A Law HawCS Approach to Data Protection and Privacy

Elias Daoud, Lucas Ausberger, Hibah Lateef, and Serena Qamhieh

University of Iowa

# 

*Our thanks goes to both Professors Rishab Nithyanand and Mihailis Diamantis for their instruction and guidance.*

# ABSTRACT

Cookies may not be so sweet after all. According to a recent Pew report, 79% of Americans are concerned about the way companies use the information stored on their browsers after visiting sites. Concerns about data collection and use have changed the way people interact with cookies. The Pew report shows that 41% percent of U.S. consumers regularly delete cookies, and 30% have installed an adblocker. To continue tracking web users, third parties such as advertising companies and data brokers, have begun to rely on digital fingerprinting. Fingerprinting allows companies to track users through digital scripts that recognize the small differences which set one’s web browser apart from another’s.

To prevent advertising companies from successfully taking user information, we propose a browser with two components. The first component targets stateful tracking through the interception and modification of information as it travels from the user’s device to a website. The second component handles stateless tracking by altering browser functionality itself in order to falsify information given to websites.

We also analyze whether a user of our privacy tool would incur any legal ramifications. First, we analyze fraud under the Computer Fraud and Abuse Act (CFAA) and mail and wire fraud. Because our privacy tool sends out false information to website regulators, we analyze whether our tool impermissibly obtains “anything of value” under the CFAA. Our tool also uses dishonest means and schemes to prevent ad companies from obtaining data, and we analyze whether our tool wrongs someone in his “property rights.”

Secondly, we analyze the deployment of our tool within the current notice and consent framework of data privacy and the possibility of a breach of contract claim resulting from our tool use. Although users are generally given notice that the website they are about to use tracks through cookies, notice is not given for tracking though digital fingerprinting. Adding notices that users are being tracked through digital fingerprinting will likely lead to the same legal problems notices of cookie tracking provide. We begin by researching what policies courts are willing to accept as cognizable contracts as opposed to mere information. Finally, we analyze the language of Terms of Service agreement to see whether users would be in violation of its terms by using the privacy tool.

# INTRODUCTION

You are being followed. Just about every time that you pull up a new web page while browsing the internet, third parties, such as advertising companies and data brokers, track the content you engage with in order to sell your data for a profit or provide you targeted and personalized advertisements later.[[1]](#footnote-0) Americans sense a loss of control over their data. “Roughly eight-in-ten or more U.S. adults say they have very little or no control over the data … companies (81%) collect about them.”[[2]](#footnote-1) Furthermore, “81% of Americans think the potential risks of data collection by companies about them outweigh the benefits…”[[3]](#footnote-2) Mass data collection and limited regulation raises concerns about the adequacy of privacy protections for web users.

The benefits that attract everyday web users also attract third-party cross-site tracking which can follow web users after closing out of a website. Cross-site tracking refers to the practice of third party companies collecting browser activity and data from users in ways that impact the types of ads received on another, separate device.[[4]](#footnote-3) Cross-site tracking can be done in two ways: Statefully and statelesslly. Stateful tracking involves monitoring user-identifiable data stored on a user’s computer through Cookies and was meant to enhance website usability by allowing websites to remember user interactions through functions such as password saving. [[5]](#footnote-4)

As web users become more privacy literate, users change their behavior. The Pew report shows that 41% percent of U.S. consumers regularly delete cookies.[[6]](#footnote-5) In response, however, third parties have employed other methods to continue collecting data. Stateless tracking, otherwise known as digital fingerprinting, digital fingerprinting, does not store identifiable information on the user’s computer. Rather, scripts are used to recognize the small differences which set one person's web browser apart from another’s which can be compiled into a profile of the person.[[7]](#footnote-6)

“[F]ully 79% of adults say they are at least somewhat concerned about how companies are using the data it collects about them.”[[8]](#footnote-7) Web users’ concerns are not unfounded. The United States currently does not have any comprehensive, federal policy that regulates how third parties can collect and use massive amounts of web user data.[[9]](#footnote-8) The lack of comprehensive policy leaves private entities to use data in ways that harm web users. One example is Epsilon Data Management LLC (Epsilon). Epsilon is one of the largest marketing companies in the world. Recently, the marketing company had been selling consumer data to entities Epsilon knew were engaged in elder fraud schemes.[[10]](#footnote-9) In January of 2021, Epsilon agreed to pay $150 million dollars in a deferred prosecution after the Department of Justice charged Epsilon with one count of conspiracy to commit mail and wire fraud.[[11]](#footnote-10) $127.5 million of the settlement amount was designated to compensate the elderly victims of the fraud schemes.[[12]](#footnote-11)

Private parties, such as Epsilon, are not the only parties to be concerned about. Police can subvert Fourth Amendment protections against unreasonable searches and seizures by piggybacking off the work of data collectors. Generally, police must obtain a warrant supported by probable cause before a search or seizure in order to protect peoples’ reasonable expectations of privacy.[[13]](#footnote-12) However, under a doctrine called the third-party doctrine, police are not required to obtain a warrant before requesting information shared with a third party.[[14]](#footnote-13) The assumption underpinning the third-party doctrine is that those who voluntarily give information to third parties no longer have a reasonable expectation of privacy.[[15]](#footnote-14) This assumption of voluntariness, however, is concerning considering that the profitability of advertising and data brokering, tracking has become so deeply embedded into the modern web experience making it nearly impossible for web users to not turn over their data. While there are some limits on how far the third-party doctrine extends, doctrine’s application is still quite broad. For example, the Supreme Court has held that cell phone location records are unique in nature and do not automatically trigger the third-party doctrine.[[16]](#footnote-15) However, “[i]t is unclear what, if any, other sort of data shares that nature—its “uniqueness” suggests the answer may be none.”[[17]](#footnote-16)

Even without the third party doctrine, personal data is merely one transaction away from governmental entities. For example, Muslim Pro is an app that reminds users when and in which direction to pray.[[18]](#footnote-17) X-Mode is a company that collects and sells location directly from apps like Muslim Pro. The U.S. military obtained location data on muslims by simply purchasing the data from X-Mode.[[19]](#footnote-18) Separately, under former President Donald Trump, the administration “... bought access to a commercial database that maps the movements of millions of cellphones in America and … [was] using it for immigration and border enforcement.”[[20]](#footnote-19)

One approach to protecting data privacy is through anonymization. Anonymization, however, can be difficult and ineffective. Even without directly identifying information such as a person’s name or social security number, “... 87% (216 millioh of 248 million) of the population in the United States [can] be uniquely identified if their 5-digit zip code, date of birth and gender is known.”[[21]](#footnote-20)

Another approach to giving users more control over their data is through comprehensive policy reform. “When asked how much government regulation there should be around what companies can do with their customers’ personal information, 75% of adults say there should be more regulation than there is now.”[[22]](#footnote-21) In 2018, the EU enacted the General Data Protection Regulation (GDPR). The GDPR imposes obligations on organizations that collect data related to people in the EU by using tactics such as high fines for entities that breach the privacy regulations, “…robust consent requirements, privacy by design, and mandatory breach notifications.”[[23]](#footnote-22) While the United States has not enacted its own data protection policy, Congress has previously considered bi-partisan privacy bills. Aside from Congress, the Uniform Law Commission, a national, non-partisan, non-profit, created a model code called the Uniform Personal Data Protection Act (UPDPA). While the UPDPA does not lean as heavily on fines as the GDPR, the proposed policy places more protections on the collection and use of sensitive information then currently available.[[24]](#footnote-23) The issue with a policy reform approach is not a lack of ideas on how to protect web users. The issue, rather, is political feasibility. When Congress has considered comprehensive privacy legislation, disagreement on whether the legislation should preempt state policies and whether the legislation should provide a private right of action to individuals has killed its passage.[[25]](#footnote-24)

In the time that it takes to mobilize and enact policy reform, a different approach to protecting privacy is required. We approach the web based, privacy vulnerability of digital tracking by equipping web users with a browser with two privacy components intended to thwart both stateful and stateless tracking. The first component targets stateful tracking through the interception and modification of information as it travels from the user’s device to a website. The second component handles stateless tracking by altering browser functionality itself in order to falsify information given to websites. We took this information-based approach in order to influence the set of data used to associate a user with a consumer profile for the purpose of targeted advertisement. After constructing the tool, we then analyzed the degrees to which we are able to affect the advertisements received while using either privacy component. If we are able to achieve a significant measurement, it will have promising implications with respect to fighting back against tracking.

Because our proposed privacy tool modifies the data trackers receive, we analyze the potential legal ramification of fraud under the Computer Fraud and Abuse Act (CFAA). The CFAA is a federal law. While enacted to address hacking, the CFAA, due to its ambiguous language, can be interpreted in an overly broad manner and allow the regulation to be weaponized. For example, in the case *United States v. Drew and United States v. Nosal*, the government argued that an employee violating the terms of a confidentiality agreement set by a company was a violation of the CFAA. For our privacy tool, we analyze whether a user obtains anything of value by presenting marketing and advertising companies with false information.

We also analyze our tool under Wire Fraud. An individual commits Wire Fraud when they dishonestly scheme to defraud someone and wrong them in their property rights. We analyze whether the data our privacy tool is reconfiguring is the property of the ad companies and browsers, or the user.

We also consider the use of our tool within a notice and consent framework for data protection and privacy as well as the potential legal ramification of breach of contract. Under the notice and consent framework, trackers present web users with a notification that the website uses tracking.[[26]](#footnote-25) The user is then given the option to read over the tracker's privacy policy and has the opportunity to decide whether they would like to be tracked. Sometimes, however, courts construe certain policies as legally cognizable contracts. When web user behavior does not conform to the language of a service provider contract that user’s bound themselves to, the user may no longer be allowed to use the product or service. In regards to our privacy tool, we analyze what would happen to a user who decides to use our privacy tool within the context of a notice and consent framework. We also assess whether the use of our tool would violate Google’s Terms of Service or Privacy Policy.

We begin the paper by discussing the function of our proposed privacy tool. We then discuss how we tested our browser to see if there is a statistically significant level of independence between advertisements received and browsing configuration used. After laying out the mechanics of our browser, we discuss the possible legal ramifications of fraud and breach of contract stemming from its use. Because we are analyzing two possible legal ramifications, the legal portion of the paper is split into two parts where the first part looks at fraud while the second part looks at breach of contract. For both parts, we lay out the current legal framework. Next, we discern whether “anything of value” relates to user data for the fraud portion and dig into the language of a privacy policy for the contract portion. Then we conclude.

# EXPERIMENT

## Background

Cross-site tracking refers to the practice of companies collecting user data across multiple websites. This practice is implemented on websites in a variety of ways including but not limited to scripts, widgets, embedded images within websites, buttons, and cookies. The data collection process will typically occur behind the scenes, without the general user's knowledge or even consent with companies using this to personalize ad experiences and develop user profiles. For example, on social media websites, “like” and “share” buttons can enable platforms to track a user’s behavior across numerous other social media accounts they are logged into and compile your browsing habits to a user-profile. Both stateful and stateless tracking techniques are used for cross-site tracking.

1. Stateful Tracking

Stateful tracking is the first major method of tracking used by companies to track user activity through information stored in data packets on the client’s side. The most commonly stored data packets are known as cookies. Cookiesare text files that store pieces of data, such as username and password, relevant to the user. These first-party cookies (cookies generated by the host website) are often used by websites to help uniquely remember web user. Cookies, however, have also been used by websites to track its users for other purposes.

Third-party cookiesare a type of cookie that are specifically created by websites that are not the one a user visits. Some examples of third-party cookies are those set by Facebook pixels, amazon-adsystem.com, and many other advertising networks. These cookies are able to store information about a user's browsing activity by collecting various pieces of information about what they have done on a website. Advertisers then use this information to deliver custom ad to users. For example, having a “Like” button on Facebook will store a cookie on a user’s computer which Facebook can later use to monitor the websites a user visits later. Over time these cookies will amass large amounts of data about user’s personal information and website behavior which can become quite intrusive into a user’s privacy.

