Fingerprint Defense in the Modern Web

Author: Issac Kim, issac.lk.kim@pm.me
Advisor: *Dr. Tim Proffitt*

Accepted: *October 23, 2023*

Abstract

Modern websites use advanced tracking and fingerprinting methods to identify users across the internet and build an entire profile of their interests and habits. Increasingly, privacy on the internet is becoming a relic of the past, and without countermeasures, internet companies and data brokers will continue to track everything a person does across the internet. For the less technically inclined, browser developers have begun implementing privacy by default measures, such as those in Apple's Safari or Firefox, yet represent only the bare minimum to avoid tracking. The most common browser worldwide, Chrome, is developed by one of the largest web advertising companies, Google, so there is no incentive to provide built-in web privacy measures for their users. As such, various privacy-enhancing measures are analyzed and proposed to regain privacy for the everyday user.

1. Introduction

The web cookie has always been a constant throughout the internet and is the first thing that comes to mind when "web tracking" is mentioned. However, in recent years, companies have developed methods of tracking users across the internet that well exceed the capabilities of a cookie. With the introduction of web technologies such as HTML5 and WebRTC, companies have developed advanced fingerprinting techniques that utilize these technologies which can build unique profiles on users to track them across the entirety of the internet. Every browser engine is different and handles JavaScript and HTML5 canvas elements differently, leading to metrics that these browsers inherently leak. This allows unique identifiers to be built that differentiate one user from another.

In this paper, the researcher aims to determine whether it is feasible for the "average" user to take basic privacy-enhancing measures to avoid fingerprinting and tracking. While some actions may include disabling modern web features such as Canvas or WebRTC, installing hardening scripts and configurations, or disabling JavaScript entirely, these solutions come at the cost of impacting the experience of most websites completely.

2. Research Method

In researching this problem, the research environment will replicate the browsing environments most commonly used by the "average" user across numerous devices and operating systems. This test will include Windows, macOS, Chrome OS, and Linux for desktop platforms, and Android and iOS for mobile platforms. The goal of choosing these specific computing environments is to create a testbed that future researchers can easily replicate. In addition, multiple ISP connections will be used to demonstrate the ability of fingerprinting frameworks to uniquely identify the same users, even across different networks.

2.1. Research Methodology

To prepare each device in the testbed for research and data collection, each device will start from an "out of the box" state, with no customization or modifications to the

operating system. The data collected with this research will involve measuring the difference in how trackable a platform is compared to its unmodified state, in contrast to adding basic privacy-enhancing measures such as ad or content blockers.

2.1.1. Test Variables

The primary test variable that will be modified to collect measurable data will be the state of the web browser and the presence of ad and fingerprinting blocking measures on the browser.

The main assumption made for this research paper is that the "average" user is not likely to make significant changes or modifications to their computing environments. With this assumption in place, the main test variable for each operating system platform will be the default installed browser on each operating system. For built-in browsers, testing will be conducted with Safari on macOS and iOS, Microsoft Edge on Windows, Chrome on Android and Chrome OS, and Firefox on Linux.

2.1.2. Tracking Platforms

Several full-featured demo platforms were utilized to collect data on tracking and fingerprint evasion methods. As most of the web still relies on analytics, scripts, and advertising networks to perform tracking, the platforms used to collect adblocking effectiveness data are AdBlock Tester[1] and D3ward's Test Ad Block[2].

Several web fingerprinting frameworks were selected for testing fingerprinting, as there are various metrics that can be used to generate a unique identifier on a user. These frameworks are FingerprintJS[3], the Electronic Frontier Foundation's Cover Your Tracks[4],

[1] https://adblock-tester.com/

[2] https://d3ward.github.io/toolz/adblock.html

[3] https://fingerprintjs.github.io/fingerprintjs/

[4] https://coveryourtracks.eff.org/

and CreepJS[5]. For more technical details on the reason for selecting these three frameworks, see Section 2.1.4.

2.1.3. Experiments and Testing

The experiment to collect research data on web fingerprinting will involve the test platforms' "identity score"— how identifiable and unique the browser and user are to the tracking and fingerprinting platform. Each of the three fingerprinting frameworks used issue a unique session identifier or use browser features to determine its uniqueness. This identifier allows both the user and the website to identify if their web browser and browsing habits are seen as unique. It can also be used to tell if this uniqueness persists across other WAN connections or sessions with different cookies. The control group will be all test platforms with no changes made to the browser, operating system, or network. The experimental group will include the installation of privacy-enhancing add-ons in the browser to test the effectiveness of anti-tracking and anti-fingerprinting measures.

In addition, identifiers from the tracking platforms will be captured at separate times for each test platform. Between each attempt, website data for each test platform website will be cleared in the browser to ensure no data persists between test runs. For browsers on operating systems that support adblocking and canvas blocking methods, each fingerprinting website will be visited two separate times on different internet connections to demonstrate the effectiveness of adding a fingerprinting blocking add-on.

On mobile platforms, alternative browsers will also be tested if necessary to present a privacy solution for users, as the default browser on Android and iOS presents limitations concerning privacy-enhancing measures. On Android, Chrome does not allow the use of extensions, therefore the Brave browser will be included in testing as the privacy-enhancing alternative. On iOS, web extensions are limited due to Apple's web browser development policy on their mobile platforms. As a result, the testing process for iOS will include installing a web content blocker through the App Store and attempting to

[5] https://abrahamjuliot.github.io/creepjs/

analyze whether Apple's built-in fingerprinting prevention measures in WebKit are sufficient.

2.1.4. Fingerprint Evasion Methods

Traditional ads and analytics methods are easily evaded with ad-blocking extensions. Fingerprinting as a tracking method is a familiar idea, and the techniques behind modern fingerprinting methods are similar to the earlier web.

In 2013, Nikiforakis et al., describe several methods of fingerprinting browsers, including deriving information with Adobe Flash, JavaScript, and CSS or HTML elements. In particular, the authors could differentiate browsers based on their HTML and JavaScript implementation quirks, where each browser would implement specific methods outside of the HTML or JavaScript standard, making them stand out and identifiable.

FingerprintJS and CreepJS both employ this idea today to track users, but in a more advanced fashion, identifying users down to their computers' specific hardware. One of the newer HTML features used by both frameworks to generate an identifying fingerprint is Canvas and WebGL, which are used for consistency across runs on the same hardware, speed, and ease of implementation for developers (Mowery & Shacham, 2012). According to Mowery and Shacham (2012), with canvas fingerprinting, determining the hardware configuration and unique identity of a user is trivial when combining CSS3 web fonts, WebGL, and canvas elements due to the minor quirks and different implementations of same standards between GPU drivers, operating systems, and web rendering engines. FingerprintJS and CreepJS were both selected for testing as they generate identifiers based on canvas, font, and WebGL metrics, and CreepJS additionally features "lie detection" to determine if the browser is spoofing results and other esoteric measures to identify a browser and user. EFF's Cover Your Tracks uses these same metrics to determine uniqueness but does not issue an identifying hash and was selected to serve as a secondary source for determining both ad and analytics plus fingerprint blocking detection. Additionally, many large websites use the commercial

version of FingerprintJS, Fingerprint Pro[6]. While a demo of Fingerprint Pro is also available, this was not used in the research process as the metrics used to determine the identifying hash are not fully documented.

The conclusion drawn is that by obscuring or spoofing the output of these web elements at the browser level, it may be possible to disrupt fingerprinting. Fingerprint frameworks that use Canvas and WebGL assume consistency in rendering elements, thus creating a unique and identifiable record of a web user when combined with other metrics such as the user's browser, IP, OS fonts, and system hardware. This research assumes that by disrupting canvas and WebGL, an unexpected data point will be created that cannot be replicated on each fingerprinting framework, as the browser will render elements differently, and thus appear as a different client or user every time a web page is loaded.

On Chrome-based desktop browsers, fingerprint evasion will be tested with the Fingerprint Spoofing[7] extension, which supports spoofing and randomizing multiple identifying bits of system information. For Firefox, evasion will be tested with the CanvasBlocker[8] extension, which functions similarly to Fingerprint Spoofing in blocking canvas elements and WebGL rendering elements. For Safari, built-in fingerprinting protection comparisons will be made between versions. When research was conducted, macOS Ventura was the latest version, with an older set of protections that have been implemented since 2019. With iOS (and iPadOS), the latest version includes an updated Safari and WebKit version that introduces new fingerprinting and privacy protections.

3. Findings and Discussion

Across all test platforms, adding an ad-blocking addon is enough to defeat non-fingerprinting analytics and web tracking software. On desktop platforms, it is possible to

---

[6] https://fingerprint.com/

[7] https://chrome.google.com/webstore/detail/fingerprint-spoofing/ljdekjlhpjggcjblfgpijbkmpihjfkni

[8] https://addons.mozilla.org/en-US/firefox/addon/canvasblocker/

defeat web fingerprinting techniques by installing additional web feature-blocking add-ons, such as an HTML canvas spoofer or blocker. Due to the limitations of the mobile test platforms and their respective app stores, blocking web features used in fingerprinting either relies on the built-in browser and web engine protections (iOS) or requires an alternative web browser entirely (Android). The screenshots of the captured data and results can be referenced in the Appendix.

3.1. Desktop Platforms

The desktop test platforms include Windows 11, macOS Ventura, and Fedora Linux 38. All operating systems were tested on the researcher's available hardware, and due to time constraints, some testing was performed on virtualized hardware.

3.1.1. Windows 11

Testing on Windows 11 was performed with the default browser, Microsoft Edge, on a Lenovo ThinkPad P1.

Edge provides limited protection against analytics script-based or fingerprint-based tracking in the default install state (see Figures 1 and 2 in the Appendix). As Edge is built on top of the Chromium open-source browser, the results will look similar to a Chrome install on Windows there is limited built-in protection for users against tracking from advertising networks and script-based analytics.

Edge provides no fingerprinting protection for users, as all advanced and newer web technologies report data to websites without modification or spoofing. Some elements of a default Windows install may allow users to blend in, such as the available fonts reported to websites. However, given the wide variety of hardware configurations possible with computers, the hardware configuration reported to websites will allow fingerprinting to maintain a unique hash on each Windows user. For both CreepJS and FingerprintJS, Edge looks like a unique client given the lack of fingerprinting protection (see Figures 3 and 4 in the Appendix).

In InPrivate mode (private browsing mode), Edge carries over Chromium's default minimal privacy protection by restricting third-party cookies. This feature prevents minor protections against ad and analytics-based tracking, but the user can still

be seen as unique due to fingerprinting, with Canvas, WebGL, and audio context elements always appearing the same with every web browsing session (see Figure 5 in the Appendix).

Edge can install any extensions that Chromium-based browsers support, and installing the two recommended extensions (Ublock Origin and Fingerprint Spoofing) significantly increases protection from analytics and fingerprinting, with nearly full coverage from most ad networks (see Figure 6 in the Appendix).

An interesting data point is that Cover Your Tracks believes that neither tracking/analytics ads nor fingerprinting are blocked sufficiently. With ad-blocking, as seen in both ad-block testers, adding an ad-blocking extension is enough to block most tracking networks. For fingerprinting, the Cover Your Tracks tool can generally identify fingerprint evasion and notify the user that their protection is good due to fingerprint randomization (see results from Brave on Android below). Based on the detailed results, Cover Your Tracks sees the mangled canvas and WebGL results from the Fingerprint Spoofing extension. However, instead of detecting this as fingerprinting protection, it reports it as a unique fingerprint instead (see figure 7 in the appendix).

Fingerprint Spoofing effectively defeats FingerprintJS's measures; randomizing elements such as Canvas rendering and fonts are enough to create a unique hash for each session, whether in private browsing or normal mode (see figure 8 in the appendix). In multiple iterations of running Edge in private browsing mode, FingerprintJS returned a unique hash each time. Although this can be considered a "unique" client, no two hashes were the same between multiple networks and browsing sessions and even remained randomized with clearing cookies.

Like all other tested platforms, CreepJS can still uniquely identify the browser and computer, even with the privacy-enhancing extensions installed (see figure 9 in the appendix). A deeper dive into CreepJS can be found in Section 4.2.

3.1.2. macOS Ventura

Testing for macOS was performed on the default browser, Safari. A particular point of interest prior to testing is Apple's built-in privacy protections, as the browser

includes cross-site tracking prevention from analytics networks and web trackers and fingerprint defenses ("Safari Privacy Overview," 2019). While Apple does not specify precisely how the built-in fingerprint defenses operate, they mention that Safari is designed to minimize the data sent to the website by limiting the system configuration details reported; these features were added due to the Safari addon limitations imposed in macOS and iOS in 2019. A key point of Apple's paper that stands out is the idea of "herd immunity," that they attempt to make all of their devices blend in with the larger group of Apple devices used worldwide to prevent fingerprinting on any user individually.

In addition, Apple introduced Lockdown Mode in macOS Ventura (the current version at the time of conducting research), and part of the restrictions in this mode account for extreme web fingerprinting prevention. As this research aims to improve fingerprinting evasion for the average user, testing on Lockdown Mode was not conducted as it disables most web features.

While Safari can protect users from some traditional trackers and analytics methods, it is not as effective as using a separate ad-blocking extension, as seen with other browsers. Even with some trackers blocked, many platforms are still allowed to load scripts on Test Ad Block (see figure 10 in the appendix).

Separate fingerprint evasion extensions are not available for Safari on desktops due to Apple's developer policies on Safari extensions; thus, a secondary objective of testing for Apple's platforms is evaluating Apple's built-in fingerprinting measures. Although the complete details behind these privacy-enhancing measures are not detailed, Apple's WebKit documentation provides some more insight; extra permissions are required to access motion data, webcams and microphones cannot be accessed by WebRTC, fonts reported to websites are limited to only system fonts, and the user agent string does not change between minor updates. To test Apple's goals of obtaining "herd immunity" with browsers between Apple devices blending in, the Cover Your Tracks tool provides insight into how many devices that have tested in the past looked similar to the current session. If Apple's measures are effective, it would be expected that the test platform would report that the browser does not look unique.

Safari on macOS appears unique in testing, with some extra protection issued in private browsing mode due to the lack of cookies or persistent browser data storage. Compared with the other desktop platforms, Cover Your Tracks implies that there is still enough identifying information among all the collected metrics to identify this browser from others in regular browsing mode. As Apple is not entirely transparent about what every built-in fingerprint prevention method does, the only determination is that these measures are not sufficient (see figure 11 in the appendix). Some additional insight into the lack of built-in protection can be seen when compared to the newest version of Safari and WebKit in iOS 17, which introduces additional fingerprinting protections (see section 3.2.2).

In addition, FingerprintJS and CreepJS can identify the browser uniquely, and the likely conclusion from these results is that while Apple has attempted to minimize the data provided to websites by the browser and system, this provides little extra privacy in practice (see figures 12 and 13). As this feature is unique to Apple's WebKit, which is only currently used by Safari, fingerprinting can likely be adapted to look for the quirks that Safari uses to appear less conspicuous and make it identifiable anyway.

That is, if a blind fingerprinting test is performed among browsers, Safari will report information about elements such as WebRTC or hardware details differently than Blink (Chromium, Edge, Chrome, Brave) or Gecko (Firefox) would, so the test can determine that the device is likely some Apple hardware easily. However, one limitation behind this research process is the available testbed of hardware; with only one desktop macOS device available, it is impossible to determine if other Apple desktops can blend in and look similar to the test device used.

To supplement tracking protection for Safari, testing on the ad-block test platforms was performed again after installing the AdGuard extension through the macOS App Store[9]. With the extension development limitations imposed by Apple, Ublock Origin is not available, so the next best alternative is the open-source AdGuard

[9] https://apps.apple.com/us/app/adguard-for-safari/id1440147259?mt=12

extension. This addition significantly increases evasion from tracking and analytics networks (see Figure 14 in the Appendix). Coverage of blocking tracking is not as efficient as Ublock Origin, and some tweaking of the rules AdGuard uses may be necessary for users who desire extra privacy protection from standard tracking methods.

Due to the platform limitations, there are no web extension options for fingerprinting evasion on Safari on macOS. Even with this limitation, as macOS is a full desktop platform, there are still other options. Users can use a different browser that supports the recommended extensions Ublock Origin and Fingerprint Spoofing or CanvasBlocker, such as Chrome, Edge, or Firefox. Alternatively, for users that prefer to use Safari, they can wait for the next macOS update (with a planned release of late September 2023), which includes the same major privacy updates for fingerprinting and tracking prevention as iOS 17. As tested on iOS, the new anti-fingerprinting measures in Safari and WebKit provide the strongest out-of-the-box protections, defeating even CreepJS's advanced fingerprinting methods (see Sections 3.2.2 and 4.2 for more details).

3.1.3. Fedora 38

Linux testing was conducted on Fedora 38 with the built-in browser Firefox on an emulated ARM laptop. Fedora was chosen as the Linux representative due to its shared lineage with Red Hat Enterprise Linux, as well as being a common choice for workstation Linux installs. In addition, of all browsers tested in the process of this research, Firefox is the only browser available on all platforms, excluding the WebKit skin on iOS, and claims to have built-in privacy protections right from the start.

Many privacy-enhancing measures are built into Firefox by default as a part of the enhanced tracking protection and total cookie protection suite of features. With enhanced tracking protection, Firefox claims protection against analytics and usage trackers, social media trackers, cross-site tracking cookies, crypto miners, and fingerprinting. As described in Steven Englehardt's (2020) Mozilla blog post, fingerprinting protection has been available in Firefox since early 2020. However, the key detail to note here is that Firefox attempts to block known fingerprinting domains rather than attempting to evade fingerprinting methods with spoofing or blocking elements on the webpage directly.

For standard analytics and tracking blocking for ad networks, Firefox has the best protections of all browsers tested out of the box. Although coverage is not perfect, it still provides a much greater privacy advantage by default out of the box compared to other browsers (see Figure 15 in the Appendix).

The testing methods used for Firefox fingerprinting protection capture data directly from the fingerprinting frameworks rather than from website browsing. As Firefox's built-in blocking operates on restricting access from known fingerprinting domains, evaluation of the protection will be done based on assuming that websites are still able to fingerprint the browser and user independent of using third-party scripts. In addition, if the fingerprinting scripts are loaded from the same domain that the user is currently on, they will likely not be blocked by Firefox as it uses domain name matching rather than element-based blocking.

With these assumptions in place, Firefox seems to have limited fingerprinting protection out of the box, as domain-based protection cannot prevent websites from capturing identifying data. In the Cover Your Tracks testing, though Firefox can block analytics and tracking well, it still presents many identifying bits of fingerprinting information, including Canvas elements, WebGL, and audio contexts (see Figure 16 in the Appendix).

The lack of fingerprinting protections is further established with FingerprintJS and CreepJS. For both of these frameworks, Firefox presents all data as requested without deviations, so the browser and user are identifiable across different browsing sessions and networks. As the hardware remains consistent across sessions, so do the outputs of Canvas and WebGL rendered elements, so both of these frameworks are able to consistently generate the same identifying hash of the user (see Figures 17 and 18 in the Appendix).

Tests on Firefox were performed again after installing the addons Ublock Origin and CanvasBlocker. For ad blocking, the addition of Ublock Origin increases blocking coverage significantly, blocking all trackers and analytics used on AdBlock Tester and Test Ad Block (see Figure 19 in the Appendix). While Firefox's built in privacy

protections are excellent out of the box, they are still supplemented by the addition of an ad-blocking extension.

With the addition of CanvasBlocker, Firefox now has protection against fingerprinting methods, and spoofs Canvas, WebGL, and audio contexts. This is enough to defeat FingerprintJS, as even on a page refresh, these elements are randomized every time. For Cover Your Tracks, the browser still appears as "unique," though this may be a product of the randomized hashes (see Figure 20-22 in the Appendix). CreepJS is still able to detect the browser, however, due to its "lie" detection metrics (see Section 4.2 for more details).

3.1.4. Chrome OS

Chrome OS testing was performed on a Lenovo Chromebook 14 with an officially supported Chrome OS build.

Like other browsers with Chromium and Blink web engine lineage, Chrome in Chrome OS does not necessarily have a focus on web privacy and provides limited built-in OS or browser controls to limit tracking. Although it is possible to install alternative browsers with better privacy protections through the Play Store on Chrome OS, these Android apps may not provide a proper desktop browsing experience. When considering how the "average" user is likely to use Chrome OS, most users will likely stick with Chrome on a Chromebook.

Using Chrome out of the box provides moderate protection from traditional analytics or tracking-based ad platforms. Although some tracking and analytics scripts are blocked, a large contingent of tracking networks are still allowed to load (see Figure 23 in the Appendix). In comparison to Edge, another Chromium-based browser, privacy protections are better by default, but still not sufficient.

With respect to fingerprinting, Chrome and Chrome OS also have no built-in protections to evade this method of tracking. This operating system is focused on the web, built around Chrome, so naturally, most users using Chrome OS would likely not want a compromised web browser experience. In some respects, running Chrome OS has the potential to "blend in" with other users, as support for Chrome OS is only provided

on Google-approved hardware, and customers are likely to buy large quantities to use in fleets for schools or businesses (Quinn, 2016). With large quantities of similar browsing hardware, it may be possible for one device to look similar to many; however, the caveat of the testing process of this research is the limited set of available test hardware. As a result, this testing assumes Chrome OS is being used by a singular user, looking to improve their fingerprinting privacy protections.

