Bielova, B. Baudry, and G. Avoine, "Browser Fingerprinting: A Survey," *ACM Trans. Web*, vol. 14, no. 2, pp. 1–33, Apr. 2020, doi: 10.1145/3386040.

[17] G. Gulyás, R. Schulcz, and S. Imre, "Comprehensive Analysis of Web Privacy and Anonymous Web Browsers: Are Next Generation Services Based on Collaborative Filtering?," p. 15.

[18] 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*, Berlin, Heidelberg, 2012, pp. 31–46. doi: 10.1007/978-3-642-29615-4_4.

[19] Y. Cao, S. Li, and E. Wijmans, "(Cross-)Browser Fingerprinting via OS and Hardware Level Features," in *Proceedings 2017 Network and Distributed System Security Symposium*, San Diego, CA, 2017. doi: 10.14722/ndss.2017.23152.

[20] S. Englehardt and A. Narayanan, "Online Tracking: A 1-million-site Measurement and Analysis," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, Oct. 2016, pp. 1388–1401. doi: 10.1145/2976749.2978313.

[21] "What is it? - NoScript: Own Your Browser!" https://noscript.net/ (accessed Dec. 13, 2022).

[22] "The Design and Implementation of the Tor Browser [DRAFT]." https://2019.www.torproject.org/projects/torbrowser/design/#fingerprinting-linkability (accessed Dec. 13, 2022).

[23] 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*, May 2013, pp. 541–555. doi: 10.1109/SP.2013.43.

[24] C. F. Torres, H. Jonker, and S. Mauw, "FP-Block: Usable Web Privacy by Controlling Browser Fingerprinting," in *Computer Security -- ESORICS 2015*, Cham, 2015, pp. 3–19. doi: 10.1007/978-3-319-24177-7_1.

[25] P. LLC, "PsyberAnalytix LLC," *PsyberAnalytix LLC*. https://psyberanalytix.com/cybersecurity (accessed Nov. 08, 2022).

[26] "Bodleian Libraries | ORA." https://ora.ox.ac.uk/objects/uuid:b80bf744-65a4-4b48-bf0e-1b159e029df8 (accessed Nov. 08, 2022).

[27] "Canvas Fingerprinting," *BrowserLeaks*. https://browserleaks.com/canvas (accessed Aug. 02, 2022).