HTTP requests, a request made by a client to a host on a web server to access a resource on the server, contain a User-Agent header field. A User-Agent header, also called the User Agent or UA string, is a string in every HTTP request that lets the server know the “application, operating system, vendor, and/or version information about the user agent” (“*User-agent - http: MDN”, n.d.)*. The User-Agent field became our primary concern for intercepting incoming HTTP requests with the stateful portion of our privacy tool. In HTTP requests, the User-Agent header fields follow this general format (“*User-agent - http: MDN”, n.d.)*:

*Mozilla/5.0 (<system-information>) <platform> (<platform-details>) <extensions>*

“Mozilla/5.0” in the header field above indicates that the web browser is compatible with Mozilla (“*User-agent - http: MDN”, n.d.)*. Although User-Agent headers do not provide specific information about the user’s browsing activity, its user specific hardware and software information allow companies to potentially handle web traffic differently depending on the type of UA string present in the HTTP request. The swapping mechanism used in our privacy tool to swap the user’s current UA string with a different one is similar to the concept of user-agent spoofing. However, while user-agent spoofing oftentimes includes nonsensical strings, our swapping of user-agent strings will consist of switching between User-Agent strings from a dataset of ten thousand valid headers representing various web browsers and devices. There are a series of vulnerabilities that accompany websites lacking valid User-Agent fields. For stateful tracking, User-Agent header fields are much easier to modify compared to cookies. Unlike cookie headers where modification can generally only occur through disabling cookies, User-Agent headers can be modified both through proxies and by individuals using their own web browser. While a low barrier to modification may sound nice at first, its simplicity also reveals points of attack. The main downside of the low barrier to modification is that because User-Agent headers are easy to modify, generally anyone could modify these fields. As a result, a low barrier to modification can pose a security threat to organizations whose web servers store User Agent values in their system. For example, bad faith actors could performcross-site scripting.Cross-Site Scripting (XSS) attacks refers to “a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites.” (S,K, n.d.). Websites that store User-Agent headers, but do not have a user validation system in place, are vulnerable to XSS attacks.To avoid exposing the vulnerabilities that accompany websites lacking valid User-Agent fields, our privacy tool only modifies User-Agent strings for valid devices.

Another piece of information present in every data packet that is used for tracking is the IP address, or Internet Protocol Address. The IP address is used to uniquely identify a device on the internet. These addresses appear as a string of numbers separated by periods, an example of an IP address is “192.168. 1.1”. ISPs (Internet Service Providers) such as Mediacom and CenturyLink assign these IP addresses to internet users. Whenever a client connects to a server, the website will see your IP address (as this is the way the server knows where to send data back to when “you” are communicating with them). IP Targeting is a method in which advertisers personalize advertisements to users based on their location, which can be determined based on the IP address of the user’s device.

1. Stateless Tracking

Stateless tracking,on the other hand, does not store identifiers directly on the user’s computer. Rather, information likely to identify a user is collected and processed by the website itself. This means there is no longer a direct mapping between a user and their consumer profile. Instead, stateless tracking relies on statistical inference to perform a “best guess” matching. While stateless tracking is a less accurate form of identification by definition, it has the advantage of being much harder to detect and is more robust against alteration by the user. JavaScript files contained in websites perform stateless tracking by leveraging the various APIs built into the user’s browser. An API, or application programming interface, is essentially just a way for different computer programs to communicate with each other. Our focus here is on the APIs that generate and transfer revealing information about a user to online trackers. One example isthe Canvas API, which uses the HTML5 canvas element to identify minute, production-related differences in graphical rendering, even between devices of the same model (Copland, n.d.). Another form of browser fingerprinting is media device fingerprinting, which reveals all connected media devices on the user’s computer device. These may include headphones, video cards, audio cards, etc. This technique requires webcam/microphone access, so it is more commonly used in streaming applications such as Zoom or Microsoft Teams. The Web Audio API outputs information that can track a user solely through nuances in how their audio devices produce sound. The Geolocation API is used by websites to access location data of its users, and is what we opted to leverage for the stateless portion of our privacy tool. Due to its high degree of specificity, this is an especially potent API for uniquely identifying a user. To optimize accuracy in generating a user’s browser fingerprint, it is often best to use a combination of the fingerprinting techniques described previously. This combined with the use of scripts to track cross-site activity enables companies to develop a browser fingerprint of its users in a stateless manner.

1. Research Questions

Through our research, we hope to gauge whether our efforts to incorporate fraudulent data into our browsing fingerprint will have a measurable impact on the advertisements we receive. Successfully doing so would open the door for future research into the set of features that play the most significant role in advertising patterns. Knowledge of these features could theoretically allow us to reverse-engineer advertisement algorithms such that we cause users of our tool to appear as “advertiser friendly” as possible while simultaneously giving away as little of their actual data as possible. However, in order to reach that point, we must first complete the groundwork of establishing feasibility. Due to the dichotomy of tracking techniques outlined above, we plan on tackling the two separately. Our hope is that this will allow us to ascertain the relative importance of each. We leave it to future work to implement a more combined approach.

## Experimental Setup

1. Browser Set

In preparation for a between-group analysis of the ads received using different browser configurations, our build of Chromium was split into three different branches. **BC** was equivalent to a vanilla build of Chromium and was used as our control. **BU** utilized the same binary as the control but incorporated a man-in-the-middle proxy to intercept outgoing request packets and change their ‘User-Agent’ headers. Finally, **BG** was a separately-compiled binary which enabled the randomization of geolocation coordinates. These browsers were set up such that we were able to switch between them at will.

1. Website Set

In order to conduct our experiment, we first needed to collect a large set of websites that would be receptive to the signals we would be leaking with our tools. Because a significant portion of the implementation was based on spoofing the coordinates returned by the Geolocation API, the only way to truly measure its effectiveness would be to crawl sites that make use of the user’s location in some way or another. For this, we instrumented *OpenWPM*, an open-source web privacy measurement framework that is built off of a popular browser automation library known as *Selenium*. *OpenWPM* provides a distinct advantage over pure *Selenium* in that it can detect calls to the JavaScript APIs built into the browser. Using this framework, we performed a crawl through *Tranco*, a research-oriented top website list, keeping track of all sites whose scripts made calls to Geolocation API functions. This gave us a list of approximately 400 websites.

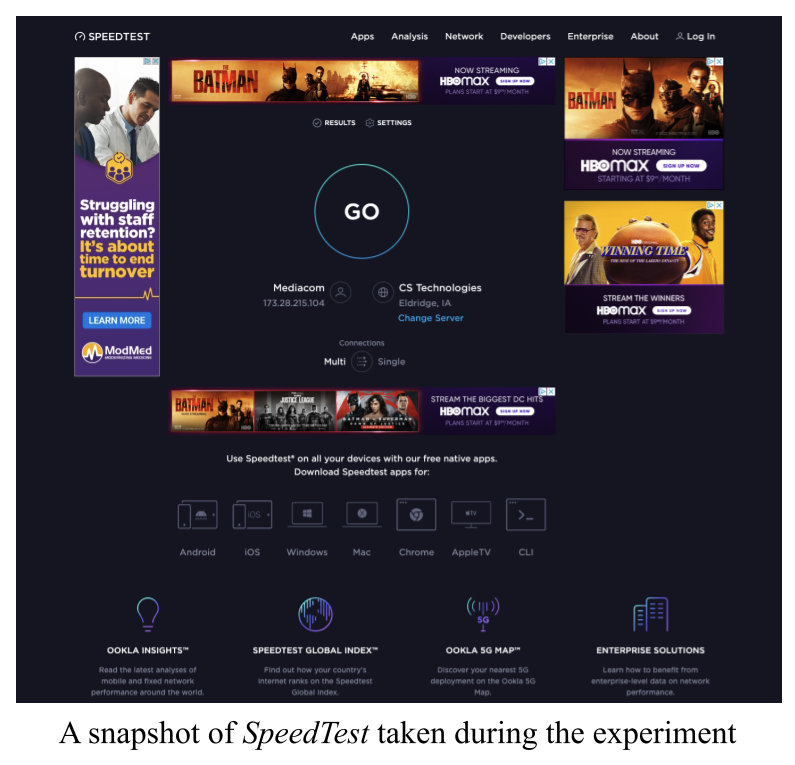
Next, we needed to identify a subset of the aforementioned websites that could be associated with a specific product or service. We felt that focusing on too broad of a category (e.g. news, online shopping, or technology) could lead to difficulty in building an effective consumer profile due to the existence of many sub-categories that are advertisable in their own right. In the end, we chose to go with travel-related websites. We went through our list and picked out sites we felt were most closely related to travel: flight-booking, hotels, long-distance transportation, and travel agencies. After filtering out websites that caused crashes and other unexpected behavior in our automated browser, we ended with a set of 21 sites including *Expedia*, *ChoiceHotels*, and *Delta*.

Finally, we needed to select a website for our browser to continually return to in order to measure changes in the advertisements received. An ideal measurement website would feature advertisements that are numerous, easy to collect, and are relatively indicative of the quality of advertisements a user is likely to encounter during normal browsing (meaning no advertisements for scams or not-safe-for-work services). Additionally, this website would have to perform well when loaded by an automated instance of our browser. This proved to be no easy task.

Originally, we were planning to use either *CNN* or *ESPN*, as these are good representations of a typical advertisement-rich ecosystem. However, we had to change our plans for two main reasons: First, the plethora of scripts and trackers present on these sites meant that crashes and slow load times were not uncommon. Second, and more importantly, these websites began to advertise upcoming specials shortly before the experiment began, which meant that whatever advertisements we saw would have nothing to do with our prior browsing activity.

We then pivoted to *DailyMail*, which initially shocked us with the amount of ads it contained. Unfortunately, this also had the effect of making the site nearly impossible for our browser to load without crashing or timing out, even when given several minutes. In the end, the website we selected for measurement was *SpeedTest*. At the time, this website featured five large advertising banners in immediate view when opening the site, in addition to playing well with our automated browser during initial testing. While the collected ads were often networking- or computer-related, the lack of a clear alternative and the site’s apparent responsiveness to outside browsing activity meant that was our best option.

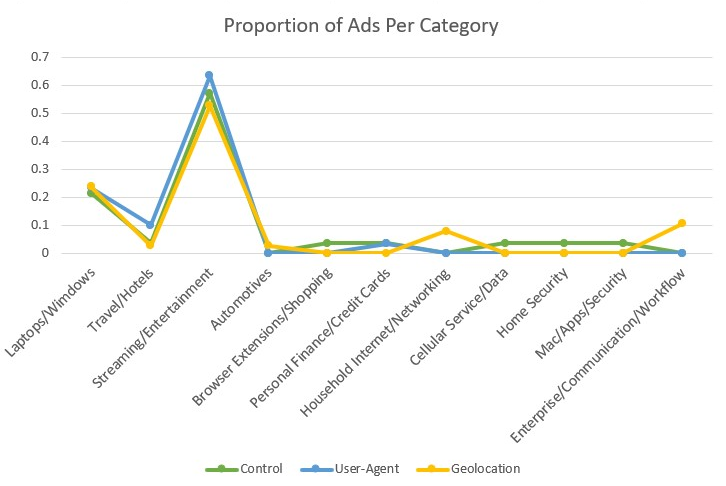
## Methodology

The experiment was conducted as follows. For each of our three browser configurations **BC**, **BU**,and **BG**, our automation tool was first instructed to traverse our list of travel-related websites in order. No actions were taken on these websites other than loading their URLs and pausing for several seconds to allow any background scripts on the page to finish running. After completing a run through the list, the browser would visit our measurement site *SpeedTest.net*, wait for the page to load, then take a screenshot of the browser window such that all ads on the page are visible, and add it to our dataset. This process of iterating through the website list and capturing the resulting measurement page was repeated ten times for each browser configuration, leaving us with 30 snapshots in total, sorted into three 10-snapshot datasets represented by **DC**, **DU**, and **DG**.

Once the datasets were filled out, all of the advertisements within each snapshot were manually extracted using an area-screenshot tool. Originally, we intended to programmatically extract advertisements from the measurement site by checking all image content request packets for URLs matching that of known advertisers, then storing the data with the body of the corresponding response packet. This approach yielded extremely unreliable results, which we will explore in detail within the Development & Limitations section.

After the advertisement images were obtained, they were then sorted into categories. To do so, we came up with several terms describing each advertisement that was unique with respect for content. Different shapes of the same ad were not counted as different. These terms were determined based on a combination of features including the image’s content, labels assigned by Google’s Vision AI, and abstract descriptors relating to the type of product being advertised. For example, we differentiated between advertisements for enterprise software and individual-use software. Similarly, we drew distinctions between household internet service and cellular carrier service, even if both involved wireless networking to some extent.

And then we did the chi square to see if ads per category was independent of

browser used. To perform a chi square test, we needed to first determine Observed/Expected values for the dataset. Observed values are the number of ads that are displayed in each category (in our dataset it was also necessary to compute row and column sums for these values). Using the computed row/column sums and total sum in the Observed dataset, we used the equation, “(Row Sum - Column Sum)/ Total Sum” to find the Expected value for each cell in the dataset (keep in mind this dataset’s size is the same as the Observed one).Next, we use the equation “ [Observe - Expected]^2 / Expected” for each data entry from the previous datasets and sum all of the computed values to get our chi square value. This chi-square value alone doesn’t say much about the relationship in the variables, so using degrees of freedom (found by taking the product of the number of rows - 1 and the number of columns - 1). Finally, we use the Excel function “CHI.DISTR.RT[chi-square value, degrees of freedom]” (or alternatively an online chi square probability table) to find the p-value for the datasets. Using this p-value, one can determine whether to accept the null hypothesis (p > 0.05) or reject the null hypothesis (p < 0.05).