The results from Cover Your Tracks, FingerprintJS, and CreepJS all demonstrate that Chrome has no fingerprinting protection, and can identify any user. Given the static nature of the hardware in Chromebooks, rendering Canvas, WebGL, and audio context elements will always report the same data back to the website. As the core functions behind fingerprinting, this will allow fingerprinting frameworks to generate a unique identity on Chrome users (see Figures 24-26 in the Appendix).

Achieving additional privacy protections for web browsing on Chrome is simple, as installing the two recommended extensions, Ublock Origin and Fingerprint Spoofing, are significant measures against analytics and fingerprinting. For analytics and tracking-based ad networks, installing Ublock Origin increases ad blocking coverage significantly, with nearly full coverage against the tests on AdBlock Tester, and 100% coverage on Test Ad Block (see Figure 27 in the Appendix).

With fingerprinting, results are similar as seen in Edge in Windows. In Cover Your Tracks, the identifying elements used in the test suite are confirmed to be randomized, yet still appears to be a "unique fingerprint." As the Canvas, WebGL, and audio context elements are randomly generated by the Fingerprint Spoofing extension, the browser will present itself as "unique" device to the tracking platform (see Figure 28 in the Appendix). In reality, the randomization of web elements is not likely to create the same result every time, and the client will look like a new user on each browsing session.

FingerprintJS further confirms this hypothesis; while a unique identifier is generated due to the randomization of reported fonts or Canvas generation, it changes even when refreshing the webpage. On each refresh, Fingerprint Spoofing presents a new set of system-reported fonts, Canvas and WebGL rendering, and fake audio contexts, which confuses the metrics used by FingerprintJS to identify the user (see Figure 29 in

the Appendix). CreepJS is not fooled, however, and uses its advanced metrics and "lie" detection to detect the spoofed elements from Fingerprint Spoofing, and still identifies the user and browser (see Figure 30 in the Appendix and Section 4.2 for a technical breakdown of CreepJS).

3.2. Mobile Platforms

The mobile test platforms consist of Android Version 13, running on a Pixel 6 Pro, and iOS 17, running on an iPhone XR. The data collection tests were run on each device with limited account logins to ensure potential tracking or fingerprinting data did not bleed over from the researcher's private accounts.

3.2.1. Android

The main limitation of attempting to use privacy-enhancing measures on Android is that the default browser on most phones is Google Chrome. Unlike its desktop counterpart, mobile Chrome does not allow the installation of addons or extensions from the Chrome Web Store, so users are stuck with limited privacy options if using only Chrome.

On AdBlock Tester, Chrome scores 39 out of 100; on d3ward's Test Ad Block, out of 150 total advertising elements, only seven are blocked (see Figure 31 in the Appendix). The critical failure in Chrome's privacy measures is not blocking any analytics or web advertising scripts, which will allow websites to track users' data.

Chrome does not contain any built-in fingerprint evasion techniques or blocking measures. As a result, across every fingerprint test platform that issues an identifying hash value, Chrome is uniquely identifiable and can be fingerprinted across sessions and networks. Without being able to spoof or disable Canvas features or mangle the available fonts reported to a website, every instance of mobile Chrome will likely look unique for any website or company employing web fingerprinting. There may still be some level of ambiguity or inherent "blending in" across users with the same Android devices (e.g., Pixel, Galaxy devices), as the similarity in hardware will present the same web element rendering to fingerprinting frameworks. However, this should not be necessarily relied upon as a method of remaining private, as a fingerprint in combination with browsing

habits and analytics data collected due to Chrome's lack of privacy measures is likely more than enough for companies to continue to build unique profiles on users.

In incognito mode, Chrome blocks third-party cookies by default, which may prevent analytics or data collection from parties external to the current website, and the lack of third-party tracking makes the user appear slightly less unique (see figure 32 in the appendix). FingerprintJS and CreepJS are still able to uniquely identify the user regardless of browsing mode as they do not rely on cookie or browser storage-based tracking. The metrics used to create the user's identifying hash, such as available fonts, Canvas, and WebGL rendering functions, remain the same regardless of browsing mode (see Figures 33 and 34 in the Appendix).

Chrome does not provide sufficient privacy controls or extension support to block or prevent the rendering elements that create a unique identifying hash, including Canvas, WebGL, or fonts. Unlike iOS, alternative browsers are a viable option as a privacy measure, as these browsers are allowed to utilize customizations on their web rendering engines.

As the focus is on providing the most straightforward possible privacy measures for the "average" user to enact, the proposed privacy solution is the Brave browser, which provides many built-in privacy features, including adblocking and fingerprinting protection. While Firefox is available on Android with extension support, a minimal subset of desktop extensions is available. Based on the desktop Firefox browser testing results, Firefox's built-in fingerprinting protections need to be improved to block even FingerprintJS, which uses a more straightforward set of metrics to create an identifying hash than CreepJS. Although there are configuration hardening scripts and methods to allow the installation of desktop Firefox extensions, these are likely beyond the technical ability of the average user.

Brave provides many built-in and user-accessible options to control privacy measures and can easily be set by any user after installation. Brave was configured based on the controls specified in Privacy Guides' mobile browser recommendation page; namely, the critical controls suggested by Privacy Guides to improve privacy and anti-

fingerprinting are setting ad blocking to aggressive, fingerprint blocking to strict, and disabling non-proxied UDP for WebRTC settings.

With these basic settings changed, Brave provides a significant increase in privacy compared to the default Chrome install on Android and can block nearly all basic ad-based analytics and tracking, with coverage as effective as Ublock Origin on desktop browsers (see Figure 35 in the Appendix).

While following the Privacy Guides recommendations on hardening the browser configuration, a secondary goal for fingerprinting protection was to replicate the functions of Fingerprint Spoofing and CanvasBlocker as best as possible on mobile. Both of these extensions randomize the outputs of the objects used for fingerprinting metrics, and Brave takes a similar approach in strict mode. Strict mode blocking in Brave randomizes Canvas, WebGL, audio contexts, and other hardware enumeration, while also blocking WebRTC from leaking the user's IP ("Fingerprinting Protections", 2021).

These methods are very effective against Cover Your Tracks and FingerprintJS's identification methods, which are largely based on collecting Canvas, WebGL, audio context, and device hardware data (see Figures 36 and 37 in the Appendix). Even refreshing the page on FingerprintJS creates a new identifier hash, meaning the device and browser looks like a new client each time. An interesting point of data is that Cover Your Tracks is able to tell that randomization of fingerprinting elements is performed, and reports it as such, while desktop browsers still appear as unique to this tool. The technical reason for this is unknown, as the Cover Your Tracks data for both desktop browsers and Brave mobile report that the elements have been randomized.

The CreepJS test can still uniquely identify the user, with the caveat that the CreepJS framework is specifically designed to defeat strict fingerprinting evasion methods such as those Brave uses (see Figure 38 in the Appendix). For more details on CreepJS's detection methods, see Section 4.2.

3.2.2. iOS

For devices running iOS or iPadOS, all third-party browsers distributed through the App Store must use the WebKit engine built into the operating system and cannot

ship their own browser engine[10]. All non-Safari browsers, such as Chrome (Blink) or Firefox (Gecko), are effectively custom user interfaces for web browsing on top of the WebKit engine and will likely report the same details to browsers for analytics, tracking, and fingerprinting. As a result, research and recommendations for iOS or iPadOS will only include results from Safari. Testing was conducted on one iOS device due to limitations in available hardware, and the only difference between Safari in iOS and iPadOS was a mobile versus a desktop user-agent string.

In iOS and iPadOS 17 (current versions at the time of conducting research), Apple introduced new fingerprinting and tracking evasion features, turned on by default for private browsing mode, with the ability to enable it for all web browsing (Apple, 2023). Like the privacy measures first introduced into Safari and WebKit in 2019, the exact method of fingerprinting protection is not disclosed. For the purposes of testing this new feature, the fingerprinting protections were configured to operate in all browsing modes, where the default is only to operate in private browsing mode.

Out of the box, Safari has reasonable privacy protections, preventing standard JavaScript analytics scripts from running through AdBlock Tester. Still, it allows most ad network scripts to be loaded in Test Ad Block (see Figure 39 in the Appendix).

Fingerprinting prevention also appears to be improved compared to the previous version of Safari and WebKit, as tested in macOS Ventura. With the advanced tracking and fingerprinting protection option turned on for all browsing, Safari now spoofs various canvas and WebGL elements like the Fingerprint Spoofing or CanvasBlocker extensions for desktop browsers.

Cover Your Tracks reports that the browser does not appear unique, meaning it can effectively "blend in" and look similar to other devices. The main driver behind this change in fingerprinting between browser versions is due to spoofing both Canvas and WebGL elements. In previous Safari versions, while the built-in protections still

[10] https://developer.apple.com/app-store/review/guidelines/#software-requirements (Section 2.5.6)

randomized the canvas, WebGL and audio fingerprinting were reported as-is (see Figure 40 in the Appendix).

In iOS 17, Cover Your Tracks now reports that the WebGL and audio context fingerprints both appear to be randomized (see Figure 41). With fewer metrics available from the browser for the tool to collect, the browser and device now look less identifiable and blend in among web users from the perspective of the EFF's test tool. However, with the data collected from the other two fingerprinting frameworks, there may be a flaw in the new privacy implementations that still allows Safari to be seen as unique.

With both FingerprintJS and CreepJS, the iPhone test device still appears as a unique user and device to both of these tracking frameworks. With FingerprintJS in particular, the new anti-fingerprinting in Safari prevents Canvas element rendering and audio context generation entirely. As a result, the device still appears unique, producing a unique identifying hash that persists across browsing sessions and IPs (see Figure 42 in the Appendix).

Of particular interest are the audio context and Canvas tests used in FingerprintJS. While Cover Your Tracks reports that both of these elements are randomized and thus not unique, FingerprintJS cannot capture audio context or Canvas rendering. As a comparison, the previous version of Safari on macOS can provide data for both (see Figure 43 in the Appendix).

On iOS 17's updated Safari, the audio and Canvas tests do not run, and as a result, this appears unique from the perspective of FingerprintJS (see Figure 44 in the Appendix). Without the implementation details from Apple, it is not apparent why these methods do not render correctly with FingerprintJS. However, it has the inadvertent side effect of allowing the browser and user to remain unique and trackable.

With CreepJS, it is apparent that the new fingerprint protections are indeed functioning, and even with CreepJS's advanced "lie" detection and other metrics such as JavaScript engine detection, Safari is still able to defeat CreepJS while no other tested browser can (see Figure 45 in the Appendix). With Canvas, WebGL, and audio context

randomizations in place, Safari appears randomized and non-unique on every session. See Section 4.2 for more details on CreepJS's detection mechanisms.

As Safari's new fingerprinting protections are sufficient, the only necessary additional testing is extensions to prevent analytics and traditional tracking methods. However, a major roadblock on iOS is that nearly all Safari ad blocking extensions are non-free and require paid subscriptions. Even with the iOS alternative browser platform limitations, browsers are still able to provide custom features.

Like with Android testing, Brave was chosen as the alternative for analytics and tracking prevention, as it has a built-in ad blocking mechanism. With ad blocking set to "aggressive," Brave provides excellent coverage against advertising tracking and analytics networks, providing near full coverage on both ad block testing platforms (see Figure 46 in the Appendix). Unfortunately, with fingerprinting protection, the new Safari and WebKit features are not carried over due to the platform limitations and web engine features which are not allowed to be used by third-party browser skins. Brave on iOS has a limited fingerprinting protection option, and no granular controls like its Android counterpart. Fingerprinting protection behaves similarly to the previous version of Safari on macOS, with limited randomization of elements used for identification. For most users, given the vastly improved fingerprinting protections in Safari, along with the built-in basic tracker protection, it is recommended to stick with Safari for Apple mobile devices.

4. Recommendations and Implications

Taking privacy-enhancing measures is enough to disrupt basic tracking and fingerprinting methods, and for most users, installing essential browser extensions may be enough to achieve better privacy.

4.1. Recommendations for Practice

For most users, it may become increasingly challenging to avoid deep fingerprinting, with capabilities becoming advanced, and every bit of information the browser provides allows a unique identifier on the user to be generated. Based on the

findings, extreme measures such as disabling advanced web features or disabling JavaScript may cause a user to stand out and be even more identifiable by tracking or fingerprinting measures. As a result, the conclusion is that for most people, unless there is a case where extreme anti-tracking measures are required, taking the approach of "blending in" is more achievable than attempting to block every fingerprinting or tracking measure.

The recommendation to block advertising and analytics networks is to install an ad-blocking add-on. For most browsers, including Chrome, Firefox, and Microsoft Edge, Ublock Origin has the best coverage in blocking ads and tracking elements across the internet without compromising the web experience for users (Mazel et al., 2019). On AdBlock Tester, just adding Ublock Origin blocks analytics tools or contextual advertisements from displaying or generating tracking data on the user. For users on macOS and iOS, it can be recommended to install AdGuard for Safari if alternative browsers are not an option. Notably, on iOS, all browsers are only different user interfaces on top of the WebKit engine built into the operating system, so it is recommended to use Safari, especially with the improved privacy protections introduced in iOS 17.

Based on the results, defeating fingerprinting methods is a bit more involved, but for the less technically inclined, can still be achieved by installing additional browser addons, as well as tweaking the rules used in the adblocking addon to block additional web elements. The primary recommendation for anti-fingerprinting is installing a Canvas blocking or scrambling addon, either Fingerprint Spoofing on Chrome and Edge, and CanvasBlocker on Firefox.

Although it is possible to vastly reduce the efficacy of tracking by entirely disabling JavaScript, changing available system fonts, or using web browsing options such as Tor, these are beyond the scope of this paper. These are beyond what one would consider easy for the average user to do to obtain privacy on the internet, and such measures are only likely in extreme edge cases.

4.2. CreepJS Analysis

Throughout all test platforms, except iOS 17, the one constant was in CreepJS's consistent ability to detect and uniquely identify every browser despite installing extensions that attempt to bypass standard fingerprinting metrics such as Canvas or WebGL.

While CreepJS is intentionally designed to look for web extensions and try to determine if the browser is presenting spoofed feature sets, there are constants in every browser and engine that do not change. CreepJS is also able to determine using "fuzzy" logic what is likely to be the actual value that the browser is attempting to hide. Along with these constant values and computer hardware information that is difficult to spoof, it can posit that the visiting browser is the same one despite element randomization. The full details of how this detection metric is computed is not documented within the source code for CreepJS, but the results collected from testing demonstrate how browsers reveal their true identity.

JavaScript implementations are based on the language specification known as ECMAScript[11]. Each web browsing engine has its implementation of the language, as the interpreted code must be able to run as performantly as possible on the native hardware. Mowery et al. (2011) find that the differing nature of JavaScript engine implementations combined with differing underlying hardware presents a method to identify a user or computer uniquely. Through the process of research, there were no method or browser extensions found that could spoof the JavaScript engine output. At the same time, Canvas elements can operate with spoofed reporting from the browser, modifying the JavaScript engine is likely to break functions entirely. As a result, every browser in the testing process effectively reveals itself to CreepJS, and along with the fuzzy logic in use, the framework can assume that the client is "lying" about the spoofed elements and connect it back to the original identifier.

[11] https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

The screenshots below demonstrate CreepJS's ability to detect that the browser is "lying" about its WebGL and Canvas features (see figure 47 in the appendix). It can also detect changes between different sessions of running CreepJS. For these sub-identifications, CreepJS issues a hash and can tie these back to the original browsing session.

Even with multiple browser components spoofed, the device still reveals its true identity, as the mathematics elements indicate running on Chromium's V8 JavaScript engine (Edge and Chrome), SpiderMonkey (Firefox/Gecko), or JavaScriptCore (Safari/WebKit) (see figure 48 in the appendix). In addition, each browser engine formats CSS based on its own unique implementation and reveals the underlying browser engine to CreepJS, with the exception of CSS on Firefox. CreepJS was still able to identify the browser with the same hash that was captured prior to installing extensions on all browsers, with the exception of Safari on iOS.

The only browser and operating system in the test suite able to defeat CreepJS's advanced metrics and lie detection is Safari on iOS 17. As described in the results, Safari and WebKit's new fingerprinting protections can spoof Canvas, WebGL, and audio context elements, much like the functions in the extensions Fingerprint Spoofing or CanvasBlocker. Unlike these extensions, CreepJS cannot detect the "lies" in the spoofed elements presented by Safari. Though Apple does not reveal the exact implementation details, the method used to randomize the outputs of these web elements is likely to be more careful and appear as if the parameters are still from a legitimate browser rather than completely randomized with browser extensions. Alternatively, as mentioned, CreepJS is explicitly designed to defeat browser fingerprinting protections, and the project author may be able to add further detection mechanisms or lie detection logic to precisely detect the spoofing performed in iOS 17's version of Safari and WebKit.

As seen in Figures 49 and 50 below, even with WebGL outputs appearing to be the same between separate sessions of CreepJS, the Canvas and audio context elements are randomized but do not appear to be the same user or client to CreepJS, as it issues a different fingerprint hash each session.

Figure 1

CreepJS iOS 17 Audio Context Randomization

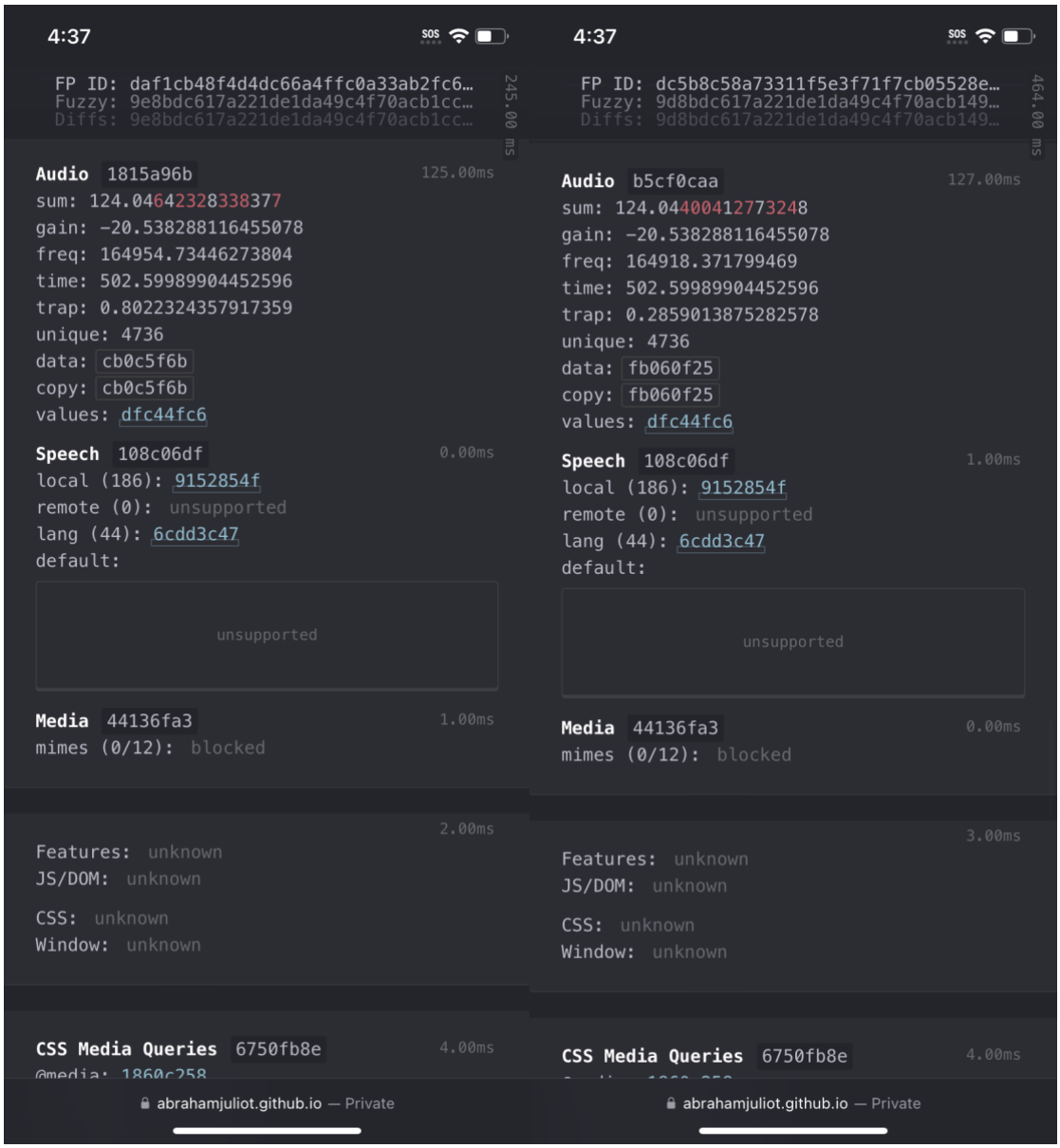


Figure 2

CreepJS iOS 17 Canvas Randomization

CreepJS does not track the user across different browsers on different systems, as it is a project demonstrating web fingerprinting capabilities. In theory, if a company utilizes these capabilities to track usage data and web browsing activity, they can likely track users across the entire web and world.

4.3. Implications for Future Research

Future research will likely be served best by determining if anti-tracking and anti-fingerprinting measures are effective across multiple websites. The main limitation of the research presented here is time constraints and available resources. For instance, the platform FingerprintJS reports many clients using their tools for fingerprinting; however, with significant time and resources, it is easier to determine precisely how these major websites use FingerprintJS. Given the widespread use of fingerprinting across major websites today (Iqbal, 2021), future research may also need to consider whether this data is used to track users across different websites.