## Results

Upon completing our three trials of experimentation for the privacy tool and performing a chi-square test of independence on the dataset, our results showed that there was no relationship between ads shown and the type of browser used.

A chi-square test of independence is a statistical test used to determine how likely a relationship is to exist between two variables. In addition to the chi-squared value being necessary to interpreting our results we also needed the degrees of freedom (maximum number of values that can vary without affecting the system) value. With both of these numbers we were able to find the p-value, which is what is used to determine whether the null hypothesis is rejected or not. The null hypothesis refers to the hypothesis that there exists no statistical significance between given data sets. Based on a p-value of 0.026 and a significance level of 0.05, we were unable to reject the null hypothesis. As shown in the above graph, the proportion of ads received in each category is fairly consistent across each browser configuration (Control, User Agent and Geolocation) which holds true to the chi square test of independence.

## Development & Limitations

Throughout the development of our privacy tool and experimental tools, there were numerous instances in which our approach had to be altered or abandoned entirely. By the end, we conducted three full-length experiments using varying combinations of automation and measurement configuration. The datasets from each of these experiments are contained in folders labeled *data1*, *data2*, and *data3* within our *browser-automator* repository. What we refer to as the “automation tool” and “measurement tool” are, in their most recent iteration, two parts of the same Python script. Due to their historical implementation as isolated components, however, we will consider them each individually. Additionally, the way in which some trials were conducted may have had confounding effects on our results. The most significant obstacles faced, as well as our attempts to overcome them, are outlined in this section.

1. Stateful Component

Originally, the plan was for the stateful component of our tool to manipulate cookies as they are being retrieved by the website. We had hoped that by altering the contents of the ‘Cookie’ header of a packet, we could force websites to collect garbage data and thus frustrate attempts to use them for tracking. The initial way in which this was performed was using pattern matching to detect every instance of a *key=value* pair within the ‘Cookie’ header and randomizing the string within the *value* section. Our assumption was that at worst, this would have a similar impact on browsing experience as constantly deleting cookies.

What we found, however, is that this approach tended to break websites. Domains such as *Target* and *Amazon* were not expected to maintain shopping cart functionality without cookies, but what we observed was the failure to even load product pages in the first place. Shopping websites aside, we found a general pattern of unexpected behavior arising as a result of replacing cookies with nonsense strings. Combined with the fact that a principled approach to altering cookie data would not be possible without server-side knowledge, it seemed that a cookie-based implementation would simply not be feasible.

This observation led us to adopting the ‘User-Agent’ header as the core of our stateful implementation. As we discovered, however, this would come with issues of its own. For one, these strings would have to be altered in such a way that the end result was still coherent. Our first implementation randomized sub-components of the User-Agent string such that all possible combinations of strings were seen as valid. For example, a nonsense system information string of “*(Linux; Windows NT 10.0; Intel Mac OS X 12.3)*” was considered equal to one which properly conveys a single operating system and CPU architecture. Further research into the purposes of User-Agent strings led us to conclude that this approach would be infeasible due to its propensity to break sites which need to parse the User-Agent string to determine compatibility.

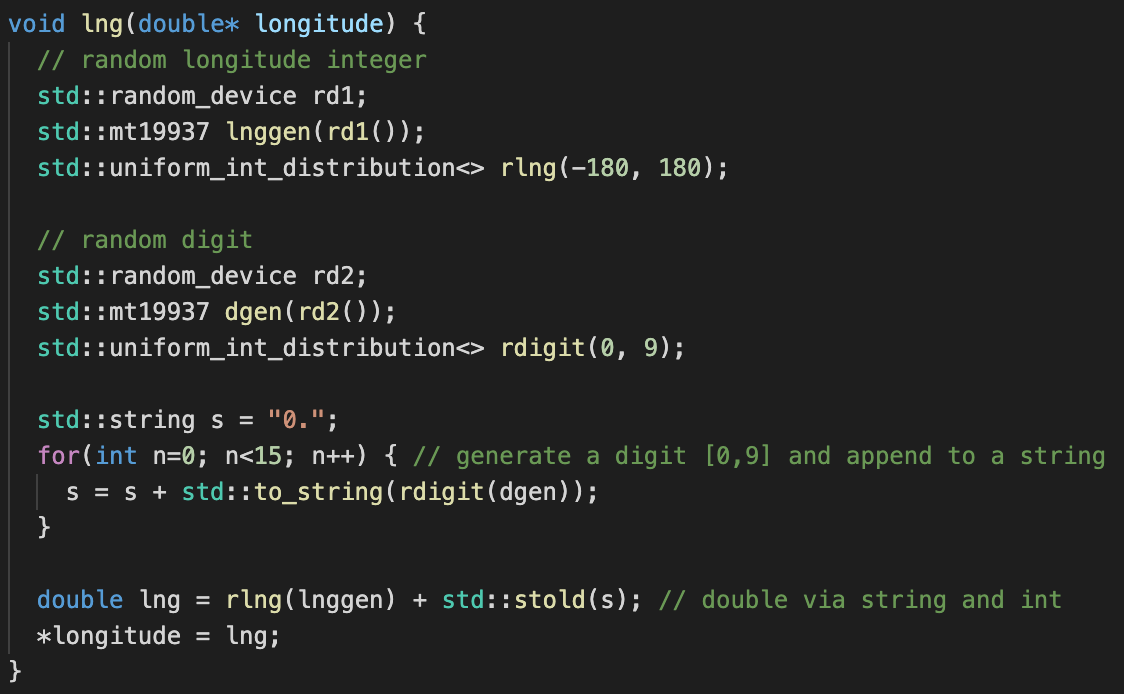
In lieu of devoting significant time to developing a User-Agent generator that never compromises on validity, we opted to randomly select entire strings from an open-source collection of approximately 10,000 User-Agents. We noticed that some of these strings would often give Google a different appearance and occasionally load mobile versions of sites, but it was not until the experiment was conducted that we saw instances of websites refusing to load at all on what they thought to be an outdated browser. Inconsistencies in the ability of our browser to successfully leak a signal, even if uncommon, can certainly be seen as adding confounding factors to our data. It is clear that more work should have gone into the stateless component in general, as well as more time spent testing its functionality.

1. Stateless Component

We chose the Geolocation API as the core of our stateless component due to its high potential to uniquely identify a user. Our initial assumption was that once a single trackable identifier had been successfully spoofed, we could easily pivot toward countering some of the other fingerprinting techniques mentioned in the Background section. However, changing the coordinates returned by the Geolocation API’s *getCurrentPosition* function took significantly longer than expected due to the overall complexity of the source code and a lack of information online regarding it. This meant that we were unable to meaningfully address other aspects of the stateless fingerprint. In addition, the disproportionate reliance on the Geolocation API meant that our experiment was sharply limited by the requirement of only browsing websites that utilize it. We further outline the issues that arose due to this in the Site Collection subsection.

In writing the coordinate-altering implementation, our first approach involved writing a custom HTML page that would display a visitor’s coordinates after being loaded. With the help of a debugger, execution of the browser’s code could be paused whenever the aptly-named *getCurrentPosition* function was called, allowing us to step through each individual instruction before it was executed by our CPU. At face value, the concept of noting which parts of the code were executed immediately before the page received coordinate information from the browser is trivial. In practice, however, the call to *getCurrentPosition* preceded coordinate generation by what we estimate to be thousands of instructions. Rather than spend hours manually stepping through instructions until coordinates appear, we chose to look for another approach.

Eventually, we shifted our focus toward the interface between the browser itself and the scripts contained in webpages. Because browser APIs are written in C++ but used by JavaScript code, there must be a point at which the data generated by the former is passed into the latter. We noticed that when a script receives geolocation information, it is delivered in the form of a *GeolocationCoordinate* object. This object contains, among other things, information such as latitude, longitude, altitude, and accuracy. If we managed to find the C++ representation of this object, we could then make alterations to its data before it is sent over. As it turned out, such a representation did exist: a component of the Geolocation source code named *GeolocationCoordinate*. We make no further acknowledgement regarding the irony that dozens of hours could have been saved with a bit more surface-level thinking.

Once we found this entry point, the only challenging part became wrestling with the low-level random number generation afforded by C++. Existing libraries such as *mt19937* and *random\_device* are good at generating decimal values of unspecified length. However, this would not be sufficient as we observed that Geolocation coordinates typically had lengthy fractional components of up to 15 digits. To create convincing values, we would need to reliably generate digit strings of said length in addition to an integer component whose possible values were dependent on whether the number represented latitude or longitude. The simplest approach we came up with involved starting with a character string of “0.” and sequentially appending string representations of singular digits generated by a uniform distribution. Once the fractional component string was generated and converted back into a double, it was then summed with a random integer [-90,90] or [-180, 180]. Upon further examination, it would have been prudent to generate integer components using non-inclusive bounds due to the possibility for the separately-generated fractional component to push coordinate values past their normally-allowed boundaries. Additionally, no checks were implemented to ensure that only realistic coordinates were generated; locations such as the middle of the Atlantic ocean and the Antarctic Circle were not uncommon.

The greatest shortcomings of this component can ultimately be traced back to time constraints, which forced an overly-narrow scope with respect to alterations made to the user’s stateless identity. It is likely that more significant results could have been obtained through a more efficient approach to inserting hooks into browser APIs. Specifically, if less time had been spent attempting to understand the complex inner workings of the Geolocation codebase and more focus was directed towards the browser-to-JavaScript interface, a greater degree of breadth in implementation could have been accomplished.

1. Chromium

Making use of an open-source Chromium build allowed unprecedented control over browser behaviors typically abstracted away from the user. As with many low-level tools, however, there is also far more potential for unexpected behavior. Throughout our research, it was a common occurrence to trigger full rebuilds of the roughly 50,000 build targets that make up the Chromium binary. Excluding the intentional clean builds used to create our set of browser configurations, we estimate that Chromium was recompiled in its entirety at least six times during development, a process which takes multiple hours. To be sure, these rebuilds were always caused by some form of user error. Google’s repository tool *gclient* and build tool *Ninja* are robust in that they never allow inconsistencies in dependencies.

One particularly memorable rebuild involved moving the build directory in its entirety to a new location due to iCloud attempting to sync the 60GB project, which had had the effect of massively slowing filesystem operations. Even when using a shell script to replace every instance of the old file path in every file (including within binaries, where paths would still appear as plain text), a rebuild was unavoidable. It was not long after this build was completed that we discovered Google’s ChromeDriver, the tool that enables browser automation, was incompatible with the latest build of Chromium. Another multi-hour rebuild was required to downgrade Chromium to an acceptable version. Fortunately, these full rebuilds were infrequent enough to allow for consistent progress to be made. Changing a single file would only require seconds to regenerate the binary.

Compilation issues aside, perhaps the largest deficit associated with an “un-Googled” Chromium build was the ever-present issue of instability. A clean build using all of the recommended options could be made to consistently suffer crashes when visiting certain sites. Ironically, the Google Cloud Console was one such site. This page was slow to the point of being nearly unusable before inevitably resulting in a crash. The crash reports generated by such occurrences were difficult to interpret and did not lead to helpful search results online. This also characterizes another issue we faced with our Chromium development: a lack of external resources.

Our impression is that making a custom build of Chromium is not a common endeavor outside of enterprise environments. While Google provides excellent documentation regarding how to build or downgrade Chromium and a powerful code-search tool, there is only so much that can be answered by high-level guides when working with such a complex application. The *chromium-dev* group gave us access to a community of people in similar positions, but we were not able to derive a significant amount of help from it. This can be partly attributed to the fact that the obstacles we encountered were of such specificity that posing effective questions regarding them was quite challenging. For example, no amount of *chromium-dev* posts or internet searches seemed to remedy an issue we faced where the API key obtained from Google Cloud was not allowing Geolocation requests to be completed. Google’s documentation regarding this process mentioned the requirement of an enabled billing account in order to use Geolocation services, and we made sure to do so. It was not until we attempted to contact Google Cloud support that an automated banner popped up asking us to reconnect our billing account to the project.

Despite the myriad difficulties we encountered when working with our fork of Chromium, we found it to be an extremely rewarding experience overall. The sheer capability afforded to us by full access to the source code and Google’s Chromium Code Search tool, which provides descriptions for most directories within the file tree, represents the opportunity to address web privacy vulnerabilities at an unprecedented level. Our goal was to take noise-based approaches like Brave Browser’s “farbling” a step further by implementing complete informational detachment, rather than simply lowering the granularity of data given to advertisers. Given a longer development period and the hindsight gained from working on this project, we believe a fully-fledged implementation to be a definite possibility.