Further testing on fingerprinting may need to be conducted with web browsing without JavaScript. As the goal of this research is to enhance privacy without compromising website functionality, disabling JavaScript is outside the scope of this paper. As the functionality of all three fingerprinting sites is based on JavaScript, future in-depth research could include testing fingerprinting without JavaScript active. In addition to its commercial offerings, FingerprintJS has a proof-of-concept demo to demonstrate fingerprinting without scripts or most web features enabled[12]. The demo uses the differences in how browsers treat CSS formatting and various HTTP headers to fingerprint the user without using JavaScript.

5. Conclusion

Even with the pervasiveness of advanced web fingerprinting methods and rampant ad tracking networks today, it is possible for the “average” user to take back their privacy by taking some simple steps. Installing privacy-enhancing extensions on desktop browsers or using proper tools on mobile platforms may be enough to disrupt the fingerprinting techniques used by major companies today. Looking to the future, with machine learning and artificial intelligence rapidly developing, future research into

[12] https://noscriptfingerprint.com/

privacy protections may need to become more aggressive to protect users from being tracked across the web.

References

Apple. (2023, June 5). *Apple announces powerful new privacy and security features* [Press release]. https://www.apple.com/newsroom/2023/06/apple-announces-powerful-new-privacy-and-security-features/

Apple. (2019, November). *Safari Privacy Overview*. Apple. https://www.apple.com/safari/docs/Safari_White_Paper_Nov_2019.pdf

Brave. (2021, September 30). *Fingerprinting Protections*. Brave Browser Wiki. https://github.com/brave/brave-browser/wiki/Fingerprinting-Protections

Englehardt, S. (2020, January 7). *Firefox 72 blocks third-party fingerprinting resources*. Mozilla Security Blog. https://blog.mozilla.org/security/2020/01/07/firefox-72-fingerprinting/

Firefox. (n.d.). *Trackers and scripts Firefox blocks in Enhanced Tracking Protection*. Firefox. https://support.mozilla.org/en-US/kb/trackers-and-scripts-firefox-blocks-enhanced-track

Firefox. (n.d.). *Introducing Total Cookie Protection in Standard Mode*. https://support.mozilla.org/en-US/kb/introducing-total-cookie-protection-standard-mode

Hu, H., Peng, P., & Wang, G. (2019, May). Characterizing pixel tracking through the lens of disposable email services. In *2019 IEEE Symposium on Security and Privacy (SP)* (pp. 365-379). IEEE.

Iqbal, U., Englehardt, S., & Shafiq, Z. (2021, May). Fingerprinting the fingerprinters: Learning to detect browser fingerprinting behaviors. In *2021 IEEE Symposium on Security and Privacy (SP)* (pp. 1143-1161). IEEE.

Mazel, J., Garnier, R., & Fukuda, K. (2019). A comparison of web privacy protection techniques. Computer Communications, 144, 162-174.

Mowery, K., Bogenreif, D., Yilek, S., & Shacham, H. (2011). Fingerprinting information in JavaScript implementations. *Proceedings of W2SP*, *2*(11).

Mowery, K., & Shacham, H. (2012). Pixel perfect: Fingerprinting canvas in HTML5. *Proceedings of W2SP*, *2012*.

Nikiforakis, N., Kapravelos, A., Joosen, W., Kruegel, C., Piessens, F., & Vigna, G. (2013, May). Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *2013 IEEE Symposium on Security and Privacy* (pp. 541-555). IEEE.

Privacy Guides. (2023, July 24). *Privacy Respecting Mobile Web Browsers for Android and iOS - Privacy Guides*. Privacy Guides. https://www.privacyguides.org/en/mobile-browsers/

Quinn, P. (2016). Google schools?: A Chromebook case study. Screen Education, (82), 90-94.

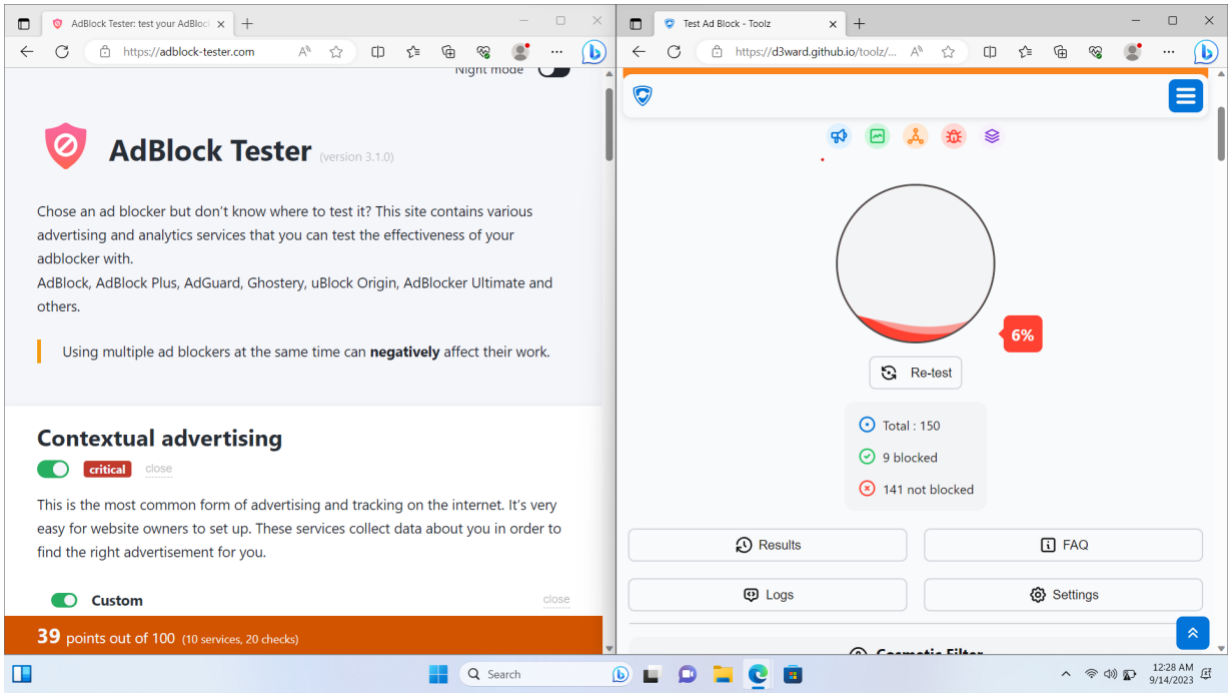
Upathilake, R., Li, Y., & Matrawy, A. (2015, July). A classification of web browser fingerprinting techniques. In *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)* (pp. 1-5). IEEE.

WebKit. (n.d.). *Tracking Prevention in WebKit*. Apple. https://webkit.org/tracking-prevention

Appendix
Screenshot Results

Figure 3

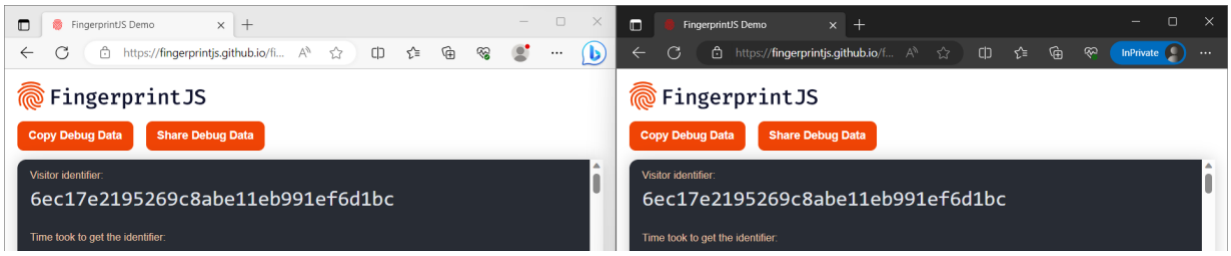
Windows 11 Adblock Test Out of the Box



Note. Edge provides very limited protection against ad network analytics and tracking.

Figure 4

Windows 11 FingerprintJS Test Out of the Box

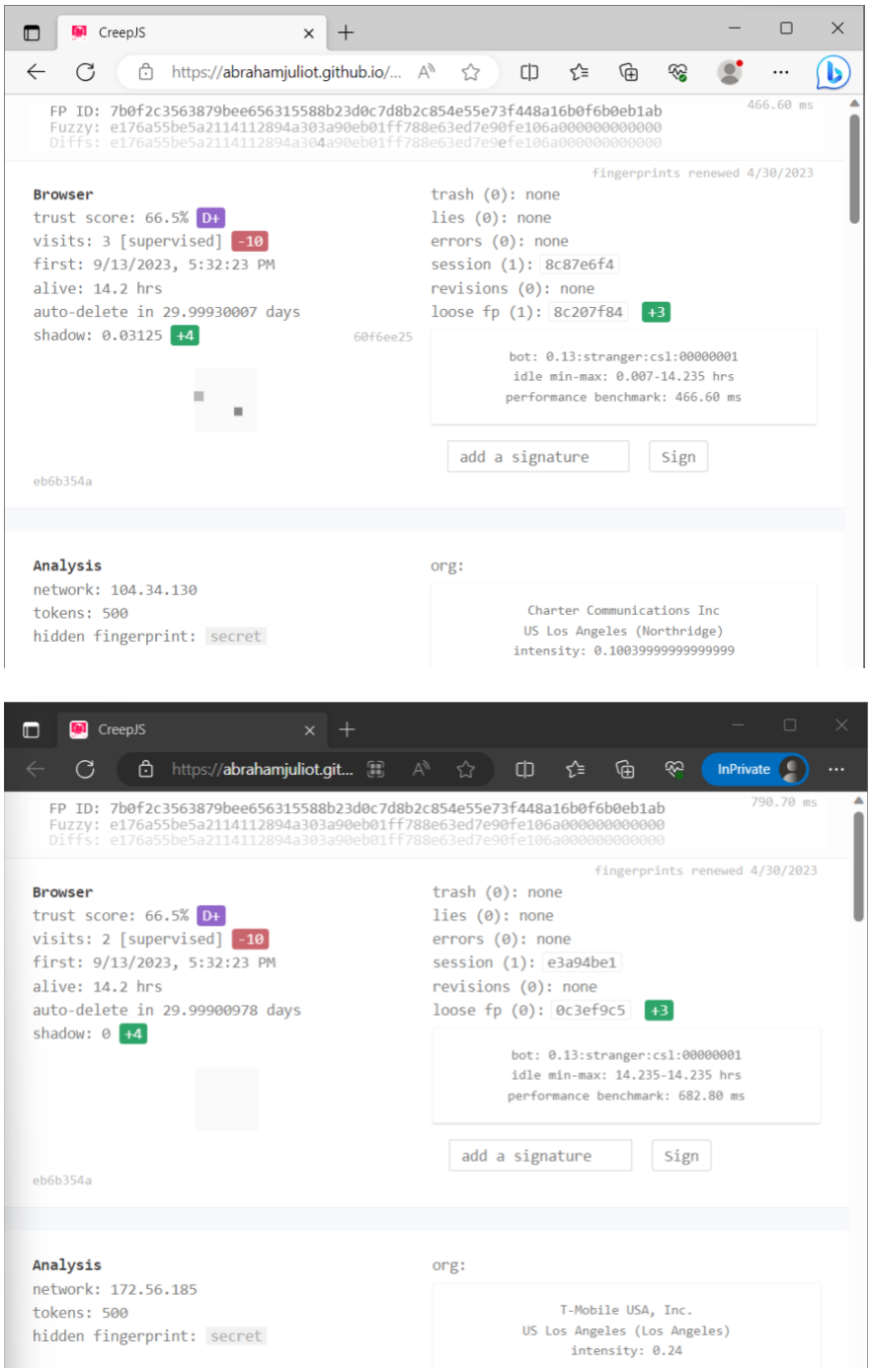


Note. Edge provides no fingerprint protection out of the box.

Issac Kim, issac.kim@student.sans.edu

Figure 5

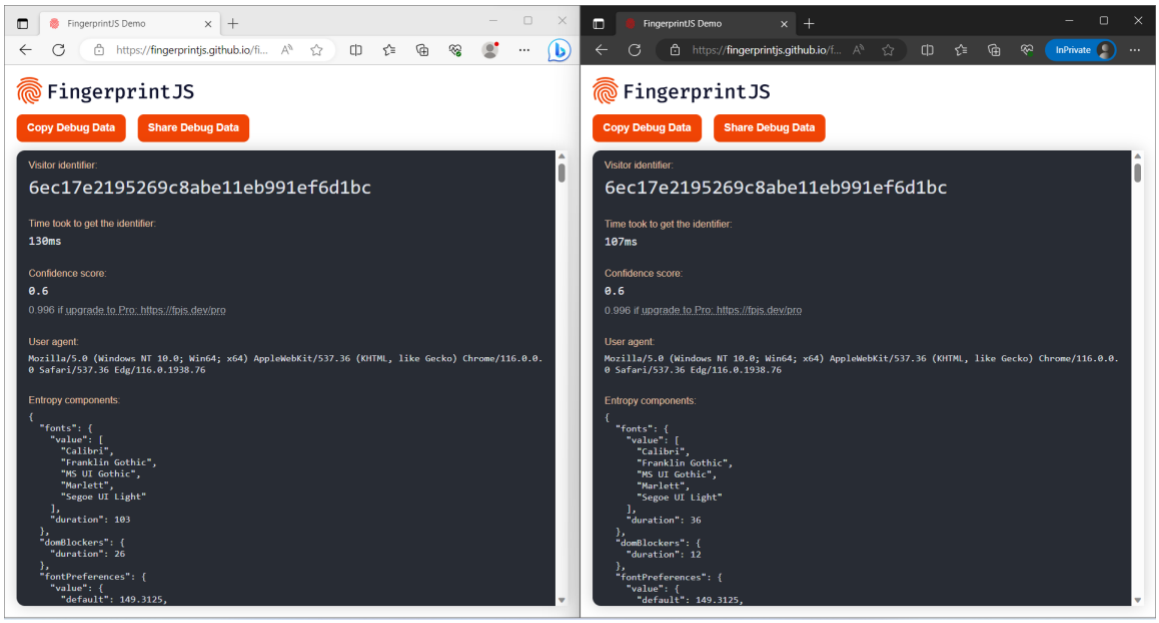
Windows 11 CreepJS Test Out of the Box



Note. Edge provides no fingerprint protection out of the box.

Figure 6

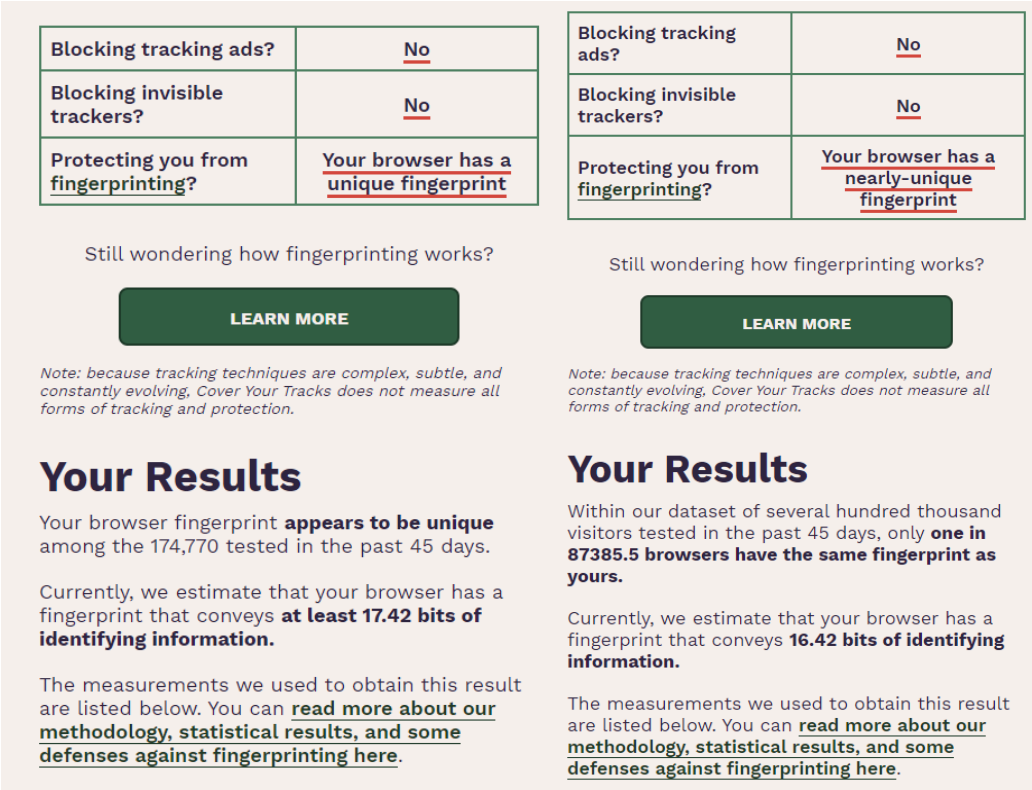
Windows 11 CreepJS Test Out of the Box



Note. Edge provides no fingerprint protection out of the box.

Figure 7

Windows 11 Cover Your Tracks Test Out of the Box



Note. Edge provides no fingerprint protection out of the box (left), but has limited third-party cookie protection in InPrivate mode (right).

Figure 8

Windows 11 Adblock Tests with Extensions

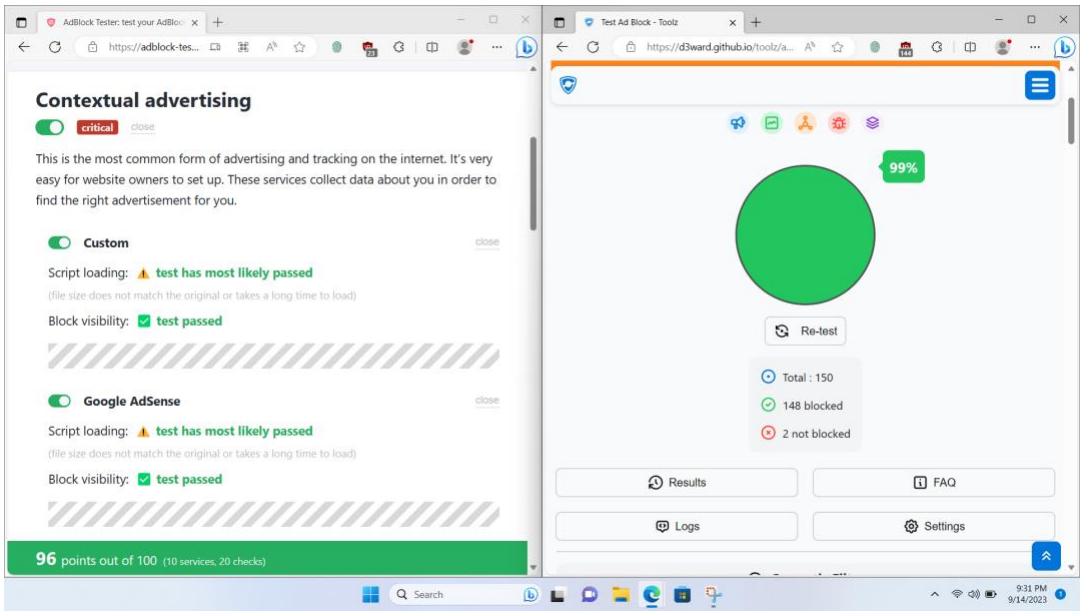


Figure 9

Windows 11 Cover Your Tracks with Extensions

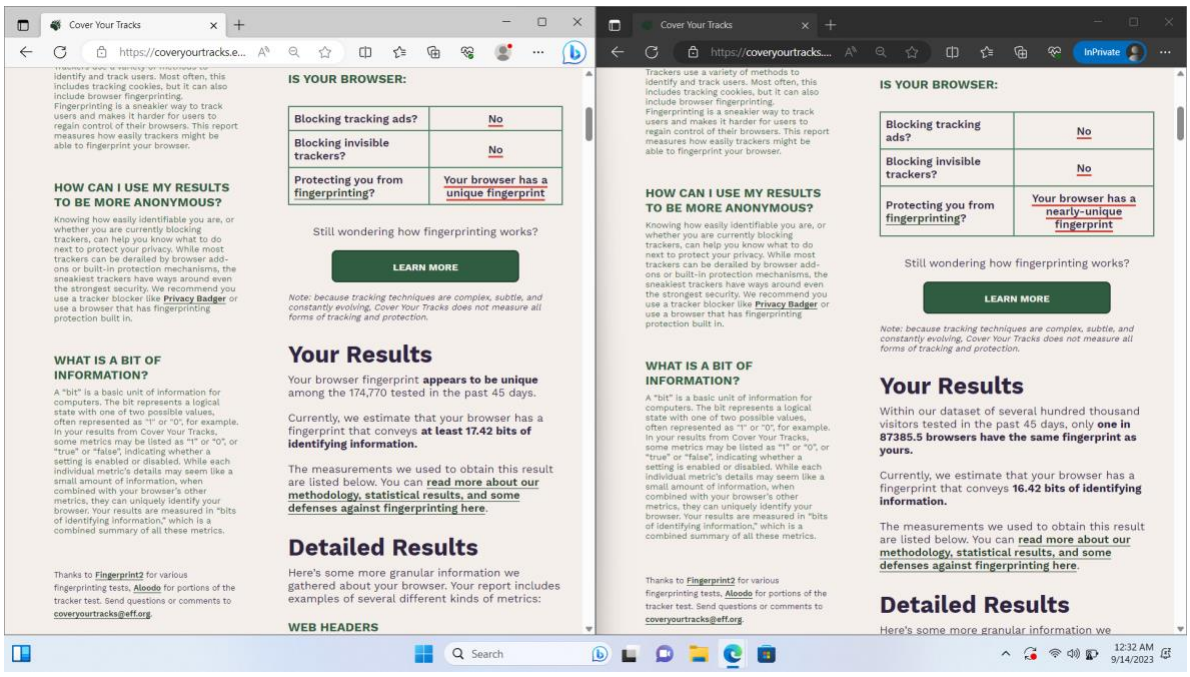


Figure 10

Windows 11 FingerprintJS with Extensions

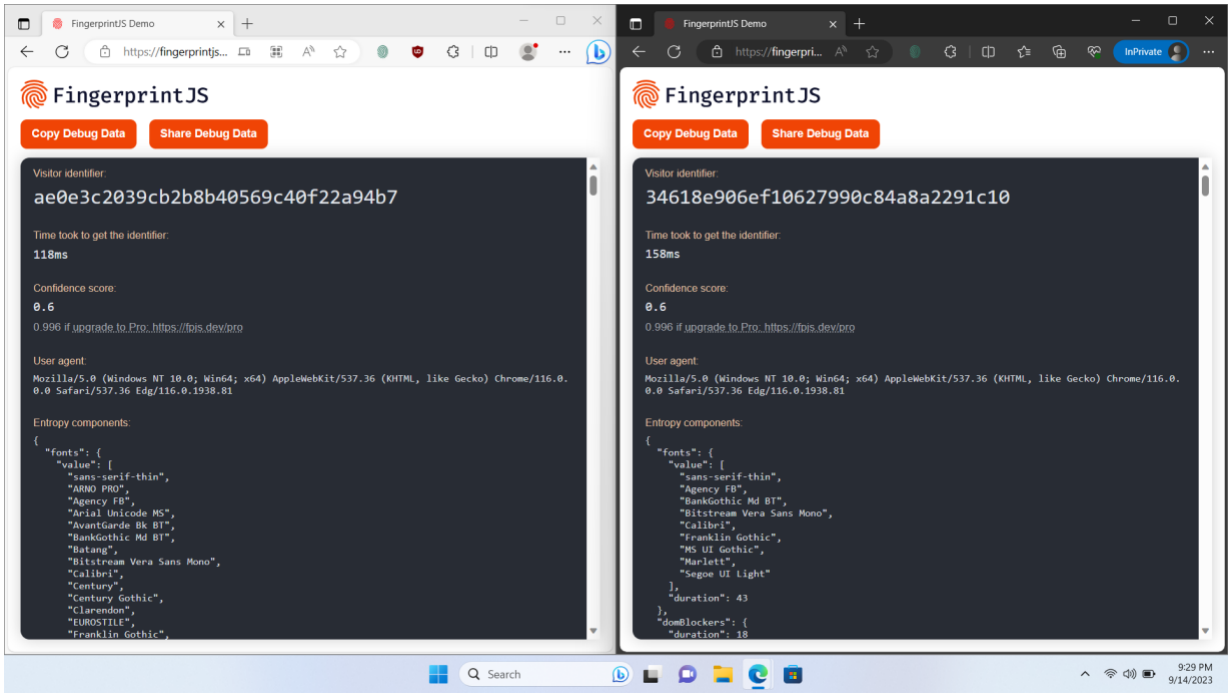


Figure 11

Windows 11 FingerprintJS with Extensions

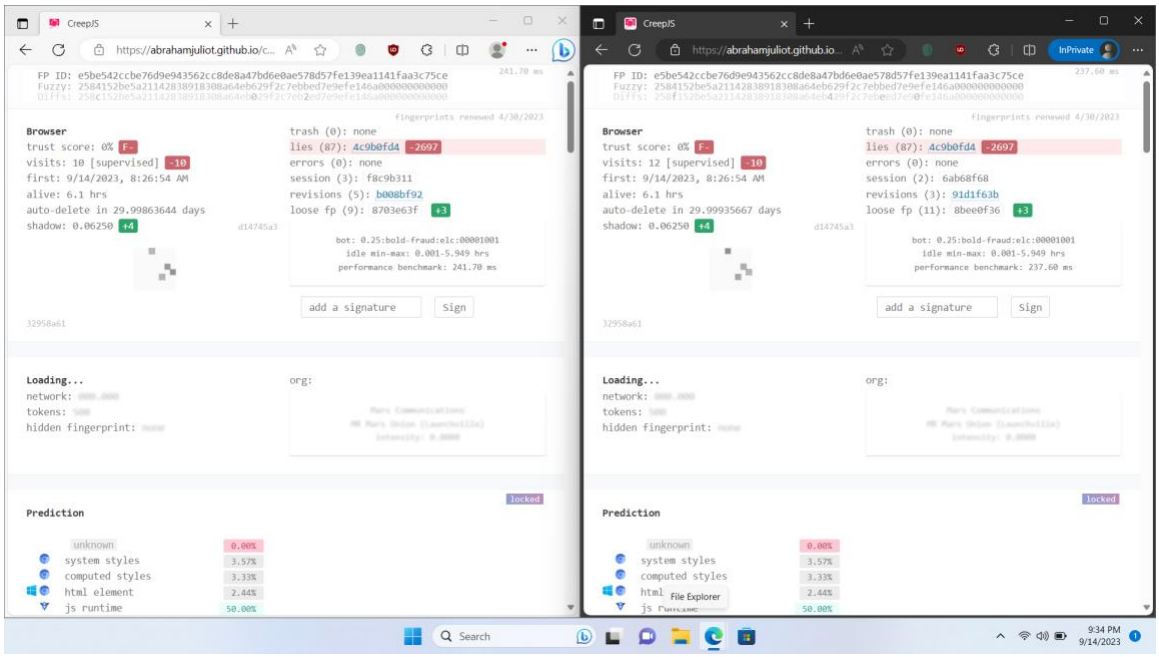
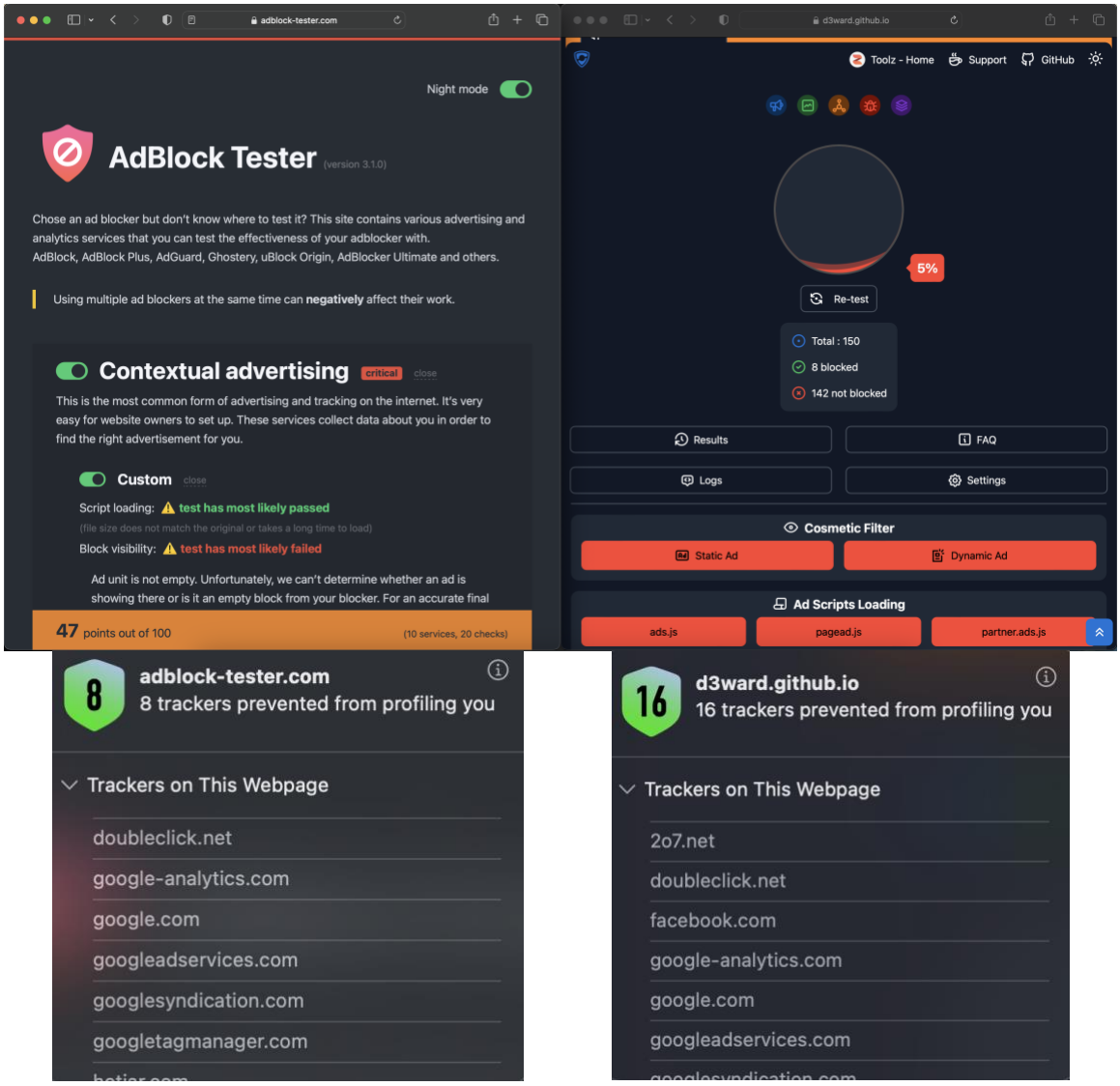


Figure 12

macOS AdBlock Tests Out of the Box



Note. Safari blocks some tracking by default but does not have good coverage compared to adblocking extensions.

Figure 13

macOS Cover Your Tracks Out of the Box

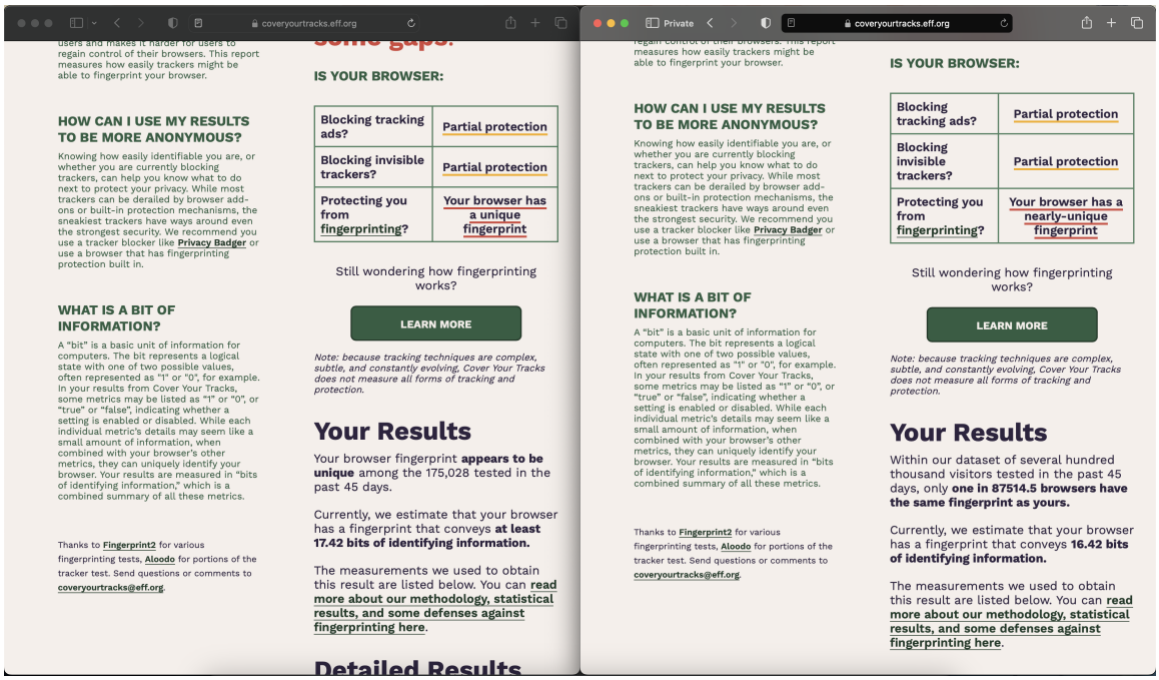
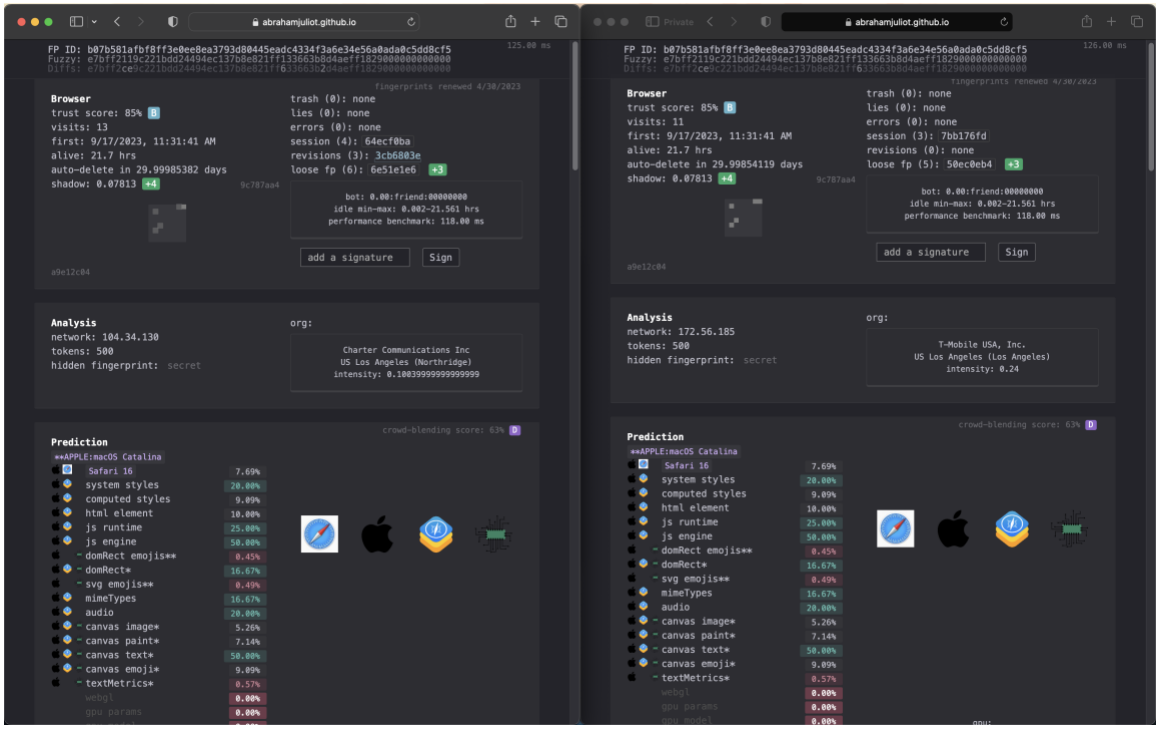