1. Site Collection

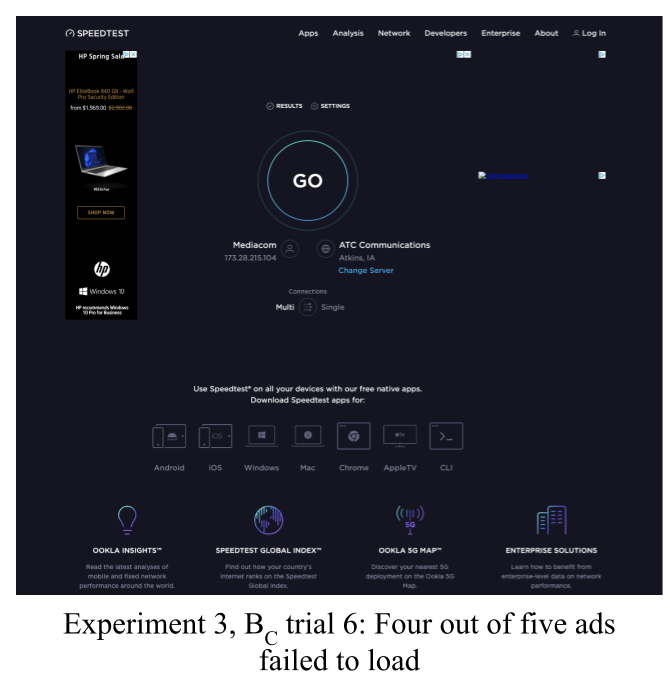
At the beginning, our plan was to use *OpenWPM* to crawl one of the top website lists and obtain a set of sites that would be receptive to our coordinate and user-agent signal leaks. However, time pressures lead us to the conclusion that it would be more prudent to incorporate the work of another group that was working on a similar project. From them, we were able to obtain a list of approximately 400 websites containing scripts that made calls to the Geolocation API. It was not until we began testing our automation tool on said list that we observed the same nearby airport (CID) being suggested regardless of whether or not we were using the coordinate-randomizing build. This suggested that the false location was not being correctly received. Closer examination revealed that while the *geolocation* function was being called on all sites in the test set, the *getCurrentPosition* function, which returns coordinate values wrapped in a callback function, was not.

Considering the main functionality of the Geolocation API derives from *getCurrentPosition*, this was not an expected observation. From what we could tell, the websites in our test set tended to call the *geolocation* function twice: once to generate something referred to in the code as a “supplement”, and again to confirm that this object had been successfully created. A comment within the file containing the *Supplement* class suggests that the purpose of these objects is to allow garbage-collected objects to be extended with additional data. What sort of data is being added to the “Navigator” parameter of the *geolocation* function remains unclear. Further examination would likely yield a more satisfying answer, but for now our best guess is that the supplement contains data relevant to the context in which the user’s location is utilized. We theorize that our approximate location is being obtained entirely through our IP address.

In an ideal scenario, we would have carved out time specifically to collect websites for the test set ourselves. However, unforeseen difficulties with respect to the implementation of our automation tool and especially with respect to our ad collection tool convinced us to adopt the most accessible alternative. The question of whether coordinate randomization across websites that make direct use of said coordinates is truly enough to impact one’s browsing fingerprint remains unanswered.

1. Automation Tool

The development of the *Selenium*-based automation tool we used to run the experiment was relatively straightforward despite some dependency issues that forced us to switch from the Java version of the library to a Python-based implementation. Using *Selenium*’s simple interface, we were able to attach the Google-provided ChromeDriver to our desired build of Chromium and have it follow simple instructions such as fetching a specific website, taking a screenshot, or pausing to allow advertisements to load. Additionally, we were pleased to find a modified version of the library known as *Selenium Wire* (though we will continue to refer to “*Selenium”* for the sake of simplicity), which boasts proxy features built directly into the tool without requiring any additional networking configuration on our end. This allowed us to automate processes such as the modification of packet information with far greater efficiency compared to our original proxy application, *Burp Suite*.

This portion of development was not without hiccups of its own, however. For one, we found the issue of instability to be significantly worsened when running our browsers in an automated manner. In addition to the occasional crashes of Chromium itself, execution of our *Selenium* code would constantly encounter runtime errors whenever a page failed to load completely. Increasing the duration of the timeout period had no effect as *Selenium* seemed to sometimes cause our browser to suffer endless load times where it had not otherwise. While this may be partly attributed to the incompatible User-Agent strings mentioned previously, we observed similar issues across all configurations in our browser set. Given that deadlines were rapidly approaching and each experiment took approximately five hours to complete, it was often all we could do to modify our automation code to skip problematic websites while running certain browser configurations. This can undoubtedly be viewed as adding a negligible, but still existent, confounding factor to our data.

A more significant issue with our data lies in the inconsistency with which advertisements are properly loaded on the measurement site when using *Selenium*. In our third and final experiment, there were numerous instances in which advertisements would simply not appear despite still being clickable on the page itself. This persisted across all three browser configurations and did not seem to be remedyable even when allowing five minutes for the measurement page to load. Due to the way we were forced to measure advertisements in the end, this has serious implications with respect to the legitimacy of our data. We will explore our difficulties in measuring ads in greater detail in the next subsection.

During our second experiment, we crawled the test set with our browsers in “headless” mode, an option which removes the user-facing application window in order to run in the background. This allowed us to capture full-page screenshots of our measurement site, consistently load advertisements, avoid all of the intermittent crashes, and run through our whole experiment in less than half the usual time. As it turned out, however, this sharp increase in performance was less of a blessing than we had originally thought. The actual reason our headless browsers were able to run so much faster was because the pages weren’t being loaded at all–most of the websites on our list employed bot-detection functionality, which could easily detect our headless application and serve us simple “Access Denied” pages. Additionally, without a visually-rendered page, we would have never realized this was occurring had it not been for the observations of another group working on a similar project. This at least provided an explanation for why the ads in the second dataset were so suspiciously contextual to *SpeedTest.net*.

Performance issues notwithstanding, developing the tool to automate our experiment was easily one of the more successful aspects of our research. Though lacking in documentation, we found the *Selenium* library to be intuitive enough that we had no problem learning how to instrument it to suit our needs. In addition, we were pleased to discover the existence of a modified version which significantly reduced the overall complexity of our tool. The main obstacles we encountered when automating our experiment can be attributed to ad collection, which will be outlined in the following section.

1. Measurement Tool

The collection and measurement of advertisements was without question the most challenging aspect of our implementation. Our approach was altered and replaced time and time again throughout development, only to eventually be abandoned entirely in favor of manual ad collection. We encountered issues in nearly every aspect imaginable from loading, to detecting, to filtering, to extracting these advertisements from a page.

Our first approach involved a browser extension which, when activated, would save the URLs of all images contained in a page’s HTML body. These URLs, once written to a text file, would then be handed to a Python script that implements the *AdblockParser* library. Through this, we would be able to create a ruleset based on the filter list of our choosing (we used the publicly-available *EasyList*). Any time a URL in the text file matched one of the patterns contained in this ruleset, we would then attempt to fetch the URL with *urllib* and save its content as an image. However, what we observed was that few, if any, of the advertisements were being saved. The issue ended up being that most modern advertisements are not added directly to the page’s body, but rather contained in what is referred to as a “frame” which essentially functions as a page-within-a-page. We had heard of a recursive strategy which can circumvent this shortfall, but due to the number of points of failure in this implementation, we decided to look for a simpler option.

Our next approach ended up not being all that much simpler. We hoped to leverage our proxy capabilities to collect image URLs directly from request packets. Using a *Burp Suite* extension written in Java, the URLs from every outgoing packet were intercepted and fetched as a buffered image. This had the added benefit of allowing us to filter images by file size to prevent tracking pixels from getting in the way. Each time an image was successfully obtained, the URL would be written to a text file. As before, this text file could then be fed into our *AdblockParser* Python file. Overhead issues from re-fetching every URL out of paranoia aside, glimpses into the generated text files made clear that we were still losing a significant portion of ads even before URL filtering was performed. Our understanding is that this is likely caused by *Burp Suite*’s inability to intercept packets written in protocols other than HTTP.

In an attempt to circumvent this, we looked to the network performance diagnostics functionality built directly into Chromium’s developer console, DevTools. With DevTools, we can track every request and response that passes through our browser and save them to a HTTP Archive (HAR) file. It would then be a trivial process to parse through this file and extract the binary data of every image contained in a response packet. In order to get to this point, however, we would need a way to automate the whole process.

At this point in the project we were still working with the Java version of *Selenium* and fortunately, there seemed to be a simple way to generate HAR files with it. The most common way of doing so involved the use of the *BrowserMob Proxy* library. However, despite our best effort, we were never able to get this proxy to work correctly. When running the proxy in standalone mode, no amount of device configuration seemed to allow us to receive network traffic. Similarly, importing the library into our code for use with *Selenium* gave us persistent dependency issues that not even a project management tool like *Maven* could solve. We surmise this to be an issue with the library itself given the fact that both the *Burp Suite* and *Selenium Wire* proxies worked out of the box with no additional configuration on our part. Without going into too much additional detail, we also attempted to use *Selenium*’s built-in DevTools interface without success. Similarly, *Selenium Wire*’s HAR generation functions did not yield any useful data either.

While the HAR functions themselves were of little use, *Selenium Wire* still gave us access to a powerful proxy interface. With it, we could save all requests generated by a visit to our measurement site. Since these are the same requests that would have been used to build the HAR file, we saw little drawback to this approach. Our process was as follows: for every request that had received a corresponding response, check its URL against our *AdblockParser* filter list. If the URL was designated as belonging to an ad, we would then examine the “Content-Type” header of the response packet and confirm that it was indeed an image. Finally, we would extract the encoded binary from the response packet’s body and, if it met our size requirement of 10KB, save it to an image file of the correct format. All of this was able to happen in tandem with the rest of our automation tool thanks to *Selenium Wire*’s capabilities.

Unfortunately, even with this low-level approach, we were unable to consistently extract all or even most advertisements from the measurement page. The dataset generated by the first experiment (located in the *data1* directory) shows this clearly; many of the images we obtained with this approach were, for example, nothing more than the blurry background of an ad frame. In the end, we realized the only reliable way to extract ads was to do so manually. We regret being unable to find a better alternative. Perhaps with a better understanding of image content delivery and more time, we would be able to deliver a proper implementation in this regard.

# LEGAL ANALYSIS

## Computer Fraud and Abuse Act

## The Computer Fraud and Abuse Act (CFAA) subjects to criminal and civil liability anyone who, “knowingly and with intent to defraud, accesses a protected computer without authorization, or exceeds authorized access, and by means of such conduct furthers the intended fraud and obtains anything of value, unless the object of the fraud and the thing obtained consists only of the use of the computer and the value of such use is not more than $5,000 in any 1-year period.”[[27]](#footnote-26) This means that to prove a claim under the civil remedy provision of the CFAA, the plaintiff must show that the defendant:

## (1) Intentionally accessed a computer

## (2) Without authorization or exceeding authorized access, and

## (3) thereby obtained information

## (4) from any protected computer, if the conduct involved an interstate or foreign communication, and that

## (5) there was loss to one or more persons during any one-year period aggregating at least $5,000 in value.

## Defining the Elements

## In order to determine if our privacy tool falls under the CFAA, we would need to further define each element.

1. *Accessing a Computer*

The courts define a computer as a device that is "an electronic … or other high speed data processing device performing logical, arithmetic, or storage functions."[[28]](#footnote-27) This definition includes any device that makes use of an electronic data processor. " Therefore, common household items that include microchips and electronic storage devices will satisfy the statutory definition of 'computer.' However, the term computer "does not include an automated typewriter or typesetter, a portable hand held calculator, or other similar device."[[29]](#footnote-28)

## *Without Authorization or Exceeding Authorization*

The phrases "accesses a computer without authorization" and "exceeds authorized access" cannot be read to include an individual's misuse or misappropriation of information to which the individual was permitted access.[[30]](#footnote-29) What use an individual makes of the accessed information is completely distinct from whether the access was authorized in the first place.[[31]](#footnote-30) Therefore, a theory that, “... the CFAA is violated whenever an individual uses information on a computer in a manner contrary to the information owner's interest would therefore require a departure from the plain meaning of the statutory text.”[[32]](#footnote-31)

The Ninth Circuit and district courts in the second circuits have held that an employee with authority to access his employer's computer system does not violate the CFAA by using his access privileges to misappropriate information.[[33]](#footnote-32) Therefore, unless an individual lacks authorization to access a computer system, or exceeds the authorization that has been granted, there can be no violation of § 1030(a)(2)(C).[[34]](#footnote-33) In Alenkyov, the defendant was authorized to access the Trading System’s source code. The defendant misappropriated the source code, but the court ruled that misappropriating the source code did not violate § 1030(a)(2) (C). Therefore, if the defendant is authorized to access the protected computers, accessing the computers for an improper purpose does violate the CFAA.[[35]](#footnote-34)

There are two main rules in analyzing authorization under the CFAA. First, a defendant can violate the CFAA when he or she has no permission to access a computer or when such permission has been revoked explicitly. Once permission has been revoked, the enlisting of a third party to aid in access will not excuse liability.[[36]](#footnote-35) Second, simply a violation of the terms of use of a website—without more—cannot establish liability under the CFAA.[[37]](#footnote-36) In neither case does the purpose of the accessor determine the level of authorization, and a company's terms of service cannot be used to make an otherwise-authorized access unlawful.[[38]](#footnote-37) Furthermore, if an employee is authorized to access confidential information, later misappropriation does create CFAA liability. “These courts recognize that the plain language of the CFAA ‘target[s] the unauthorized procurement or alteration of information, not its misuse or misappropriation.’”[[39]](#footnote-38)

1. *Thereby Obtained Information*

LVRC Holdings LLC v. Brekka defines that actions to include not only obtaining information, but also damaging a computer or computer data. This is illustrated in the case P.C. Yonkers, Inc. v. Celebrations the Party and Seasonal Superstore, LLC. The plaintiff’s primary claim was that the defendant without authorization and on behalf of defendant Celebrations and defendant Bailen,” accessed PCC’s Tomax computer system 125 times over seven days during October and November of 2003. Eight of the alleged incursions occurred after Hack ceased working as a consultant to PC Management. The plaintiffs claim that unauthorized access purportedly was gained again in December 2003 and a final time in April 2004, when the defendant was no longer associated with any of the PC plaintiffs. The access in December 2003 lasted a total of 19.4 minutes. The defendant testified that he had a home office during his years with PCC and had been authorized to use his computer from home. As proof, he offered emails demonstrating that he did so. He stated that he imagined it would have been for PC Management business as he never accessed the Tomax system for anything but PC Management related work. The PC plaintiffs contested the defendant's asserted authorization in their submissions. “There is a paucity of information as to precisely what could have been obtained from the system in these incursions, although the PC plaintiffs' computer consultant, Joseph Savin, stated that ‘reports’ could be ordered in a matter of seconds and then ‘later, with a few keystrokes,’ downloaded and sent to a remote location.”

The PC plaintiffs claimed that the defendant used the information obtained from this access to decide where to locate their stores, where to focus marketing efforts and budgets, and to obtain valuable information as to sales during the Halloween season. They urge that by using this valuable information, the defendant purportedly obtained an unfair competitive advantage. The PC plaintiffs specifically averred that defendants' unauthorized access resulted in damage or loss to the PC plaintiffs of not less than $5,000 within the meaning of CFAA, 18 U.S.C. § 1030. The court stated:

“... the PC plaintiffs urge that we draw inferences of intent and the obtaining of valuable information from the mere fact that unauthorized access has been shown, and ask defendants to rebut these inferences by demonstrating the innocence of their purpose or actions. However, the elements of the claims asserted are part of a plaintiff's burden. That information was taken does not flow logically from mere access. Access could be accidental, and, even if access were purposeful and unauthorized, information could be viewed but not used or taken. Furthermore, without a showing of some taking, or use, of information, it is difficult to prove intent to defraud, and indeed, the PC plaintiffs have not shown that they can do so.”

Therefore, in order for the defendant to have “obtained information” or “damage data,” there needs to be more than mere access to the data.

1. Legal Analysis
2. *Intentionally accessed a computer*

In order to change the data that ad companies would receive through stateless or stateful tracking, our privacy tool would be intentionally accessing a computer. Because courts state that the definition of a computer includes any device that makes use of an electronic data processor, every device that our privacy tool could be used on would fall under the category of a computer, including phones and tablets.

The privacy tool is a browser, and any user would need to intentionally access a computer when installing or using the browser. Whether or not a user intentionally accessed a computer by using our privacy tool may change depending on the circumstance. For example, if a user is unaware they are using a browser that would change the data that is being sent to the ad companies, and only uses it thinking it is a regular program, they may be able to argue they did not intentionally access the browser. However, they would still have intentionally accessed the computer, tablet, or phone that the browser is on, so the element may still be satisfied even if they did not intentionally access the browser.

1. *Without authorization or exceeding authorized access*

Our privacy tool would not have gotten authorization from any browser or any ad companies, however, it is accessing the user’s own data and altering it, so it could fall under the proper authorization assuming that the packets that the tool is changing the data belong to each user. If the packets do belong to each user, then accessing and altering each user's data that the packet would send would not satisfy this element. However, if the packet belongs to the browser and not each individual, then our tool would be violating this element. Even though the tool may be changing our own data, it is accessing and altering information on packets that belong to the browser. Whether or not this element is satisfied depends heavily on who owns the packets.

1. *Thereby Obtained Information From a Protected Computer*

If a court were to rule that our privacy tool satisfied the elements of accessing a computer without authorization, then the next step would be to determine whether the tool obtains information from any protected computer. The CFAA contains various subsections that makes the law applicable to deleting and accessing computer files without authorization. While our privacy does not directly delete files of a protected computer, it still accesses the data and reconfigures it. The question is whether reconfiguring the data is a form of deleting it. Our tool takes the data, and through the program, it changes the information on the packets, giving the ad companies false information. While this may appear as deleting the original data and replacing it with new data, the original data is still on the computer, only the ad companies are seeing the false data, therefore the privacy tool does not delete any data. Since the original data will still be there, the tool does not delete or “damage” the data as courts require. Ultimately, even if courts were to assume that the data belongs to ad companies and not users, it is very likely that this element would still not be found to be satisfied because the original data still exists.

1. *Loss in value*

Our privacy tool would cause a form of loss to the ad companies. The question is whether the loss would have a monetary value, and if it does, whether it would aggregate to $5,000 over any one year period. This would have multiple variables. First, it would depend on how the ad companies would measure the loss. If it would be based on the loss of money the ad companies pay to browsers for the user’s data, then it would depend on whether ad companies can combine their losses, instead of one single ad company. However, it is possible that the data has no monetary value to begin with, even though ad companies may lose money from not getting the data, if courts rule that each user’s data does not have monetary value, then this element would not be satisfied.

Ultimately, it is likely the our privacy tool does not violate the CFAA, however, it could be possible that a court rules that it does violate the CFAA if the court deems that the ad companies own the packets, the tool legally “deletes” the data through its function via unauthorized access, and the ad companies suffered a loss of more than $5,000 dollars in a year.

**Mail and Wire Fraud**

1. Background

While the CFAA applies to various forms of computer crimes, Wire fraud applies on a broader scale to acts that take place over a wire. Since our privacy tool revolves around wire transactions, we must analyze it under wire fraud as well. A person commits wire fraud when he or she:

1. devises or intends to devise a scheme or artifice to defraud or to obtain money or property by means of false pretenses, representations, or promises; and
2. for the purpose of executing the scheme or artifice, transmits or causes to be transmitted any writings, signals, pictures, sounds, or electronic or electric impulses by means of wire, radio, or television communications:
   1. from within this State; or
   2. so that the transmission is received by a person within this State; or
   3. so that the transmission may be accessed by a person within this State.

While fraud is generally hard to define, the Supreme Court of the United States defines it as “to wrong someone in his property rights through dishonest means and schemes”

1. *Defining the element “in furtherance of a scheme to defraud”*

In order for the scheme to defraud elements to be satisfied in mail and wire fraud, the mailing or wire needs to be in furtherance of a scheme to defraud, but the mail or wire itself does not need to contain false information. The wire or mail also does not need to be an essential element, just a step in the scheme. This is illustrated by the case *Schmuck v. U.S*., where the defendant purchased cars, rolled back the odometers, and then sold the car to dealers for inflated prices. The automobile dealers who were not a part of the scheme, mailed title applications to resell the cars. The court held that the mailing was in furtherance of a scheme to defraud. The court stated: “The mailings at issue satisfy the mailing element of the crime of mail fraud. Such mailings need not, as petitioner contends, be an essential element of the scheme to defraud, but are sufficient so long as they are incident to an essential part of the scheme. Here, although the mailings may not have contributed directly to the duping of either the retail dealers or the customers, they were necessary to the successful passage of title to the cars, which in turn was essential to the perpetuation of the scheme to defraud, since a failure in the passage of title would have jeopardized petitioner's relationship of trust and goodwill with the dealers upon whose unwitting cooperation the scheme depended.”[[40]](#footnote-39)

However, the false information in the scheme to defraud must be material. In *Neder v. United States*, the defendant was convicted of filing false federal income tax returns and of federal mail fraud, wire fraud, and bank fraud. At trial, the District Court determined that materiality with regard to the tax and bank fraud charges was not a question for the jury and found that the evidence established that element. The court did not include materiality as an element of either the mail fraud or wire fraud charges. The Eleventh Circuit affirmed. It held that the District Court's failure to submit the materiality element of the tax offense to the jury was an error under *United States* v. *Gaudin,* 515 U.S. 506, but that the error was subject to harmless-error analysis and was harmless because materiality was not in dispute and thus the error did not contribute to the verdict. The court also held that materiality is not an element of a "scheme or artifice to defraud" under the mail fraud, wire fraud, and bank fraud statutes, 18 U.S.C. §§ 1341, 1342, 1344, and thus the District Court did not err in failing to submit materiality to the jury. Therefore, false information must be material under the reasonable person test.

Most jurisdictions state that actual reliance on the false information is sufficient in most jurisdictions. However, the party that is injured does not necessarily have to rely on the scheme, as is illustrated by *Bridge v. Phoenix Bond and Indemnity*. Plaintiffs, Phoenix Bond & Indemnity Co. and others (Phoenix) filed suit, alleging that Defendants John Bridge and others (Bridge) fraudulently obtained a disproportionate share of liens by filing false compliance attestations. As relevant here, Phoenix claimed that Bridge violated and conspired to violate the Racketeer Influenced and Corrupt Organizations Act (RICO), 18 U.S.C.S. §§ 1961 et seq., through a pattern of racketeering activity involving mail fraud, which occurred when Bridge sent property owners various notices required by Illinois law. The U.S. Supreme Court unanimously held that Phoenix, as the bidders, in asserting the RICO claim predicated on mail fraud, were not required to show, either as an element of their claim or as a prerequisite to establishing proximate causation, that Phoenix relied on Bridge's, its competitors, alleged misrepresentations. A RICO claim based on predicate acts of mail fraud only required use of the mails as a significant part of the alleged pattern of misrepresentations, and reliance by the county was sufficient to support the RICO claim. Further, the bidders could be injured by the pattern of mail fraud through loss of valuable tax liens even if they did not rely on the misrepresentations.

1. *Defining the Element Intent to Defraud*

*​​​​*To be held liable for mail and wire fraud, one must also intend to defraud. Defendants Regent Office Supply Co., Inc., and Oxford Office Systems, Inc., were in the business of selling stationery supplies through salesmen, or "agents," who solicited orders for their merchandise by telephone. Defendants admitted the "sales pitches" employed by their agents used false pretenses and representations to customers, such as asserting that the agent had been referred to the customer by a friend of the customer. Defendants were indicted on mail fraud charges and tried in federal district court under a procedure whereby defendants agreed to be indicted and expeditiously tried upon certain admissions and stipulations of fact constituting the alleged crime. The government's case consisted entirely of defendants' stipulations. On the stipulation, the government rested its case. The district court found defendants guilty as charged, and they received minimal fines for their violations. On appeal, the court ruled that the conduct described in defendants' admissions and stipulations of fact did not come within the prohibition of the mail fraud statute, and particularly the "scheme to defraud" prong. The court ruled that the solicitation of a purchase by means of false representations not directed to the quality, adequacy or price of goods to be sold, or otherwise to the nature of the bargain, did not constitute a "scheme to defraud" or "obtaining money by false pretenses" within the prohibition of 18 U.S.C.S. § 1341. The government did not contend that defendants' agents made any false representations regarding the quality or price of their nationally advertised merchandise. The government offered no direct proof that any customer was actually defrauded by defendants. Moreover, the agents did not attempt to deceive their prospective customers with respect to the bargain they were offering; rather, they gave a false reason for being able to offer the bargain. This shows that while intent to deceive may be there, there must be an intent to defraud in mail and wire fraud.

2. Legal Analysis

To determine if our privacy tool would be mail and wire fraud, we must analyze whether our privacy tool would be wronging someone in his property rights through dishonest means and schemes, also known as fraud. While fraud is generally hard to define, the Supreme Court of the United States defines it as “to wrong someone in his property rights through dishonest means and schemes.”

To determine whether the browser is in furtherance of a scheme to defraud, we must determine who the property rights belong to and whether the privacy tool uses dishonest means and schemes. Because the function of the program is to reconfigure a user’s data and make it appear to be different from what is the truth, it can be reasonably concluded that the privacy tool uses dishonest means and schemes as the primary purpose is for ad companies to not obtain the true data. However, in order to determine if there is fraud, it depends whether ad companies have property rights over our data, and accordingly, whether giving them false data would wrong them in their property rights.