Figure 14

macOS CreepJS Out of the Box



Issac Kim, issac.kim@student.sans.edu

Figure 15

macOS FingerprintJS Out of the Box

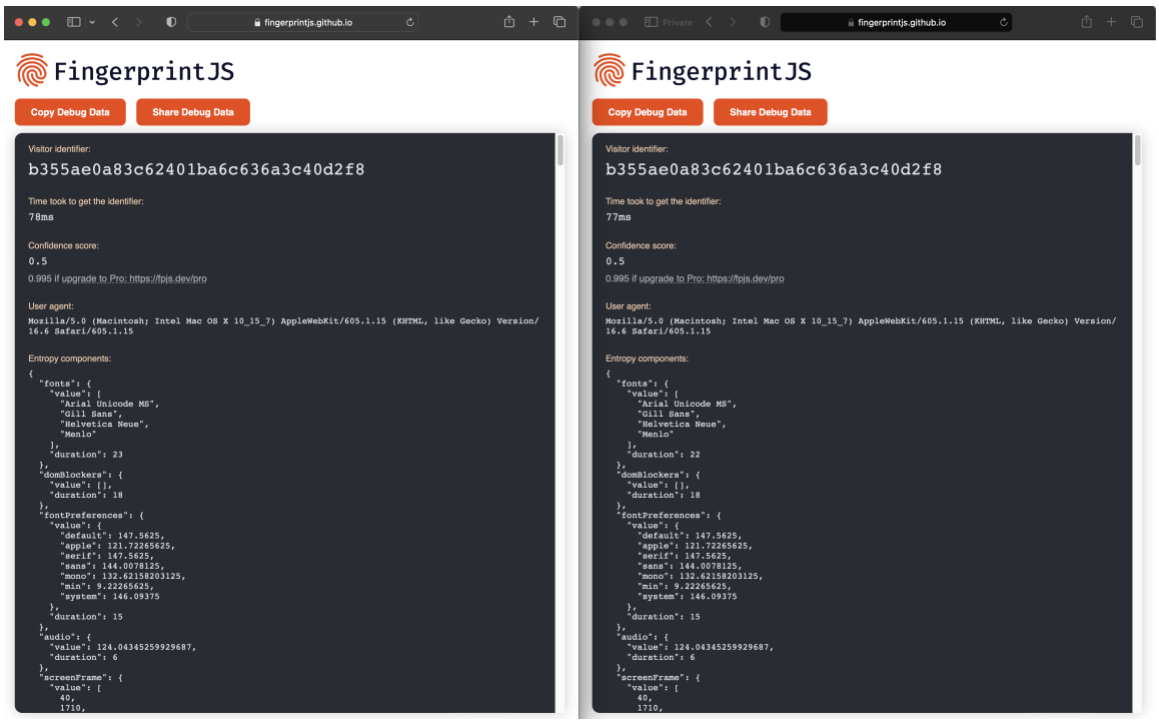


Figure 16

macOS AdBlock Tests with AdGuard

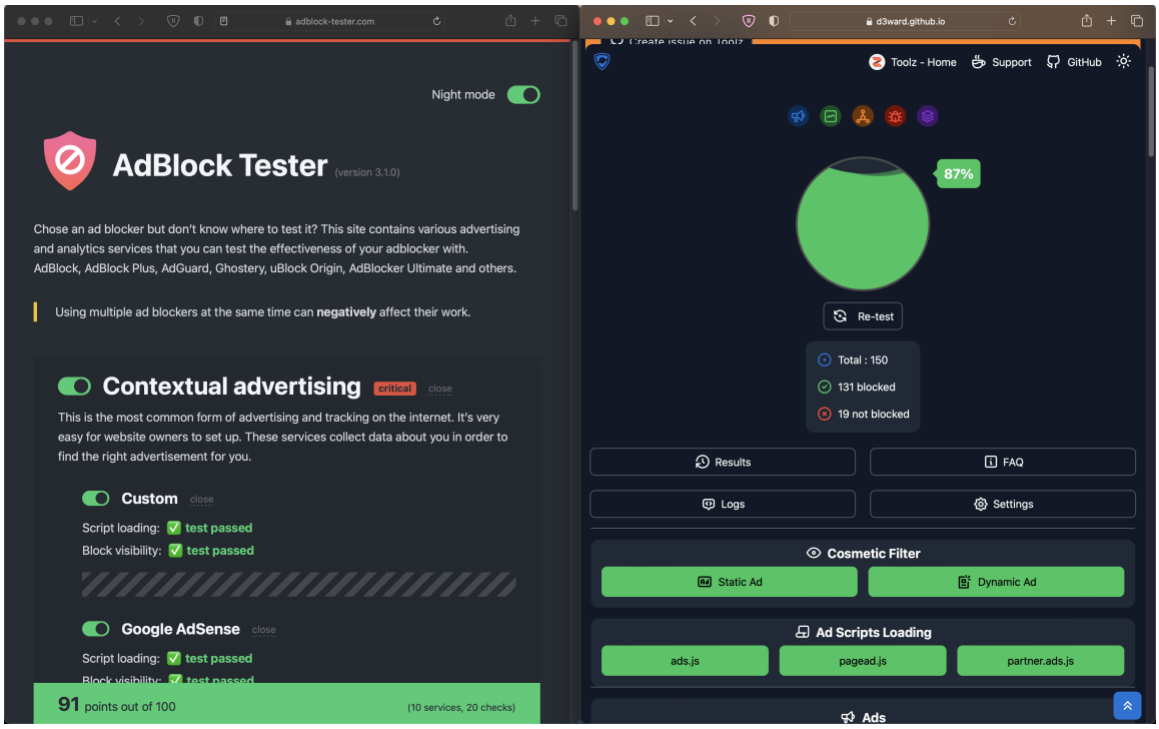


Figure 17

Firefox Adblocking Tests Out of the Box

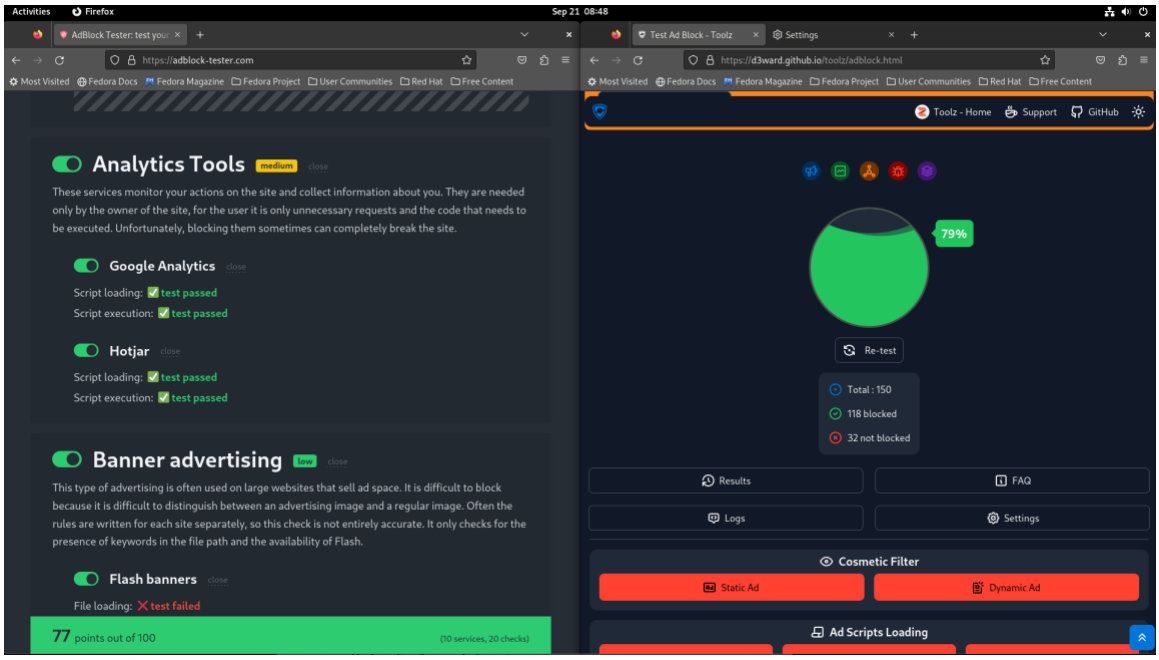


Figure 18

Firefox Cover Your Tracks Out of the Box

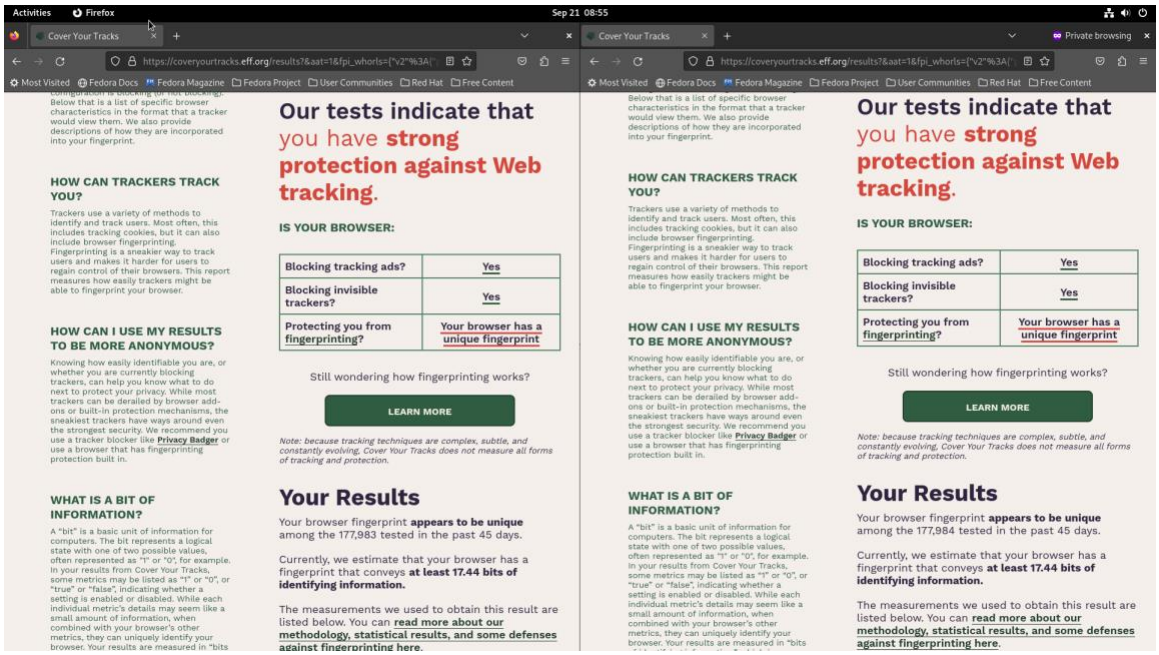


Figure 19

Firefox FingerprintJS Out of the Box

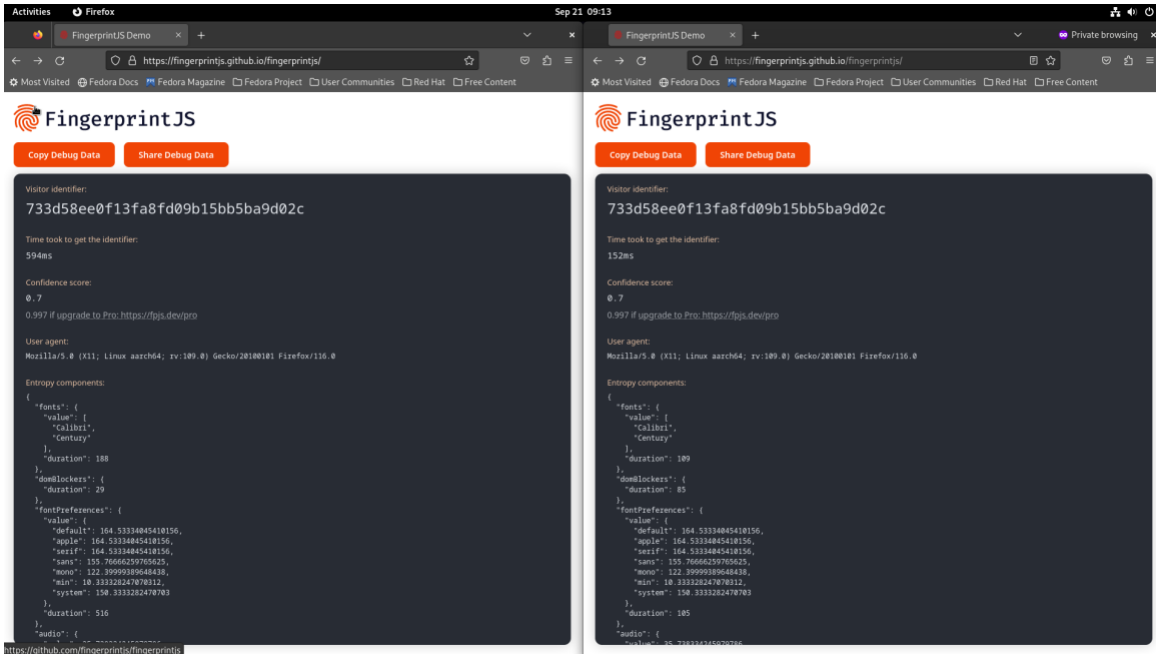


Figure 20

Firefox CreepJS Out of the Box

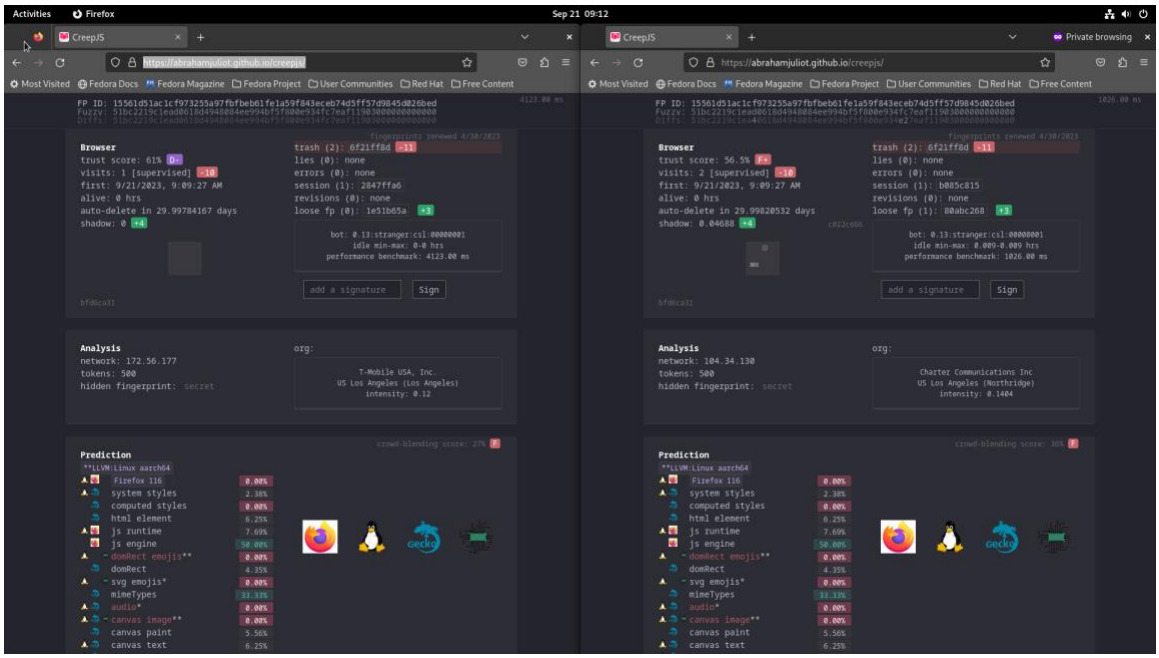
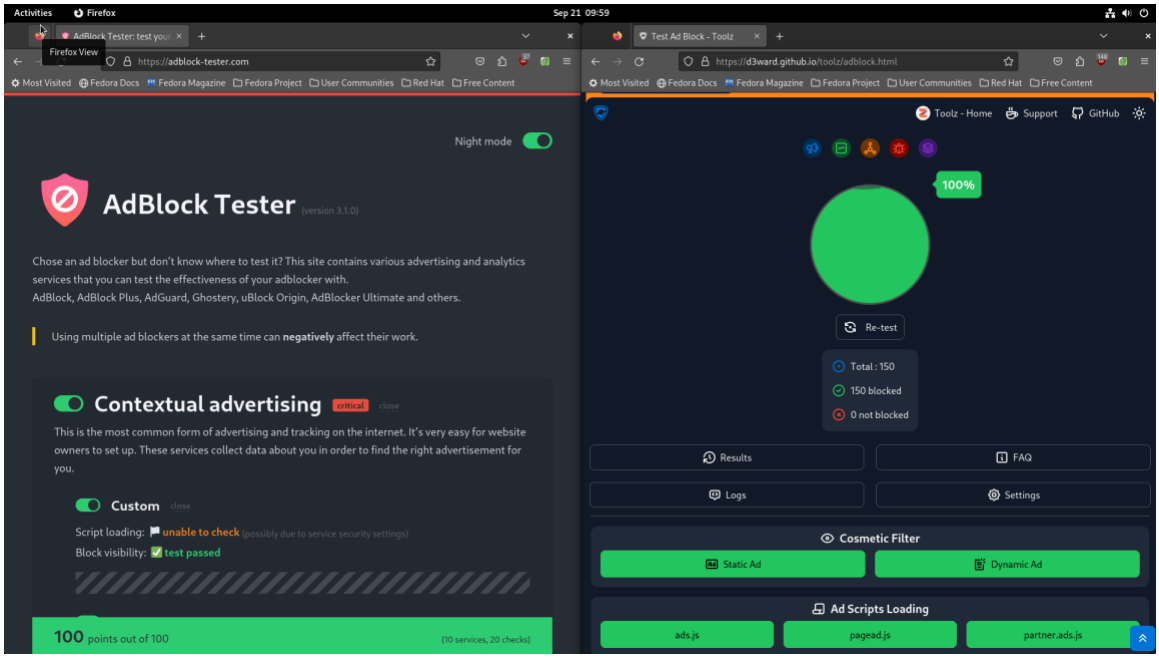


Figure 21

Firefox Adblock Tests with Ublock Origin



Issac Kim, issac.kim@student.sans.edu

Figure 22

Firefox FingerprintJS with CanvasBlocker

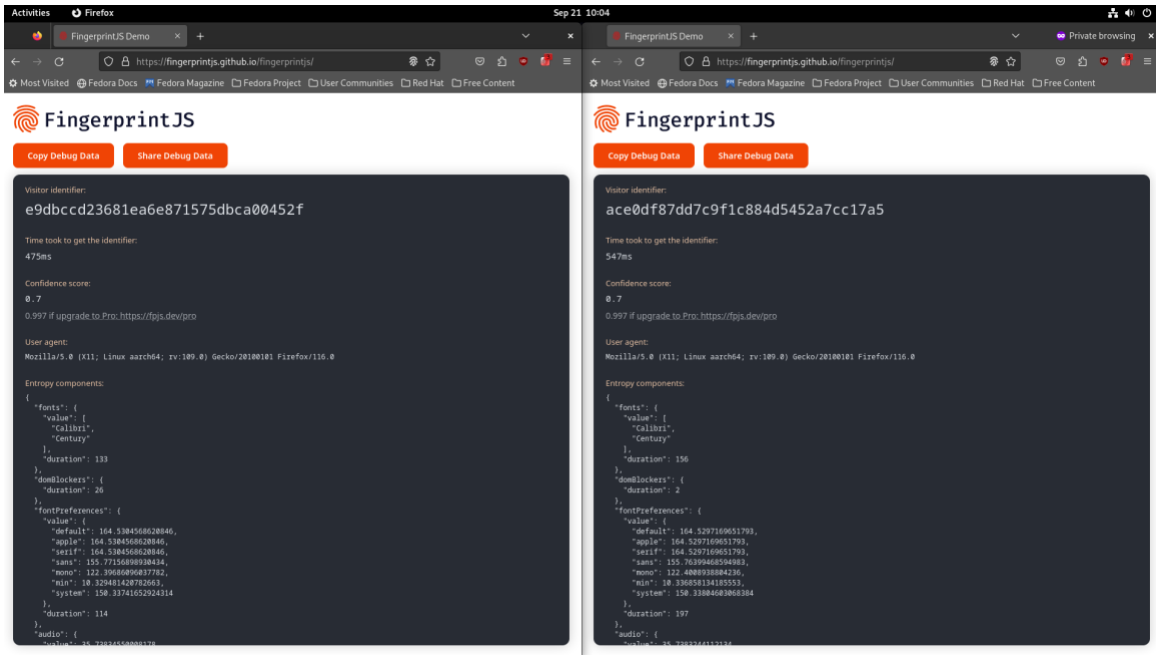


Figure 23

Firefox Cover Your Tracks with CanvasBlocker

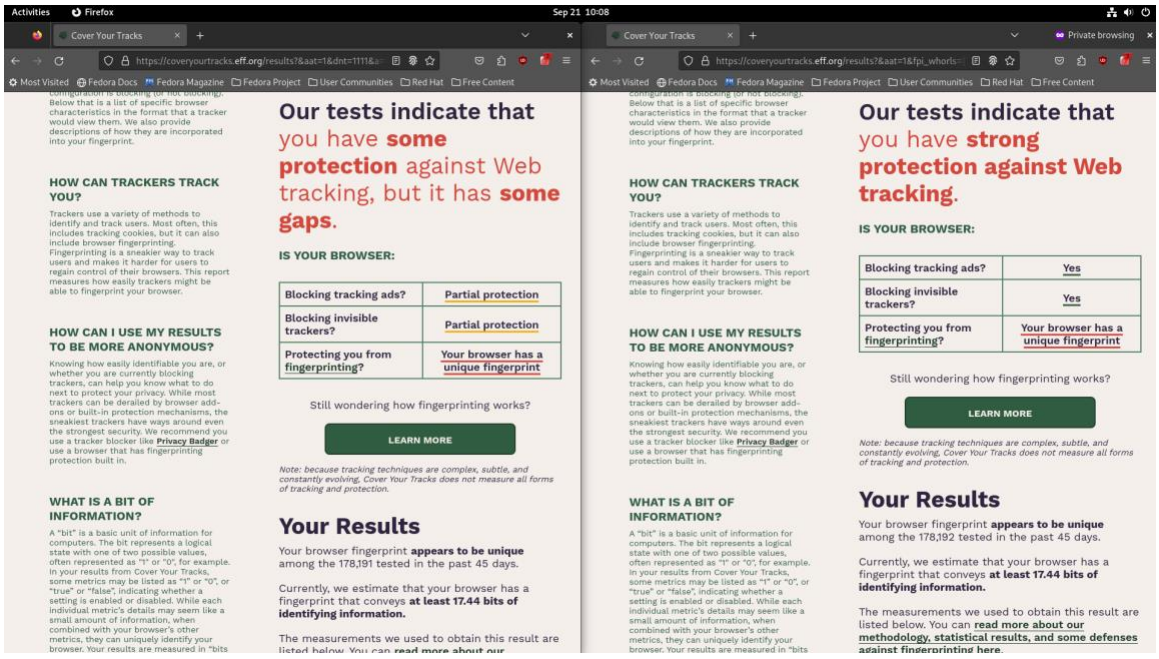


Figure 24

Firefox CreepJS with CanvasBlocker

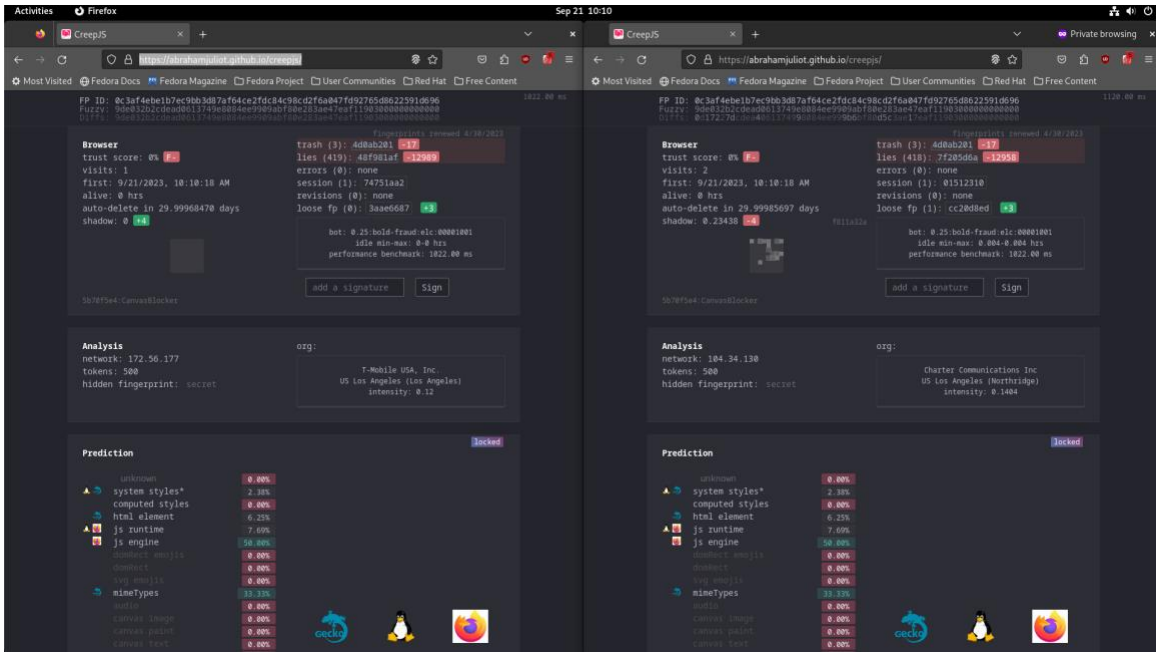
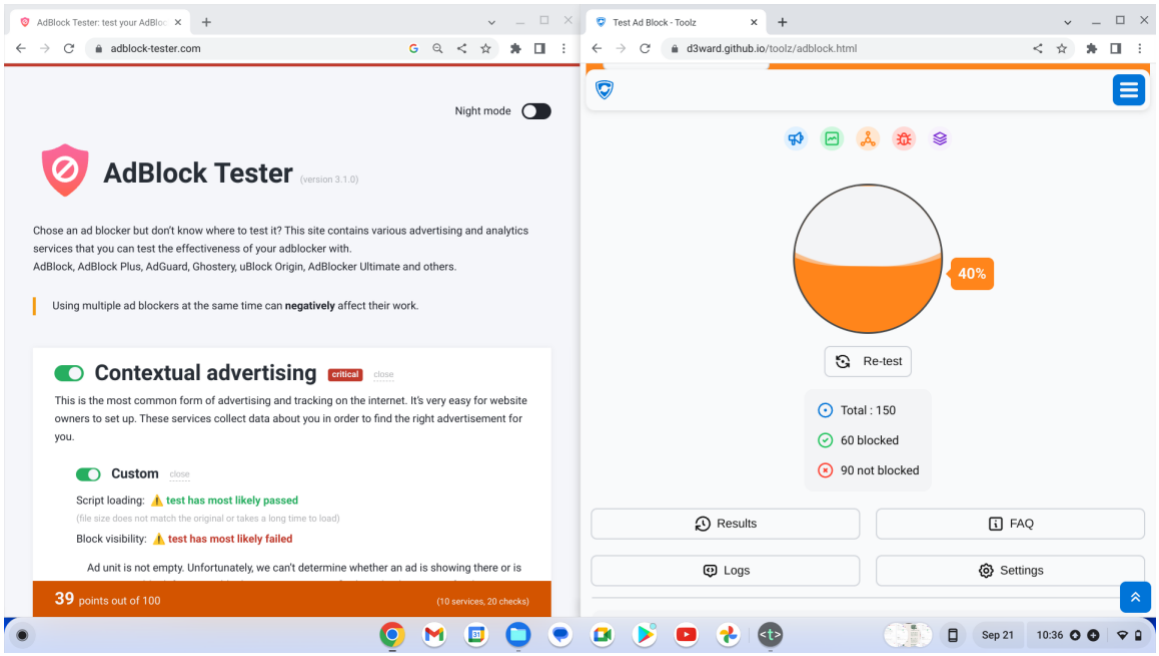


Figure 25

Chrome Adblock Testing Out of the Box



Issac Kim, issac.kim@student.sans.edu

Figure 26

Chrome Cover Your Tracks Testing Out of the Box

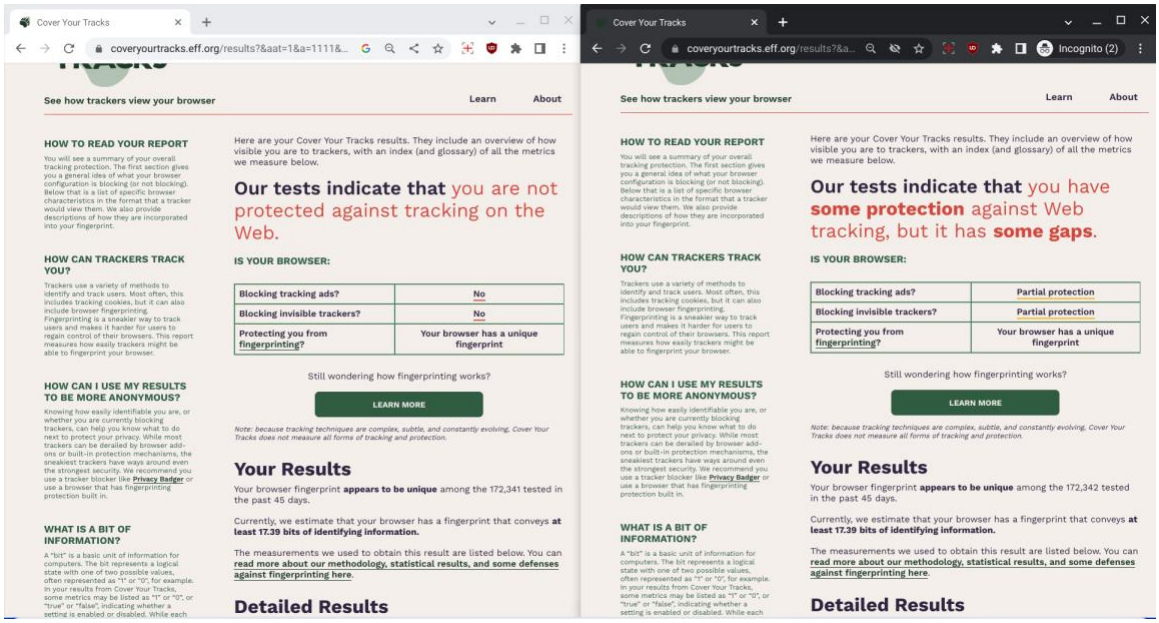


Figure 27

Chrome FingerprintJS Testing Out of the Box

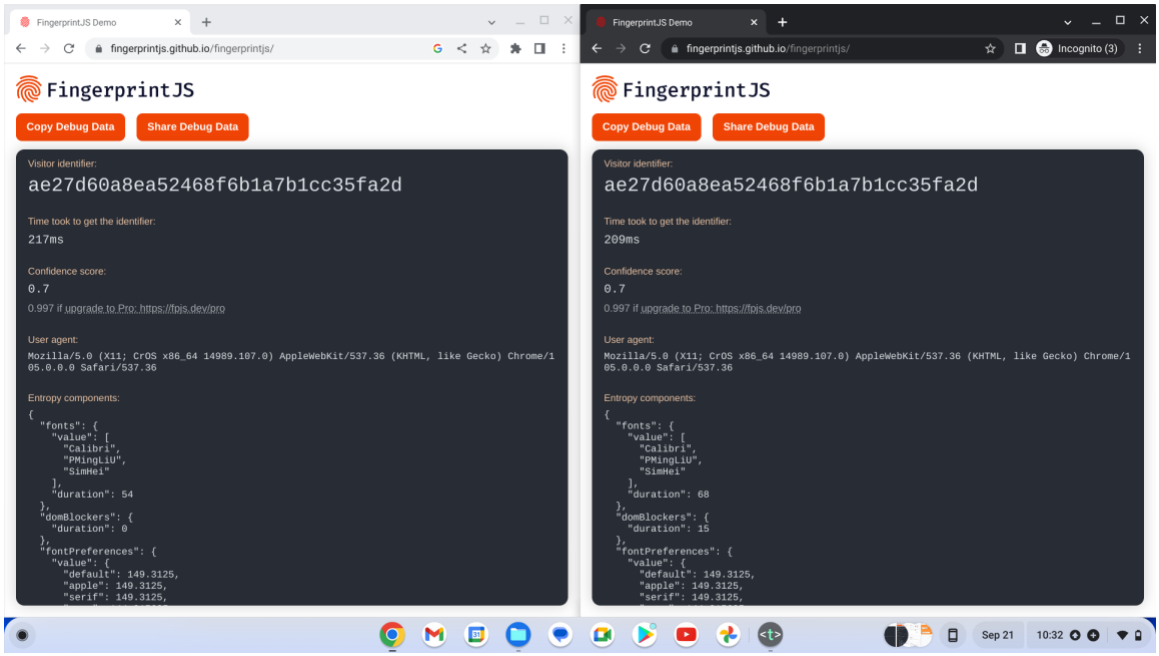


Figure 28

Chrome CreepJS Testing Out of the Box

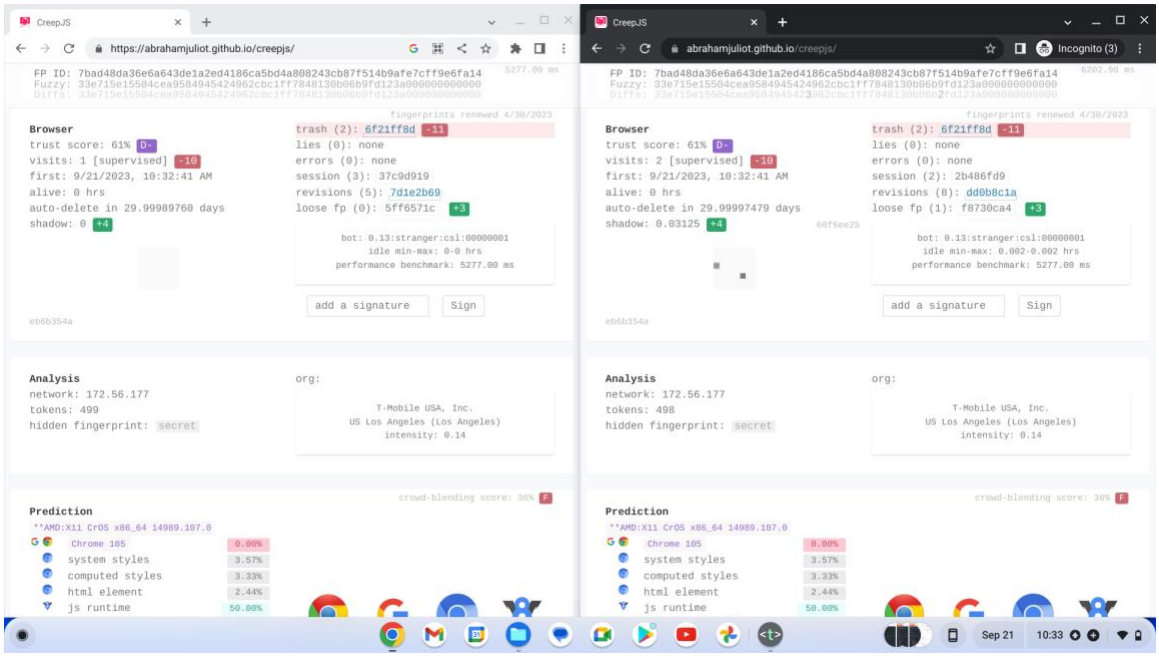


Figure 29

Chrome CreepJS Testing Out of the Box

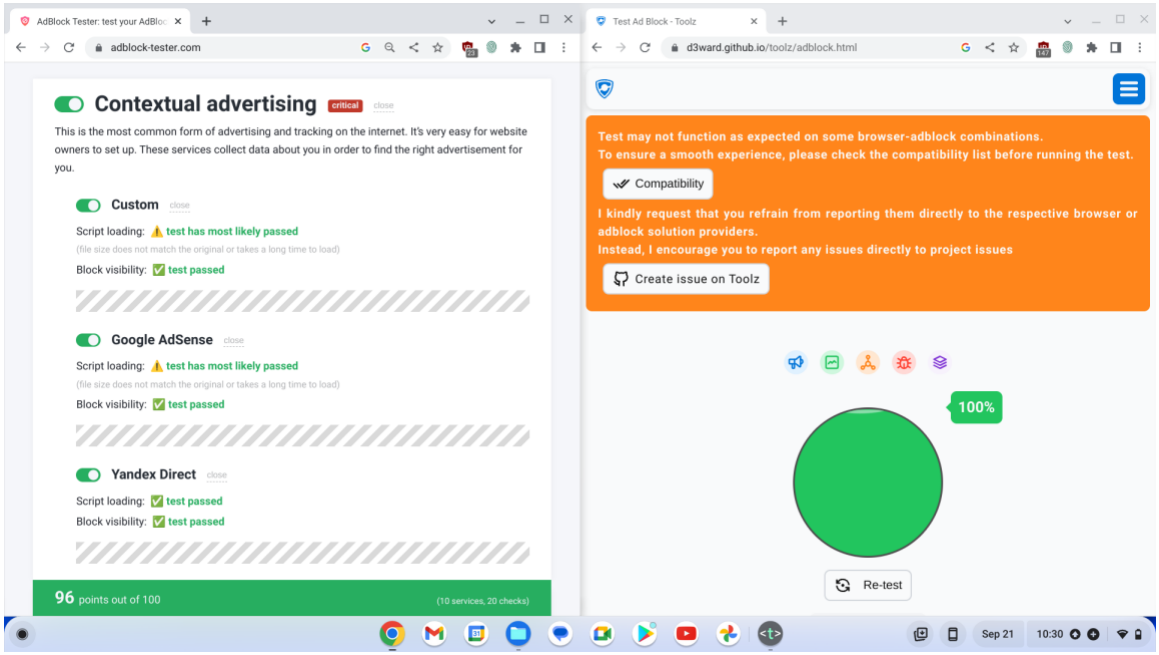


Figure 30

Chrome Cover Your Tracks with Extensions

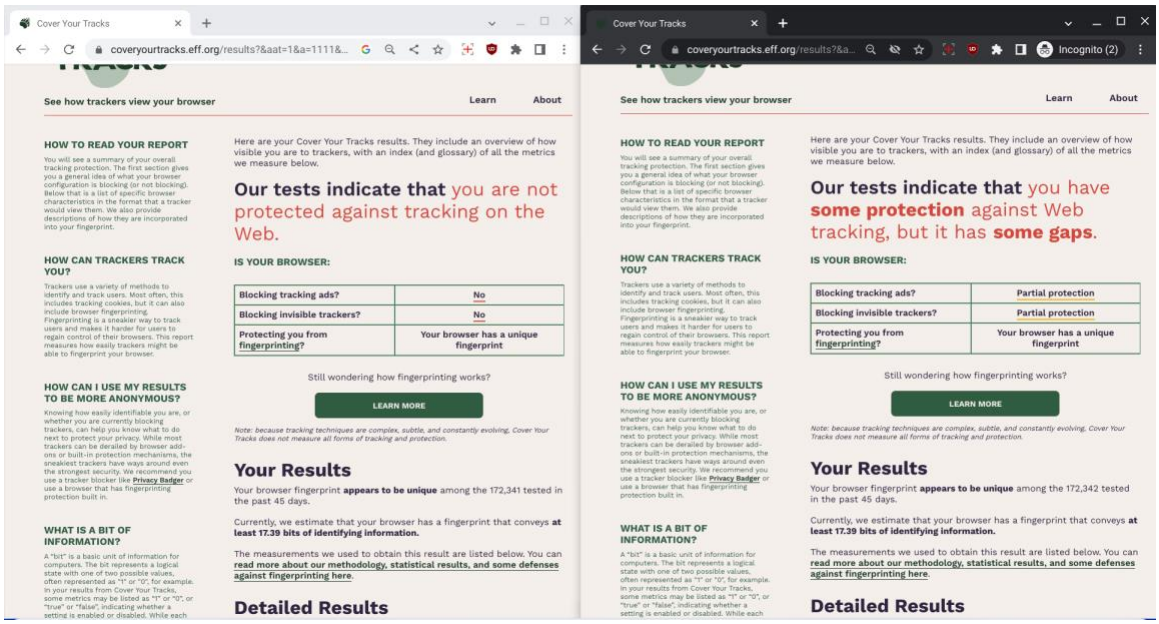


Figure 31

Chrome FingerprintJS with Extensions

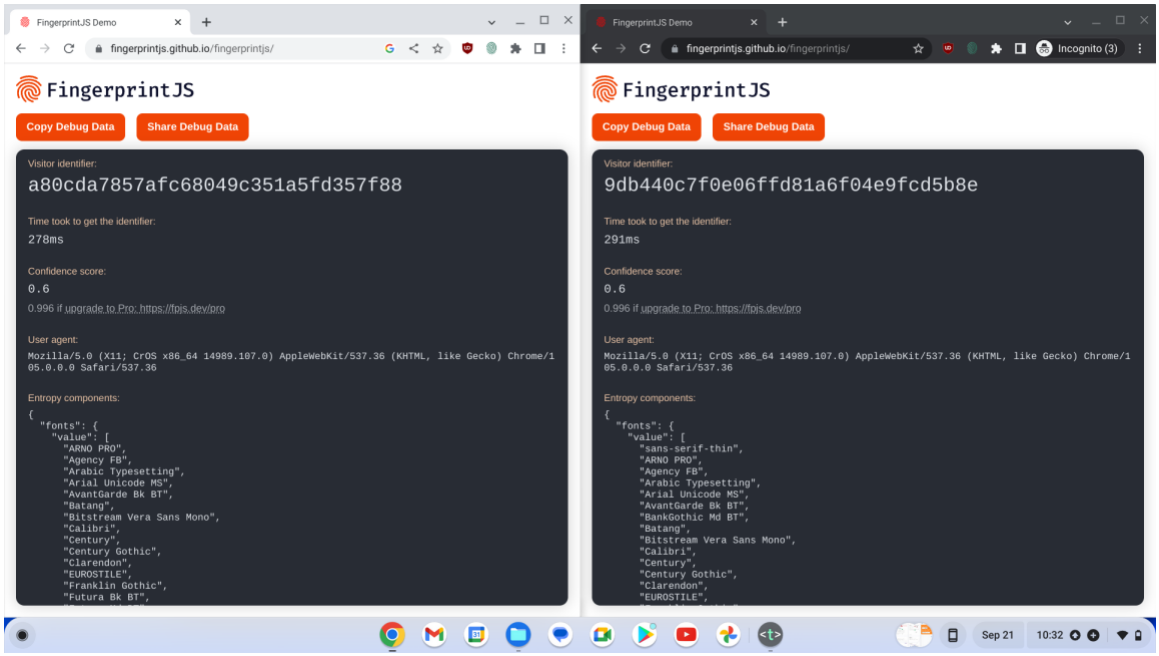


Figure 32

Chrome CreepJS with Extensions

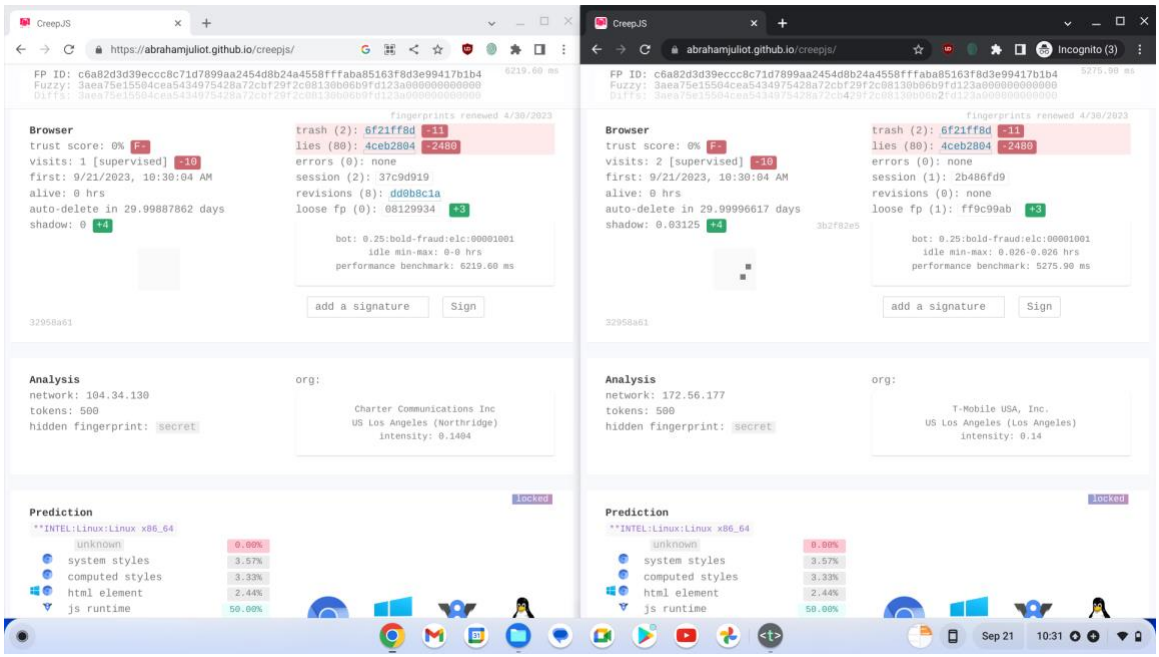
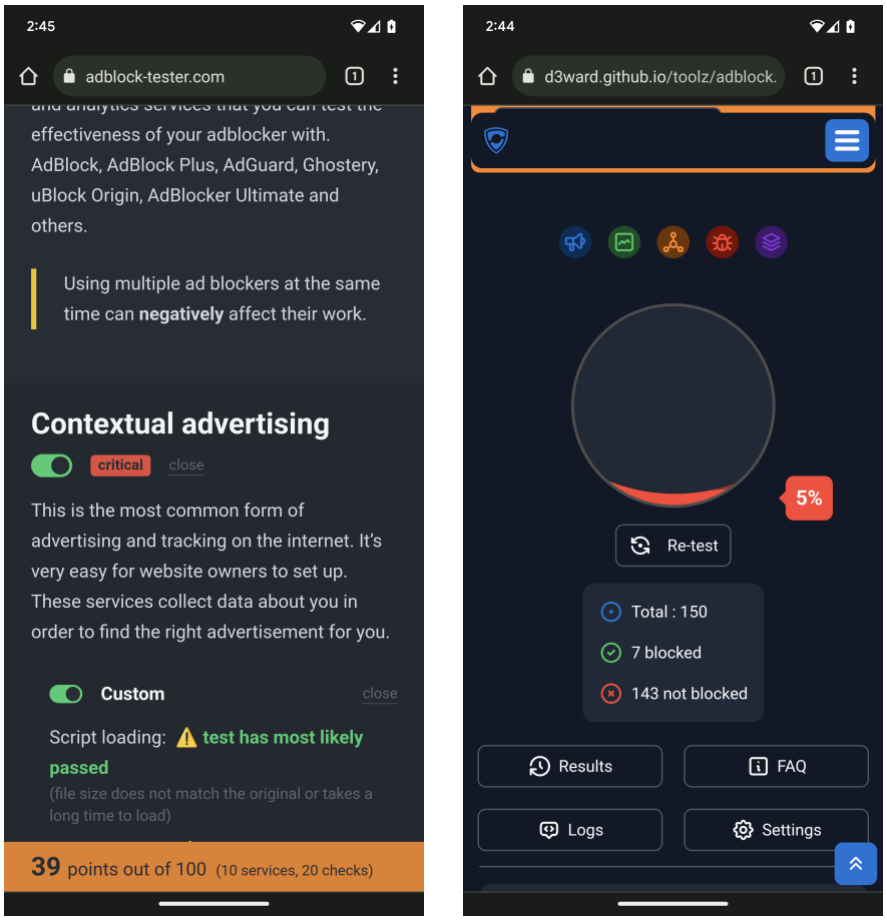