Ad companies have a right to access that data. It is not a form of trespass since there are no laws prohibiting browsers from selling our information to ad companies in the same way that trespass laws prevent people from entering private property. However, our data can be our property, yet other people can still have access to it in some circumstances. For example, Restatement (second) of Torts suggests that a driver may have no cause of action for mere observation or even for having her photograph taken. Similarly, we don’t have a cause of action against browsers and ad companies for having our data as long as they don’t violate any contracts. Under the example of a driver being photographed or observed, while she may have no cause of action against the photographer or observer, she is still allowed to change her appearance in order to deceive the photographer. She can wear a wig, make-up, a costume, or anything else to make herself appear different. The photographer cannot claim that she is violating or wronging him in his property rights by changing her appearance and deceiving him, in spite of the photographer or observer having the right to observe and photograph her. Therefore, just because someone has a right to photograph or observe you without your consent, does not mean they have the right to your honest appearance. You would not be violating their property rights by changing your appearance after finding out they are observing or photographing you. Analogous to this, while ad companies and browsers may have a right to your information, that does not mean they have a right to your real information. Because it is your information, you have a right to change it, and it would not be violating their property rights. Therefore, it is likely that the privacy tool would not be fraudulent. Since no one is being wronged in their property rights, then the privacy tool does not violate Wire Fraud laws.

## Notice and Consent and Breach of Contracts

1. Background

While there are federal policies regulating certain types of data, such as health data or data generated by children, the US does not have a comprehensive data privacy and security policy that covers all types of data.[[41]](#footnote-40) The lack of federal policy means that a large amount of data collection goes unregulated.[[42]](#footnote-41) Only three states have enacted comprehensive data policies: California, Colorado, and Virgina. The California Consumer Privacy Act (CCPA) requires website owners to share what data is being collected and, if any, its eventual use.[[43]](#footnote-42) Although the CCPA does not reach all website owners, it is difficult for a company to conduct commerce without engaging with one of the most populated states in the U.S. Therefore, inadvertently, the CCPA benefits web users living in other states by cornering website owners into changing their practices to comply with the CCPA.

Although Californians are the only people that can vindicate their data privacy rights under the CCPA, could the broader impacts of regulators publishing privacy policies on their websites still mean something for everyone else? Private litigation has played a large role in data privacy and protection enforcement.[[44]](#footnote-43) Two legal approaches to data protection and privacy are through a notice and consent framework and through a contract framework of privacy.

## *The Notice and Consent Approach*

“Privacy policies are the primary mechanism for ensuring valid Notice and [Consent].[[45]](#footnote-44)” At the heart of the notice and consent framework is choice. The Federal Trade Commission (FTC), an agency meant to protect consumers against unfair and deceptive practices, stated in its in its 1998 report to Congress on online privacy:[[46]](#footnote-45)

These core principles require that consumers be given notice of an entity’s information practices; that consumers be given choice with respect to the use and dissemination of information collected from or about them; that consumers be given access to information about them collected and stored by an entity; and that the data collector take appropriate steps to ensure the security and integrity of any information collected.

The European Data Protection Board (EDPB) released guidelines clarifying what valid consent in the digital space looks like. Valid consent must be a clear and affirmative action on the part of the user and must be (1) freely given (2) specific (3) informed and (4) unambiguous indication that the user accepts a website’s use of cookies and trackers to process their personal data.[[47]](#footnote-46) Two methods of obtaining user consent is through clickwrap and browsewrap agreements.[[48]](#footnote-47) While clickwrap agreements present “I Accept” or “I Agree” buttons along with the website terms, browsewrap agreements are visible on a separate webpage which users can reach usually through a hyperlink.[[49]](#footnote-48) Because courts can easily identify when users agree in clickwrap agreements, courts generally find that while clickwrap agreements are enforceable, browsewrap agreements are not.[[50]](#footnote-49)

While choice is generally applauded as a right, perhaps even a human right, it may be better to understand choice in the data privacy world as a method of imposing a burden or duty on web users.[[51]](#footnote-50) In the context of privacy policies, the policy itself is simply information that the user can use to make a choice. If the web user agrees to the terms of the privacy policy, the agreement has the legal effect of shifting liability for when things go wrong from the data collector to the user.[[52]](#footnote-51)

## *The Contract Approach*

Another method of approaching privacy policies is through a contract framework of privacy. Under a contract framework, privacy policies would be legally binding documents holding the parties to the agreement accountable as opposed to simply being informational. Through a breach of contract claim, web users would enforce their privacy rights by pointing out the ways an owner’s conduct violates the policies owner’s voluntarily post to their websites. Generally, to state a claim for breach of contract, a plaintiff must show:[[53]](#footnote-52)

1. The existence of a contract with the defendant;
2. Plaintiff’s performance under that contract;
3. Defendant breached that contract; and
4. Plaintiff suffered damages.

Holding data collectors responsible for either failing to follow through on the terms of their voluntarily published privacy policies or abusing the collected information afterwards seems like the type of accountability that contracts were meant to afford. Unfortunately, courts have construed contract law to conform to and motivate the current practices of commerce at the potential expense of consumers. For example, where a consumer purchased a computer delivered with a physical contract stating that keeping the computer beyond 30 days meant agreeing to the terms of the contract, the court stated that, “[a] contract need not be read to be effective; people who accept take the risk that the unread terms may in retrospect prove unwelcome.”[[54]](#footnote-53) While the breach of contract claim in the case of the computer purchase was over the terms of a physical contract, the practice of courts producing outcomes that promote commerce translates into the digital space as well. Even when consumers have not read privacy or terms of service policies, the existence of a contract may still be recognized.[[55]](#footnote-54)

## *The Current Legal Framework: A Blended Approach*

The current legal framework for approaching privacy policies, for the most part, is notice and consent for two main reasons. First, courts seem to be unwilling to acknowledge even the existence of a contract when using a privacy policy as the basis for a breach of contract claim. For example, where website users brought a breach of contract claim against [Facebook] for “… [using] plug-ins to track logged-out users’ browsing histories when they visited third-party website and then compiled the browsing histories into personal profiles that were sold to advertisers to generate revenue,” the United States Court of Appeals for the Ninth Circuit held that the networking website’s privacy policy merely provided information to cite users and therefore could not sustain a breach of contract claim because no promises had been made.[[56]](#footnote-55) While the District Court for the Eastern District of New York arrived at a different conclusion regarding the legal status of privacy policies, the court noted that its decision to simply assume that a website’s privacy policy was a validly formed contract during the summary judgment stage ran contrary to the approach of other courts.[[57]](#footnote-56)

The second obstacle in website users winning breach of contract claims is showing harm. When website owners breach the terms of their privacy policies and share user information with third parties without user consent, users suffer a loss of privacy. Courts, however, have been unwilling to recognize neither loss of privacy nor the emotional turmoil that stems from a breach as harms sufficient to satisfy a breach of contract claim.[[58]](#footnote-57) Ultimately, “[r]ecovery in contract, unlike recovery in tort, allows only for economic losses flowing directly from the breach.”[[59]](#footnote-58)

Therefore, users are left to try and monetarily quantify how much the data they produce is worth. While users might think that their personal data is valuable, courts have found otherwise. Where a class of passengers brought a breach of contract claim against an airway they had purchased tickets from after discovering that the airway had “...unlawfully transferr[ed] their personal information to [a third party],” the court found that there was “absolutely no support for the proposition that the personal information of an individual … passenger had any value for which that passenger could have expected to be compensated.”[[60]](#footnote-59) Even when users decide not to seek damages and instead ask the court to simply enforce a promise through promissory estoppel, users are still left stranded because they are generally unable to show detrimental reliance on the privacy policy.[[61]](#footnote-60)

There are some instances, however, where contracts can be used to protect user privacy. Unlike privacy policies, courts generally recognize Terms of Service agreements as legally binding contracts. For example, when Google’s Terms of Service and Privacy Policy told users that it would not collect users’ personal data unless the synchronization function was turned on, Google collected user personal information nonetheless. Following the Ninth Circuit’s opinion that privacy policies do not form a cognizable contract due to their lack of commitments, Google attempted to use the same rationale.[[62]](#footnote-61) However, the United States District Court for the Northern District of California rejected Google’s argument finding that a breach of contract had adequately been pled because Google’s Terms of Service, as opposed to privacy policies, required website users to make a commitment to Google’s Terms. Google’s Terms of Service stated, “Terms of Service help define Google’s relationship with you as you interact with our services. [U]understanding these terms is important because, by using our services, you’re agreeing to these terms.”[[63]](#footnote-62)

1. Legal Analysis

In the context of the current legal framework for data protection, privacy tools are attractive. Firstly, the public seems to be willing to use privacy tools in order to protect their data and insulate themselves from the reach of companies. Some 30% percent of U.S. consumers regularly have installed an adblocker.[[64]](#footnote-63) Secondly, while individual data may not be valuable to data collectors, collective data is because it can be bundled and sold.[[65]](#footnote-64) Privacy tools empower the public and allow web users to collectively leverage their data against data collectors.[[66]](#footnote-65)

Data poisoning is one way of collectively leveraging data. Poisoning tools are privacy tools that contribute meaningless or harmful data to data collectors.[[67]](#footnote-66) One example of a poisoning tool is a browser extension called AdNauseam which makes it harder for data collectors to create a consistent profile on users by clicking on all ads presented to a user. AdNauseum therefore hides the ads the user intentionally interacts with.[[68]](#footnote-67) Another poisoning tool is HARPO.[[69]](#footnote-68) HARPO also makes it harder for data collectors to create a consistent profile on web users, but by “... interleav[ing] real page visits in a user’s browsing profile with fake pages.”[[70]](#footnote-69) Our privacy tool poisons data in a method closer to HARPO by feeding trackers false information such as incorrect location data.

## *Banning*

One potential response platforms have to the deployment of privacy tools is an outright ban. For example, AdNauseam was made freely available to consumers in 2014.[[71]](#footnote-70) However, in 2017, Google banned AdNauseum from its Chrome Web Store for purportedly violating Google Web Store’s terms of service.[[72]](#footnote-71) In response to outreach by AdNauseam, Google directed the privacy activists' attention to the requirement that, “[a]n extension should have a single purpose that is clear to users…”[[73]](#footnote-72) Chrome Web Store further clarifies its single purpose requirement for extensions by stating:

An extension must have a single purpose that is narrow and easy-to-understand. Do not create an extension that requires users to accept bundles of unrelated functionality. If two pieces of functionality are clearly separate, they should be put into two different extensions, and users should have the ability to install and uninstall them separately. Common violations include: Functionality that displays product ratings and reviews, but also injects ads into web pages.[[74]](#footnote-73)

Unlike AdNauseam, however, our privacy tool is an entirely separate browser as opposed to an add-on or extension. Users would be able to use our privacy tool by downloading it from our host website. Therefore, our privacy tool would not be subject to removal for violation of Chrome, or other similar, Web Store policies. In fact, a user of our privacy tool could even download and deploy the AdNauseum extension on our browser to offer themselves more protection if they wished.

1. *Clear and Affirmative Action in Denial*

Website regulators often use various tactics to obtain consent that undermine user choice. While clickwrap agreements do have the benefit of showing that a user clicked “I Agree” before using a website with cookies, the notion that freely given, specific, informed and unambiguous user consent in the data privacy world is often illusory.

Denying tracking can be difficult for users. Firstly, while the “I Accept” button may be clearly displayed, the option to deny tracking may be presented more ambiguously such as telling users to “View Cookie Settings.”[[75]](#footnote-74) “The owners of Google and Facebook were both heavily fined for using cookies illegally [by using tactics making it harder for users to freely consent] at the tail end of 2021 by the French data protection authority, Commission Nationale de l’Informatique et des Liberté (CNIL).”[[76]](#footnote-75) Fines, however, can only mitigate Considering the lack of a comprehensive data protection policy, U.S. users are not always given the same protectionary effort.