Figure 33

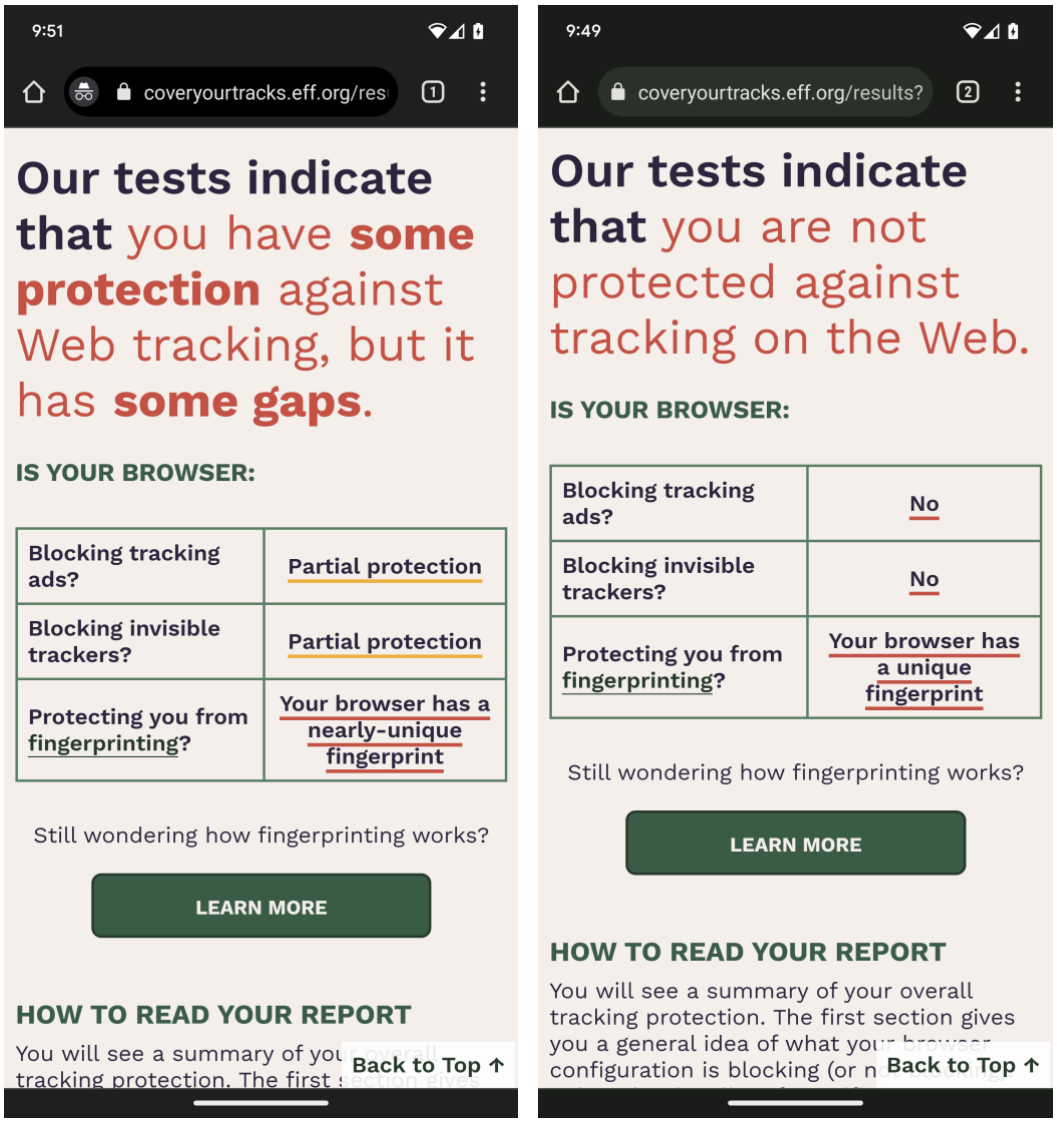
Android Adblock Test Out of the Box



Note. AdBlock Tester and Test Ad Block on default Android Chrome install

Figure 34

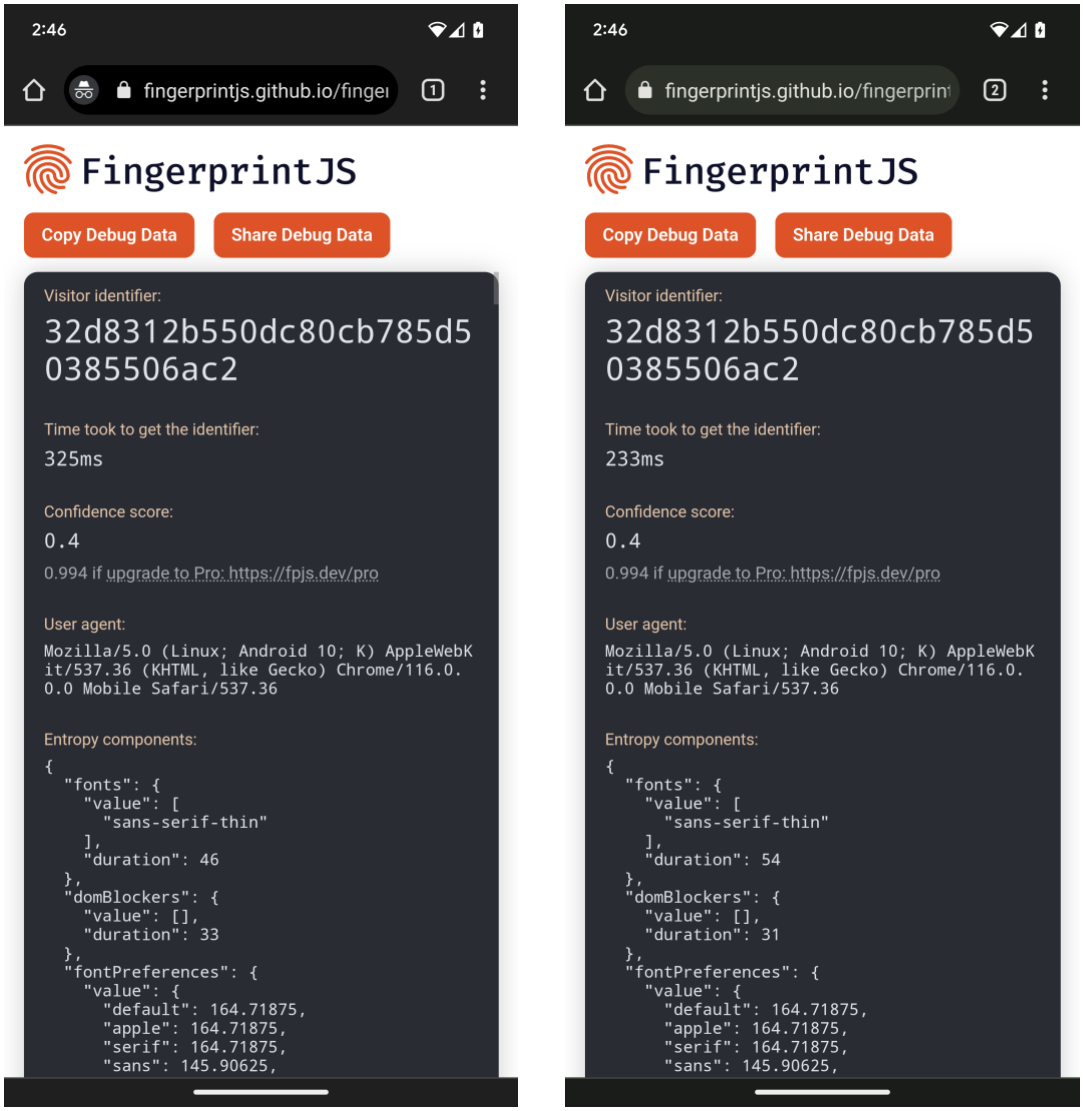
Android Cover Your Tracks Test Out of the Box



Note. Chrome demonstrating basic third-party cookie blocking in incognito mode, making the device's fingerprint slightly less unique.

Figure 35

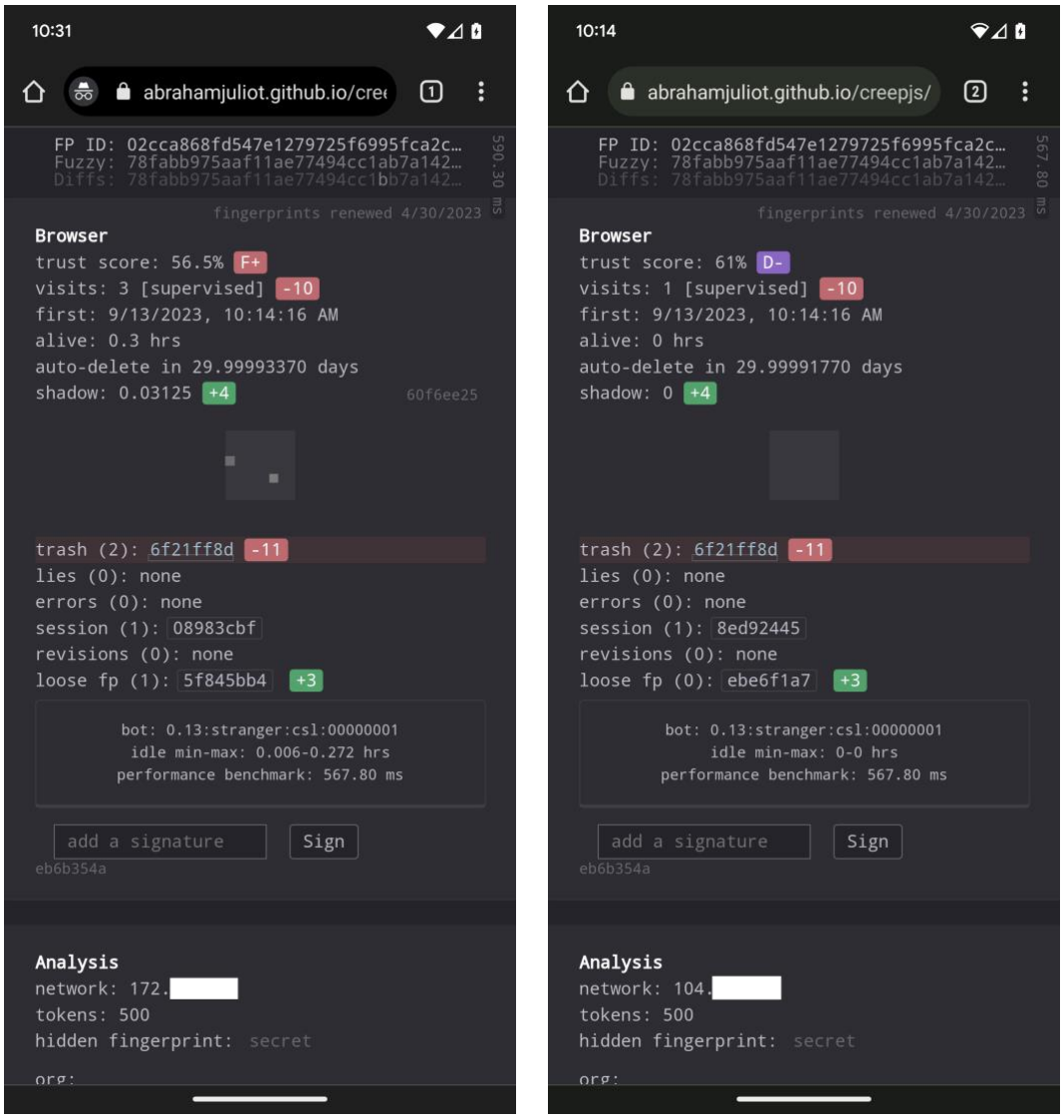
Android Fingerprint Test Out of the Box



Note. FingerprintJS demonstrating the unique identifier regardless of session or location on default Chrome install

Figure 36

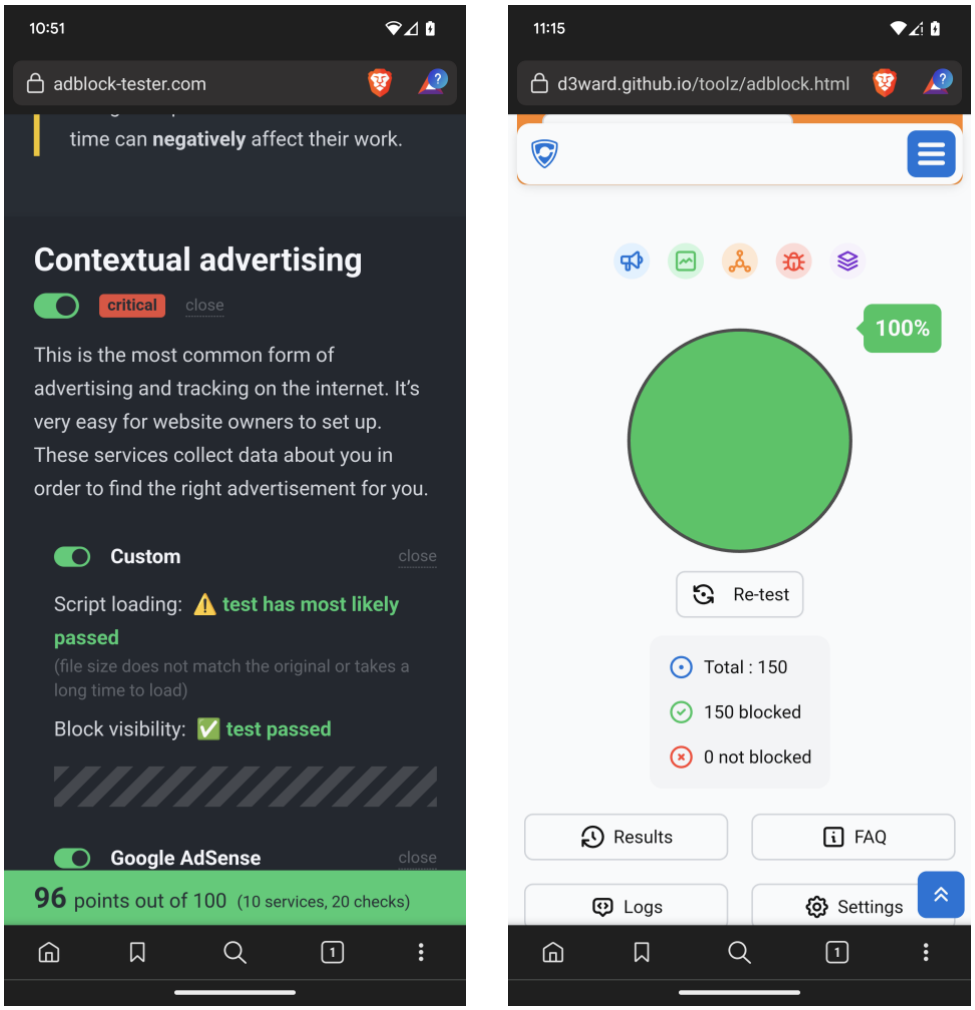
Android CreepJS Test Out of the Box



Note. CreepJS is able to develop an identifying hash regardless of browsing mode and network.

Figure 37

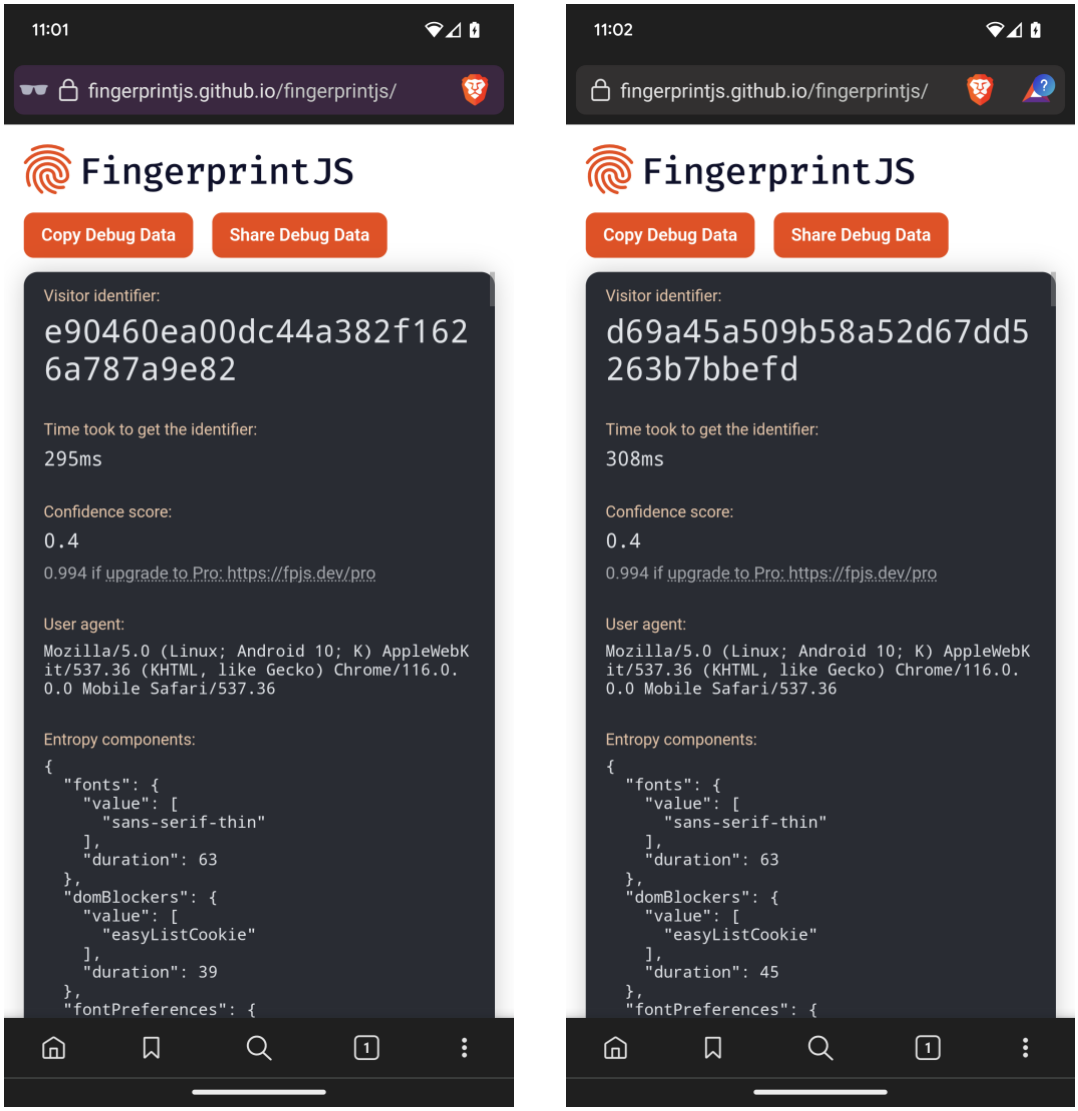
Android Adblock Test with Brave



Note. AdBlock Tester and Test Ad Block on privacy-configured Android Brave install

Figure 38

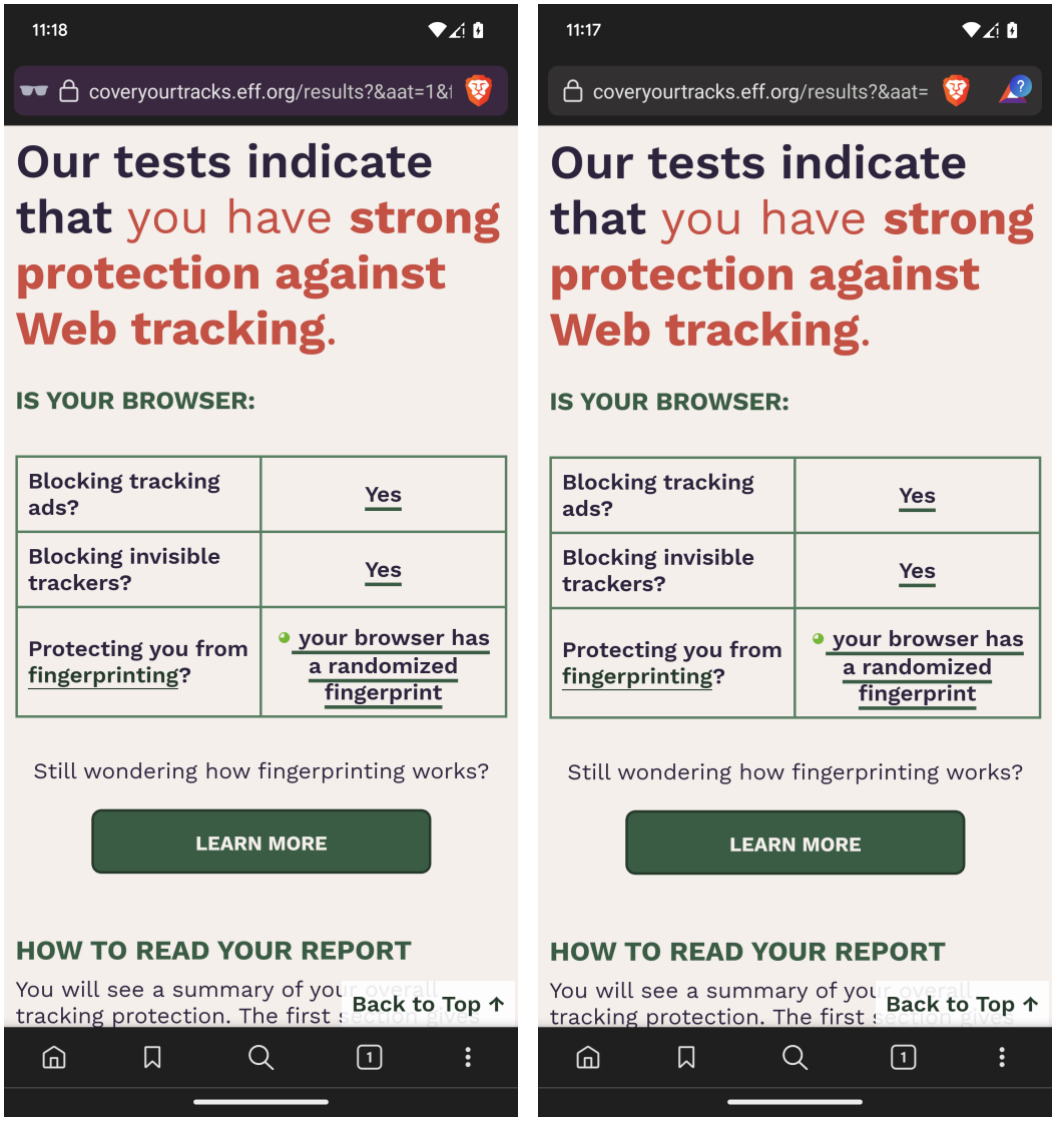
Android Fingerprint Test with Brave



Note. FingerprintJS demonstrating the browser's ability to defeat fingerprinting and appear to be a different client every session.