Regulators even use specificity as a method of wearing down users. Privacy policies and Terms of Service are notoriously ambiguous, lengthy, and technical. After taking 75 of the most popular websites’ privacy policies and assuming an average reading rate of 250 words per minute, researchers found that web users would likely need about 76 work days to read and understand the policies.[[77]](#footnote-76) “The current policy decisions surrounding online privacy suggest that Internet users should give up an estimated $781 billion of their time to protect themselves from an industry worth substantially less.”[[78]](#footnote-77) Furthermore, while users can generally accept cookie policies by simply clicking an “I Accept,” button, users may have to endure a lengthier process for refusing by going through the list of data regulators purportedly collect and choosing which data to deny individually.[[79]](#footnote-78)

The end result of making it difficult to deny consent and weaponizing specificity is that users tend not to read the privacy policies and may not be informed on what they are agreeing to. “Fully 97% of Americans say they are ever asked to approve privacy policies, yet only about one-in-five adults overall say they always (9%) or often (13%) read a company’s privacy policy before agreeing to it.”[[80]](#footnote-79) Although privacy policies may seem to be protecting users, users may actually be agreeing to terms that compromise their privacy in actuality. “[Web users are] being asked to do things that [they] would never be asked to do in other consumer protection settings. Imagine a consumer protection law where [fraud would be legally permissible so long as the consumer was asked] if it is okay to defraud them.”[[81]](#footnote-80)

Infact, while the current notice and consent framework may apply to cookie tracking, the same is harder to say for tracking for digital fingerprinting. The use of digital fingerprinting is particularly concerning considering that this form of tracking, “... is more intrusive than cookie-based tracking for two reasons: (1) while cookies are observable in the browser, browser fingerprints are opaque to users; (2) while users can control cookies (e.g., disable third-party cookies or delete cookies altogether), they have no such control over browser fingerprinting.”[[82]](#footnote-81)

In recognition of the notice and consent framework’s inability to adequately protect web users, the FTC backed away from supporting the framework in its most recent report to Congress and instead called for the enactment of a comprehensive data protection policy.[[83]](#footnote-82) Ultimately, our privacy tool can be used to show that users, by the clear and affirmative action of downloading and using our browser, do not consent to being tracked. The lack of consent is particularly clear considering that our tool attempts to offer a form of blanket protection to users by fighting instances where no notice of tracking was given, but the website tracks users through scripts.

1. *Breach of Contract*

Google is a technology company that is best known for its search engine which allows users to navigate the internet. Overtime, however, Google has grown into a giant that dominates spaces like advertising and cloud computing. The tech company even offers its own brand of electronic devices. With over 1 billion users worldwide of its products and services, Google is able to collect massive amounts of user data.[[84]](#footnote-83) Its wide user base and data collection practices are not the only factors, however, that make Google a prime candidate for contract analysis in regard to use of our privacy tool.

Terms of service are legal agreements between a service provider and users. While Google tries to make its Terms understandable through simple language and short explanatory videos, the policy in its downloadable PDF format is lengthy sitting at about 16 pages. Google tells users in its Terms, “Understanding these terms is important because, by using our services, you’re agreeing to these terms.”[[85]](#footnote-84) So long as they comply with the Terms, users are assured that they can continue to use Google products and services.[[86]](#footnote-85) The assurances that Google makes to its users are examples of promissory language.[[87]](#footnote-86)

The terms of service, however, are not the only thing users agree to comply with when using Google services. Google writes, “You also agree that [Google’s] privacy policy applies to your use of our services.”[[88]](#footnote-87) The privacy policy, “...explains the legal grounds Google relies upon to process [user] information.”[[89]](#footnote-88) Google’s privacy policy is longer than Google’s Terms of service with its PDF format sitting at about 32 pages. Because of its promissory language, Google’s terms of service are a cognizable contract.[[90]](#footnote-89) Furthermore, because its Terms of Service incorporate its privacy policy, Google’s privacy policy is also a cognizable contract.[[91]](#footnote-90) Agreeing to Google’s Terms and Privacy policy through use of services permits Google to collect and use data in accordance with their policies.

Google employs a variety of tracking techniques to obtain user data, “... including cookies, pixel tags, local storage such as browser web storage or application data caches, and server logs.” Tracking allows Google to collect two general categories of information. The first is information that users create or provide to Google, like when making a Google account. Provided information usually includes name, address, or billing information. Users also create information by uploading a photo to Google Cloud or sending a message through Google services. Provided information may also be considered personal information, meaning, “... information that [users] provide … which personally identifies [them], … or other data that can be reasonably linked to such information by Google, such as information … associate[d] with [a user’s] Google Account.”[[92]](#footnote-91) Google states that it does not share personal information without the consent of the user.

The second category of information that Google collects is user activity. User activity information is the … “searches one makes and the advertisements they interact with, visits to websites that use Google services, and one’s location as they use Google services.”[[93]](#footnote-92) Although Google requires user consent When a user is not signed into or does not have a Google account, Google still collects and then stores information, “...with unique identifiers tied to the browser, application, or [user] device…” which allows Google to continue analyzing data to provide targeted advertisements to users. Unlike personal information, Google states:

“We may share non-personally identifiable information publicly and with our partners — like publishers, advertisers, developers, or rights holders. … We also allow specific partners to collect information from your browser or device for advertising and measurement purposes using their own cookies or similar technologies.”[[94]](#footnote-93)

The burden to escape the default practice of data collection and sharing rests on the user. Users are told that through Google’s Privacy Checkup function , “... across Google services, you can adjust your privacy settings to control what we collect and how your information is used.”[[95]](#footnote-94) Users are also given the opportunity to change their advertisement settings. Alongside Google functions that can preserve user privacy, Google also notes that there are both browser and device level settings users can also take advantage of. “For example, [users] can configure [their] browser to indicate when Google has set a cookie in [their] browser. [Users] can also configure [their] browser to block all cookies from a specific domain or all domains.”[[96]](#footnote-95)

Two activities that Google does not permit is fraudulent behavior and disruption to Google services. In regard to fraud, Google tells users, “... don’t abuse or harm others or yourself (or threaten or encourage such abuse or harm) — for example, by misleading, defrauding, illegally impersonating, defaming, bullying, harassing, or stalking others.”[[97]](#footnote-96) Against disruption of its services, Google states, “... don’t abuse, harm, interfere with, or disrupt the services — for example, by accessing or using them in fraudulent or deceptive ways, introducing malware, or spamming, hacking, or bypassing our systems or protective measures.”[[98]](#footnote-97)

While users could use the privacy functions offered by Google, users must overcome the barrier of spending the time and effort to sort through and understand the various settings of Privacy Checkup. Privacy tools that do not require the input of users and can automatically prevent tracking, therefore, can be more attractive to users and ease some of the burden in use. Using a browser that thwarts stateful and stateless tracking is likely not a violation of neither Google’s Terms of Service nor Privacy Policy. In fact, Google’s Privacy Policy even makes users aware of the option to “configure [their] browser’[s]” in ways that can thwart tracking through cookies. Our browser is simply configured to prevent tracking through cookies and scripts.

Our specific browser configuration, however, sends false information in response to tracking retrieval calls. However, along with defrauding, Google lists actions that users might do to other users such as “illegally impersonating” or “bullying.” “Defrauding… others,” therefore, is likely in reference to other users and not Google itself.

Google does however make clear that users should not engage in activity that harms or disrupts their services. Disruptive activity not only includes accessing services fraudulently, but deceptively. As concluded in the fraud portion of our legal analysis, our browser does not defraud others. To be mail and wire fraud, there needs to be material interference with the property of others. While data collectors can try to obtain data from users through methods that conform with their Terms and Privacy Policies, a user’s data is their own and users can change the information that they wish to provide. In regard to disruption, our browser does not harm the functionality of Google’s services. While our browser may make it more difficult for Google to collect data on users, the browser will not impact other user’s ability to use Google services in the way they would like to. Separately, the impact on service providers of using our privacy tool does not arise from a single individual’s use. Data poisoning to a degree that may concern a service provider would likely only arise when a large number of people use the tool. Therefore, an individual’s use of our privacy tool would likely not violate either Google’s Terms of Service nor Privacy policy.

# 

# CONCLUSION

As the internet’s platform continues to expose user privacy vulnerabilities, it’s important to consider potential solutions to assist in protecting our privacy when browsing the internet. While policy reform that shifts the burden of privacy protection onto service providers and focuses on security is the best option, reform is slow and requires consensus. Privacy tools, like our browser, are meant to help lift the burden of privacy protection in the meantime.

Our research aimed to limit third party companies’ ability to reduce user privacy on the internet through cross-site tracking. Through stateful and stateless means, the privacy tool we developed attempts to enhance user anonymity through manipulation of their browsing fingerprint, as measured through the types of advertisements received on a benchmark website. During our experimentation phase, we crawled through a list of location-tracking websites on each of the three browser configurations (Control, User-Agent, and Geolocation). After analyzing the results, we determined that there was not a statistically significant level of independence between advertisements received and browsing configuration used, which indicates that we were unable to exert a meaningful influence on our browsing fingerprint.

Despite this result, we believe that there is still significant room for additional research in this area. A future iteration of this tool could involve the modification of multiple browser APIs, as well as a more nuanced approach to packet alteration. More robust integration of the tool’s two main components, in addition to more time spent testing its behavior, would enable a greater degree of synergy and allow us to inhibit stateful and stateless tracking at the same time. While our privacy tool aims to preserve user privacy in a digital world plagued with cross-site tracking and third-party companies attempting to mishandle its users’ data, further research efforts are necessary to provide users with the virtual privacy they deserve.

In regard to legal ramifications, a user would be unlikely to encounter either fraud nor breach of contract claims. Ultimately, we chose to analyze thow he relationship between users and service providers might be disturbed. Another point of analysis for future research, however, could be legal ramifications privacy tool developers may incur. This proposal for future legal analysis is particularly attractive for data poisoning tools considering that a poisoning tool’s impacts are most felt when large collections of people decide to use the tool. Ultimately, despite all of the difficulties we outlined in our experiment, the development of our privacy was both an incredibly useful and fulfilling experience.

# 

# References

S, K. (n.d.). *Cross site scripting (XSS)*. Cross Site Scripting (XSS) Software Attack | OWASP

Foundation. Retrieved April 18, 2022, from <https://owasp.org/www-community/attacks/xss/>

*What is browser fingerprinting & how does it work?* SEON. (2022, February 23). Retrieved April 18, 2022, from <https://seon.io/resources/browser-fingerprinting/>

*User-agent - http: MDN*. HTTP | MDN. (n.d.). Retrieved April 18, 2022, from <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent>

*Kevin Ripa*. SANS Institute. (2022, February 25). Retrieved April 18, 2022, from <https://www.sans.org/latest/white-papers/33874/>

Englehardt, S., & Narayanan, A. (n.d.). *Online tracking: A 1-million-site measurement and*

*analysis*. Retrieved April 18, 2022, from <https://www.cs.princeton.edu/~arvindn/publications/OpenWPM_1_million_site_tracking_measurement.pdf>

Copland, S. (n.d.). *The top browser fingerprinting techniques explained - fingerprintjs*.

FingerprintJS Blog RSS. Retrieved May 7, 2022, from <https://fingerprintjs.com/blog/browser-fingerprinting-techniques/>