Figure 39

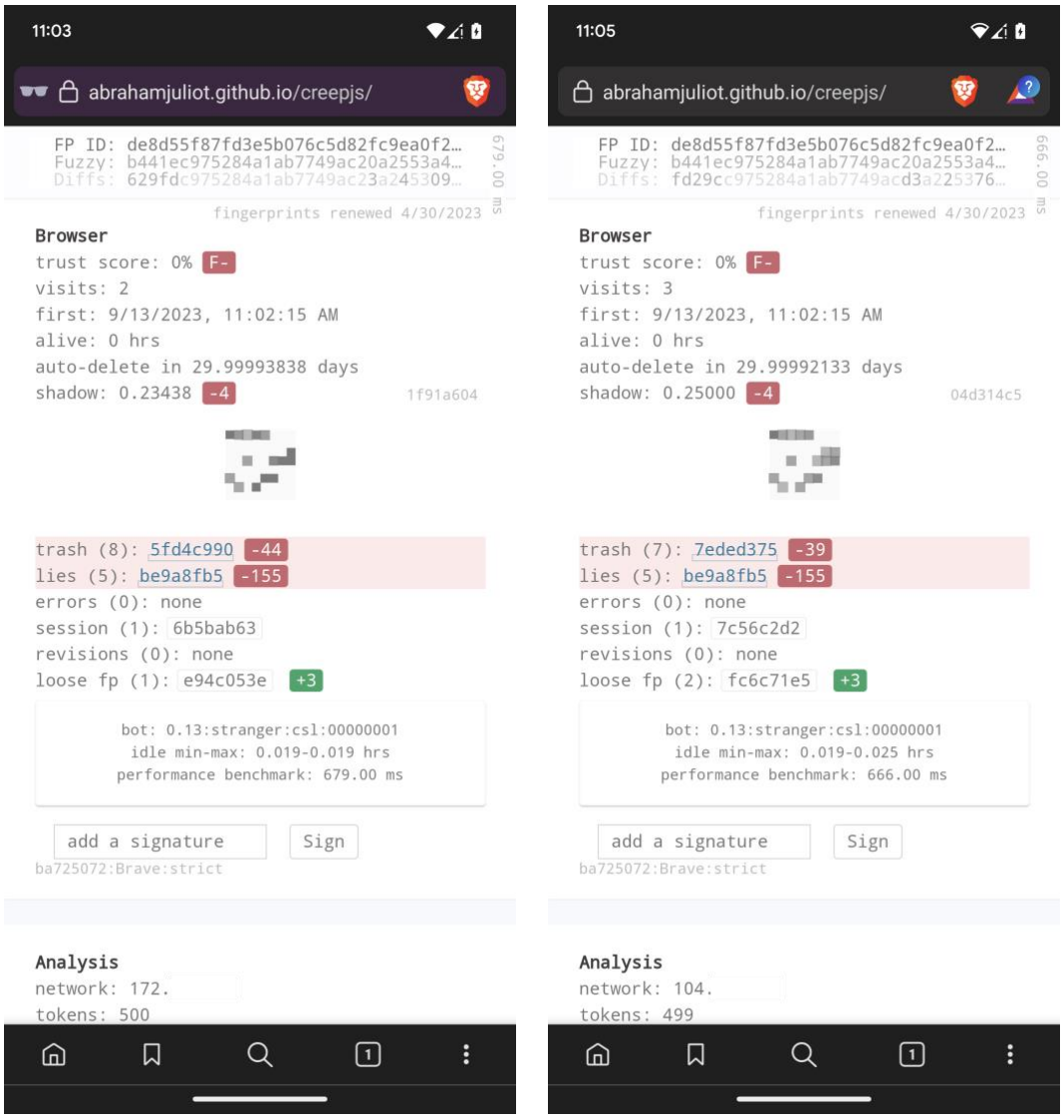
Android Cover Your Tracks Test with Brave



Note. Brave demonstrating that its fingerprint is seen as random with each session.

Figure 40

Android CreepJS Test with Brave



Note. CreepJS is still able to maintain an identifying hash of the user across sessions and networks. See description of caveat with CreepJS's detection above.

Figure 41

iOS 17 Adblock Tests Out of the Box

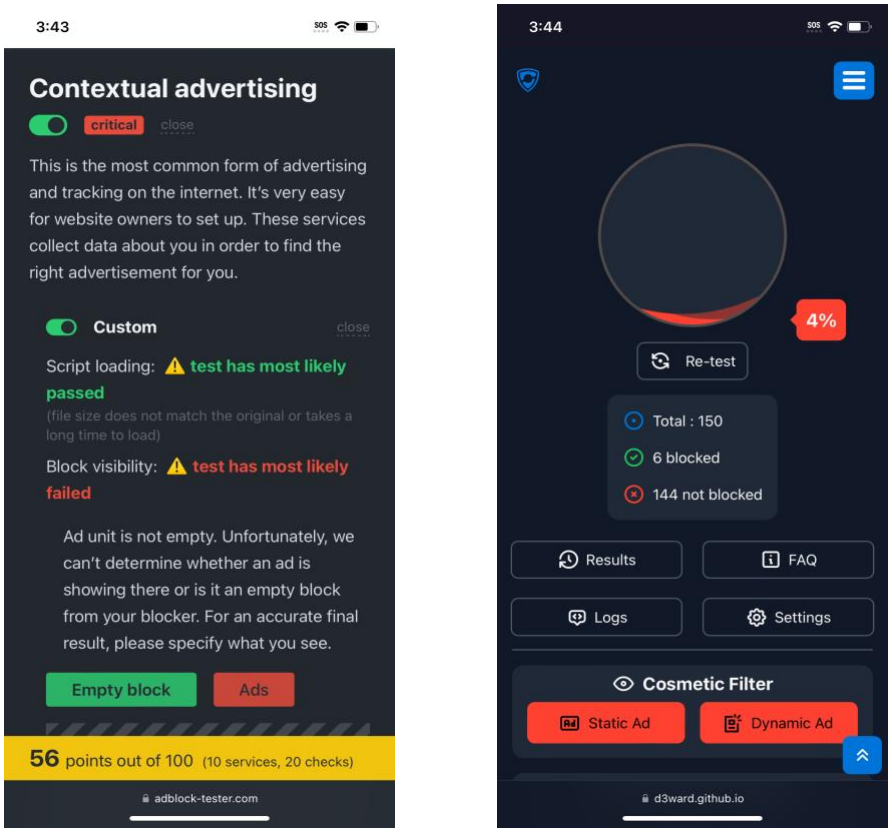
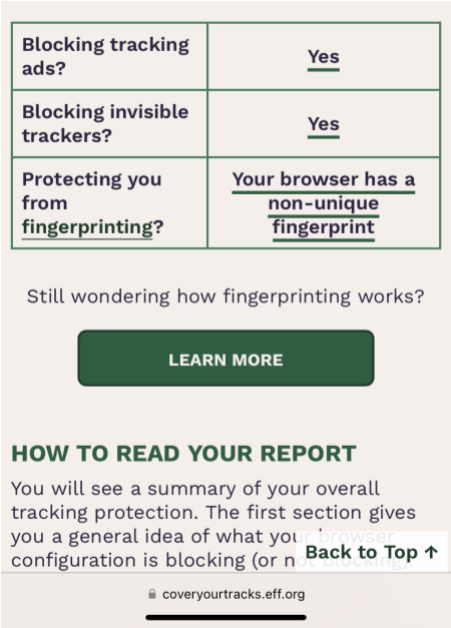


Figure 42

Cover Your Tracks Identifying Hashes (Safari on macOS)

Figure 43

Cover Your Tracks iOS 17 Out of the Box



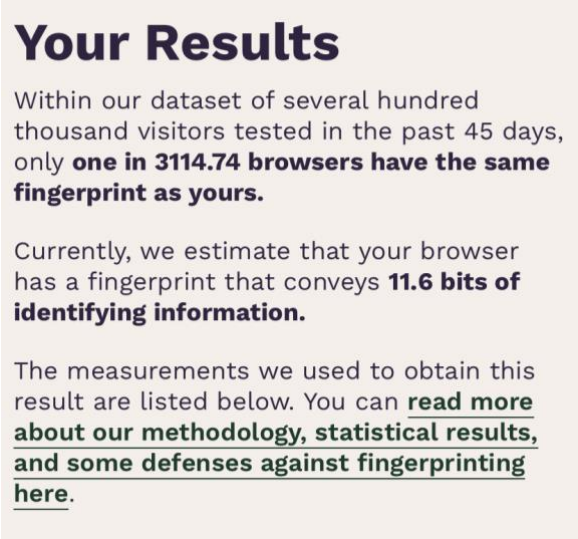


Figure 44

FingerprintJS iOS 17 Out of the Box

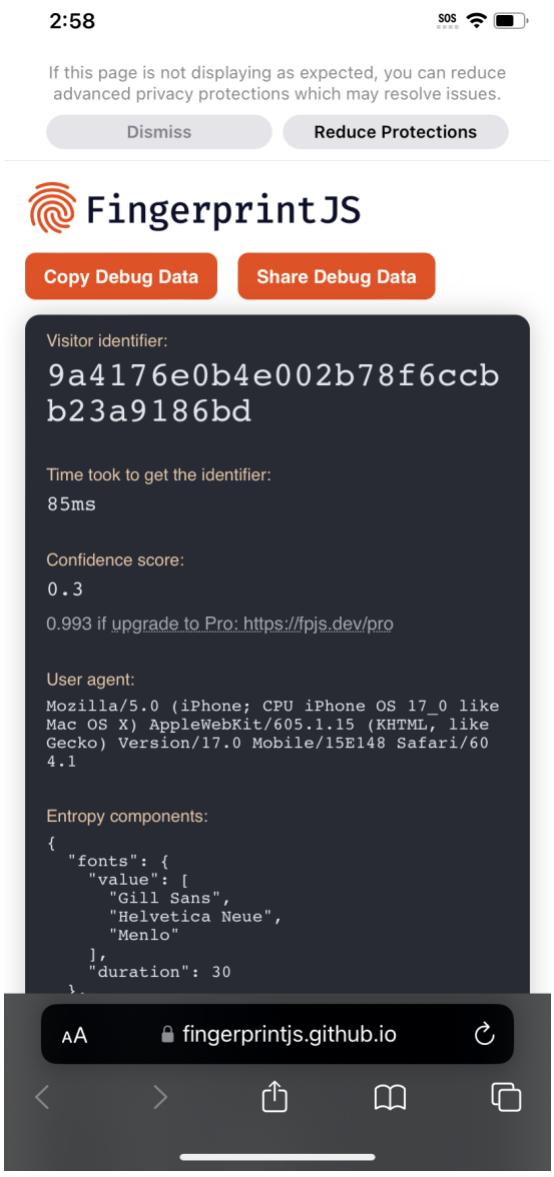


Figure 45

FingerprintJS Canvas Test (macOS)

Figure 46

FingerprintJS Canvas Test (iOS 17)

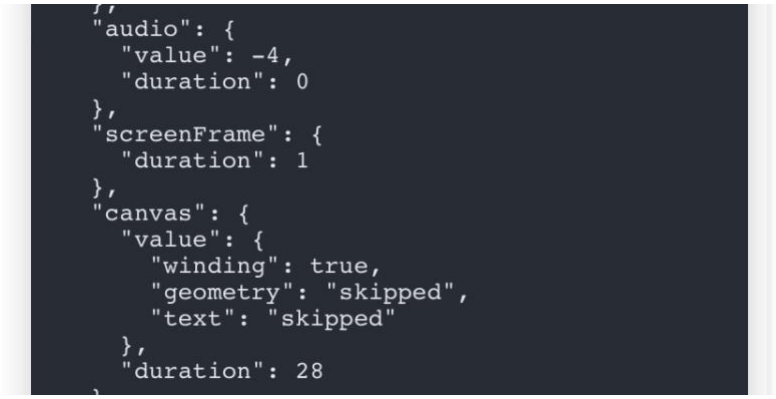


Figure 47

CreepJS iOS 17 Out of the Box Test

Figure 48

Ad Block Tests on Brave iOS

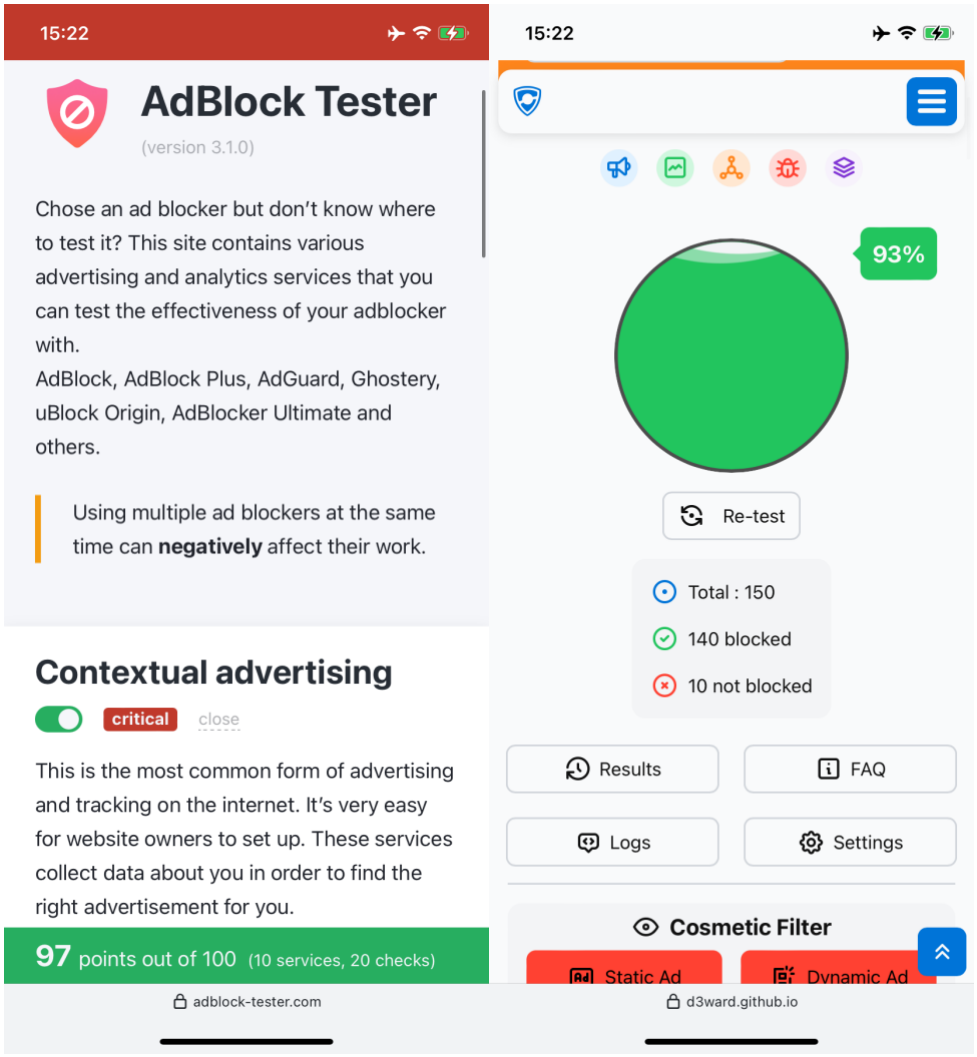
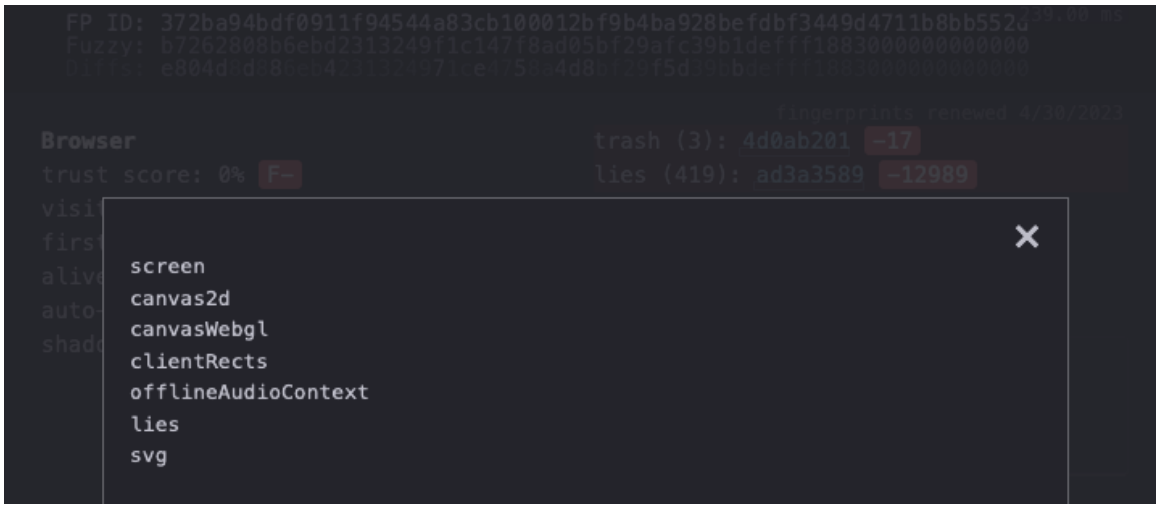


Figure 49

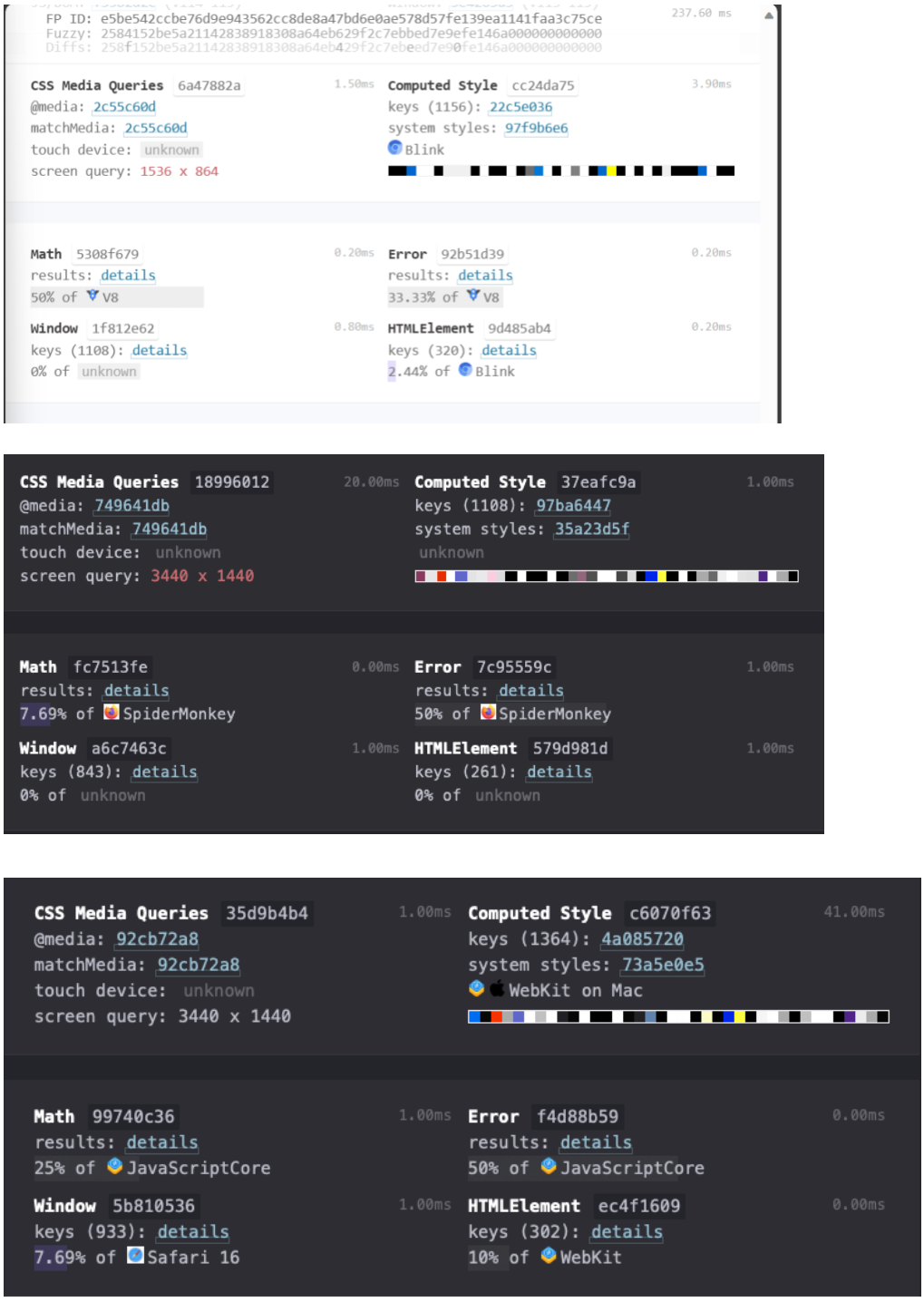
CreepJS "Lie" Detection



]

Figure 50

CreepJS JavaScript Engine Detection



Note. In order from top to bottom: Microsoft Edge, Firefox on Linux, desktop Safari on macOS.