**Repositories**

Lucas and Elias credit reel, resources used, old implementations, and more

* <https://github.com/lausberger/privacy-activism>

Geolocation API modification code

* <https://github.com/lausberger/privacy-browser>

Experiment tool, datasets, website sets, and more

* <https://github.com/lausberger/browser-automator>

1. Englehardt, S., & Narayanan , A. (n.d.). *Online tracking: A 1-million-site measurement and analysis*. Princeton Web Census. Retrieved April 15, 2022, from <https://webtransparency.cs.princeton.edu/webcensus/index.htm> [↑](#footnote-ref-0)
2. Auxier, B., Rainie, L., Anderson, M., Perrin, A., Kumar, M., & Turner, E. (2020, August 17). *Americans and privacy: Concerned, confused and feeling lack of control over their personal information*. Pew Research Center: Internet, Science & Tech. Retrieved April 15, 2022, from <https://www.pewresearch.org/internet/2019/11/15/americans-and-privacy-concerned-confused-and-feeling-lack-of-control-over-their-personal-information/> [↑](#footnote-ref-1)
3. Id. [↑](#footnote-ref-2)
4. Maheshwari, S. (2017, December 28). *That game on your phone may be tracking what you're watching on TV*. The New York Times. Retrieved April 15, 2022, from <https://www.nytimes.com/2017/12/28/business/media/alphonso-app-tracking.html> [↑](#footnote-ref-3)
5. Englehardt, S., & Narayanan , A. (2016). *Online tracking: A 1-million-site measurement and analysis*. Princeton Web Census. Retrieved April 15, 2022, from <https://webtransparency.cs.princeton.edu/webcensus/index.html> [↑](#footnote-ref-4)
6. Auxier, B., Rainie, L., Anderson, M., Perrin, A., Kumar, M., & Turner, E. (2020, August 17). *Americans and privacy: Concerned, confused and feeling lack of control over their personal information*. Pew Research Center. [↑](#footnote-ref-5)
7. *What is fingerprinting?* Surveillance Self-Defense. (2021, July 29). Retrieved April 15, 2022, from <https://ssd.eff.org/en/module/what-fingerprinting#:~:text=Digital%20fingerprinting%20is%20the%20process,%2C%22%20of%20the%20user's%20device>. [↑](#footnote-ref-6)
8. Auxier, B., Rainie, L., Anderson, M., Perrin, A., Kumar, M., & Turner, E. (2020, August 17). *Americans and privacy: Concerned, confused and feeling lack of control over their personal information*. [↑](#footnote-ref-7)
9. Klosowski, T. (2021, September 6). *The State of Consumer Data Privacy Laws in the US (and why it matters)*. The New York Times. Retrieved April 15, 2022, from <https://www.nytimes.com/wirecutter/blog/state-of-privacy-laws-in-us/> [↑](#footnote-ref-8)
10. *Marketing company agrees to pay $150 million for facilitating elder fraud schemes*. The United States Department of Justice. (2021, July 6). Retrieved April 15, 2022, from <https://www.justice.gov/opa/pr/marketing-company-agrees-pay-150-million-facilitating-elder-fraud-schemes> [↑](#footnote-ref-9)
11. The United States Department of Justice. (2021, July 6). [↑](#footnote-ref-10)
12. The United States Department of Justice. (2021, July 6). [↑](#footnote-ref-11)
13. *Katz v. United States*, 389 U.S. 347 (1967). [↑](#footnote-ref-12)
14. *Smith v. Maryland*, 442 U.S. 735 (1979) [↑](#footnote-ref-13)
15. *Id*. [↑](#footnote-ref-14)
16. *Carpenter v. United States*, 138 S. Ct. 2206, 2214 (2018) [↑](#footnote-ref-15)
17. Diamantis, M. (2018). Privileging Privacy: Confidentiality as a Source of Fourth Amendment Protection. *Journal of Constitutional Law*, *21*(2), 485–582. <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3167588> [↑](#footnote-ref-16)
18. Cox, J. (2020, November 16). *How the U.S. military buys location data from ordinary apps*. VICE. Retrieved April 16, 2022, from <https://www.vice.com/en/article/jgqm5x/us-military-location-data-xmode-locate-x> [↑](#footnote-ref-17)
19. Ibid. Cox. [↑](#footnote-ref-18)
20. Tau, B., & Hackman, M. (2020, February 8). *Federal agencies use cellphone location data for Immigration Enforcement*. The Wall Street Journal. Retrieved April 17, 2022, from <https://www.wsj.com/articles/federal-agencies-use-cellphone-location-data-for-immigration-enforcement-11581078600?mod=hp_lead_pos5> [↑](#footnote-ref-19)
21. L. Sweeny, *Uniqueness of Simply Demographics in the U.S Population*, LIDAPWP4. Carnegie Mellon University, Laboratory for International Data Privacy, Pittsburgh, PA: 2000. Forthcoming book titled, The Identifiability of Data. [↑](#footnote-ref-20)
22. Auxier, B., Rainie, L., Anderson, M., Perrin, A., Kumar, M., & Turner, E. (2020, August 17). *Americans and privacy: Concerned, confused and feeling lack of control over their personal information*. [↑](#footnote-ref-21)
23. Heine, I. (2022, April 11). *3 years later: An analysis of GDPR enforcement*. 3 Years Later: An Analysis of GDPR Enforcement | Center for Strategic and International Studies. Retrieved April 15, 2022, from <https://www.csis.org/blogs/strategic-technologies-blog/3-years-later-analysis-gdpr-enforcement#:~:text=Despite%20these%20criticisms%2C%20the%20European,where%20it%20was%20successfully%20used> [↑](#footnote-ref-22)
24. Layton, R. (2022, February 23). *New model code for personal data protection is better than GDPR*. Forbes. Retrieved April 15, 2022, from <https://www.forbes.com/sites/roslynlayton/2022/02/22/new-model-code-for-personal-data-protection-is-better-than-gdpr/?sh=1ac5d3863aee> [↑](#footnote-ref-23)
25. Person. (2022, April 7). *Trends in Privacy & Data Security: Looking back at 2021 and ahead to 2022*. Reuters. Retrieved April 12, 2022, from <https://www.reuters.com/legal/legalindustry/trends-privacy-data-security-looking-back-2021-ahead-2022-2022-04-07/> [↑](#footnote-ref-24)
26. Thomas B. Norton, *The Non-Contractual Nature of Privacy Policies and a New Critique of the Notice and Choice Privacy Protection Model*, 27 Fordham Intell. Prop. Media & Ent. L.J. 181 (2016). Available at: <https://ir.lawnet.fordham.edu/iplj/vol27/iss1/5> [↑](#footnote-ref-25)
27. 18 U.S. Code § 1030 [↑](#footnote-ref-26)
28. U.S. v. Kramer, 631 F.3d 900, 58 A.L.R. Fed. 2d 611 (8th Cir. 2011). [↑](#footnote-ref-27)
29. 18 U.S.C. § 1030(e)(1). [↑](#footnote-ref-28)
30. U.S. v. Aleynikov, 737 F. Supp. 2d 173 (S.D. N.Y. 2010). [↑](#footnote-ref-29)
31. Id. [↑](#footnote-ref-30)
32. Id. [↑](#footnote-ref-31)
33. Id. [↑](#footnote-ref-32)
34. Id. [↑](#footnote-ref-33)
35. LVRC Holdings LLC v. Brekka. [↑](#footnote-ref-34)
36. *Facebook*, 844 F.3d at 1067. [↑](#footnote-ref-35)
37. Id. [↑](#footnote-ref-36)
38. Id. [↑](#footnote-ref-37)
39. ​​*Nosal I*, 676 F.3d at 863 (quoting *Shamrock Foods Co. v. Gast*, 535 F.Supp.2d 962, 965 (D. Ariz. Feb. 20, 2008)). [↑](#footnote-ref-38)
40. *Kann v. United States,* 323 U. S. 88; *Parr v. United States,* 363 U. S. 370; and *United States v. Maze,* 414 U. S. 395. [↑](#footnote-ref-39)
41. Klosowski, T. (2021, September 6). *The State of Consumer Data Privacy Laws in the US (and why it matters)*. [↑](#footnote-ref-40)
42. Ibid. Klowski. [↑](#footnote-ref-41)
43. *California Consumer Privacy Act (CCPA)*. State of California - Department of Justice - Office of the Attorney General. (2022, March 28). Retrieved April 16, 2022, from <https://oag.ca.gov/privacy/ccpa> [↑](#footnote-ref-42)
44. Person. (2022, April 7). *Trends in Privacy & Data Security: Looking back at 2021 and ahead to 2022*. Reuters. Retrieved April 12, 2022, from <https://www.reuters.com/legal/legalindustry/trends-privacy-data-security-looking-back-2021-ahead-2022-2022-04-07/> [↑](#footnote-ref-43)
45. Thomas B. Norton, *The Non-Contractual Nature of Privacy Policies and a New Critique of the Notice and Choice Privacy Protection Model*, 27 Fordham Intell. Prop. Media & Ent. L.J. 181 (2016). Available at: <https://ir.lawnet.fordham.edu/iplj/vol27/iss1/5> [↑](#footnote-ref-44)
46. FED. TRADE COMM’N, PRIVACY ONLINE: A REPORT TO CONGRESS 8 (1998), https:// www.ftc.gov/sites/default/files/documents/reports/privacy-online-report-congress/ priv-23a.pdf [<https://perma.cc/U4RS-RF7A>] [↑](#footnote-ref-45)
47. *What is valid consent?* ICO. (n.d.). Retrieved May 10, 2022, from <https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/consent/what-is-valid-consent/> [↑](#footnote-ref-46)
48. Thomas B. Norton, *The Non-Contractual Nature of Privacy Policies and a New Critique of the Notice and Choice Privacy Protection Model.*  [↑](#footnote-ref-47)
49. Ibid. Thomas B. Norton [↑](#footnote-ref-48)
50. Ibid. Thomas B. Norton [↑](#footnote-ref-49)
51. Cate, Fred. “Data Privacy and Consent.” TED, Jan 2020, <https://www.youtube.com/watch?v=2iPDpV8ojHA> [↑](#footnote-ref-50)
52. Cate, Fred. “Data Privacy and Consent.” [↑](#footnote-ref-51)
53. *In re JetBlue Airways Corp. Privacy Litig.*, 379 F. Supp. 2d 299 (E.D.N.Y. 2005). [↑](#footnote-ref-52)
54. *Hill v. Gateway 2000, Inc*., 105 F. 3d 1147 (7th Cir. 1997) [↑](#footnote-ref-53)
55. *In re JetBlue Airways Corp. Privacy Litig.*, 379 F. Supp. 2d 299 (E.D.N.Y. 2005) [↑](#footnote-ref-54)
56. *In re Facebook, Inc. Internet Tracking Litig.*, 956 F.3d 589 (9th Cir. 2020). [↑](#footnote-ref-55)
57. *In re JetBlue Airways Corp. Privacy Litig.* [↑](#footnote-ref-56)
58. Low v. LinkedIn Corp., 900 F. Supp. 2d 1010, 1028 (N.D. Cal. 2012); *In re JetBlue Airways Corp. Privacy Litig.*, 379 F. Supp. 2d 299 (E.D.N.Y. 2005) [↑](#footnote-ref-57)
59. *In re JetBlue Airways Corp. Privacy Litig.* [↑](#footnote-ref-58)
60. *Id. JetBlue.* [↑](#footnote-ref-59)
61. Thomas B. Norton, *The Non-Contractual Nature of Privacy Policies and a New Critique of the Notice and Choice Privacy Protection Model.*  [↑](#footnote-ref-60)
62. Calhoun v. Google LLC., 526 F. Supp. 3d 605 (N.D. Cal. 2021). [↑](#footnote-ref-61)
63. Id. *Calhoun*. [↑](#footnote-ref-62)
64. Auxier, B., Rainie, L., Anderson, M., Perrin, A., Kumar, M., & Turner, E. (2020, August 17). *Americans and privacy: Concerned, confused and feeling lack of control over their personal information*. Pew Research Center. [↑](#footnote-ref-63)
65. *Marketing company agrees to pay $150 million for facilitating elder fraud schemes*. The United States Department of Justice. [↑](#footnote-ref-64)
66. Vincent, N., Li, H., Tilly, N., Chancellor, S., & Hecht, B. (2021). Data leverage: A framework for empowering the public in its relationship with technology companies. In *FAccT 2021 - Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (pp. 215–227). essay, Association for Computing Machinery, Inc. <https://arxiv.org/pdf/2012.09995.pdf> [↑](#footnote-ref-65)
67. Ibid. Vincent, N., Li, H., Tilly, N., Chancellor, S., & Hecht, B. [↑](#footnote-ref-66)
68. Hao, K. (2021, March 5). *How to poison the data that Big Tech uses to Surveil you*. MIT Technology Review. Retrieved April 18, 2022, from <https://www.technologyreview.com/2021/03/05/1020376/resist-big-tech-surveillance-data/> [↑](#footnote-ref-67)
69. Zhang, J., Psounis, K., Haroon, M., & Shafiq, Z. (2021). Harpo: Learning to subvert online behavioral advertising. *Proceedings 2022 Network and Distributed System Security Symposium*, *3*. <https://doi.org/10.14722/ndss.2022.23062> [↑](#footnote-ref-68)
70. Ibid. Zhang, J., Psounis, K., Haroon, M., & Shafiq, Z. [↑](#footnote-ref-69)
71. *Adnauseam banned from the Google Web Store*. AdNauseam. (2017, January 5). Retrieved April 18, 2022, from <https://adnauseam.io/free-adnauseam.html> [↑](#footnote-ref-70)
72. Ibid. AdNauseam. [↑](#footnote-ref-71)
73. Ibid. AdNauseam. [↑](#footnote-ref-72)
74. *Google chrome web store developer agreement*. Chrome Developers. (n.d.). Retrieved May 10, 2022, from <https://developer.chrome.com/docs/webstore/terms/> [↑](#footnote-ref-73)
75. Gikay, A. A. (2022, February 22). *Cookies: I looked at 50 well-known websites and most are gathering our data illegally*. The Conversation. Retrieved April 19, 2022, from <https://theconversation.com/cookies-i-looked-at-50-well-known-websites-and-most-are-gathering-our-data-illegally-176203> [↑](#footnote-ref-74)
76. Ibid. [↑](#footnote-ref-75)
77. McDonald, A. M., & Cranor, L. F. (2008). The Cost of Reading Privacy Policies. *A Journal of Law and Policy for the Information Society*, *4*(3), 543–568. <https://kb.osu.edu/bitstream/handle/1811/72839/ISJLP_V4N3_543.pdf?sequence=1> [↑](#footnote-ref-76)
78. Ibid. McDonald and Cranor. [↑](#footnote-ref-77)
79. Ibid. Gikay, A. A. [↑](#footnote-ref-78)
80. Auxier, B., Rainie, L., Anderson, M., Perrin, A., Kumar, M., & Turner, E. (2020, August 17). *Americans and privacy: Concerned, confused and feeling lack of control over their personal information*. Pew Research Center. [↑](#footnote-ref-79)
81. Cate, Fred. “Data Privacy and Consent.” [↑](#footnote-ref-80)
82. Iqbal, U. (2021). *Towards A Privacy-Preserving Web* (thesis). The University of Iowa, Iowa City. [↑](#footnote-ref-81)
83. FED. TRADE COMM’N, REPORT TO CONGRESS ON PRIVACY AND SECURITY (2021), <https://www.ftc.gov/system/files/documents/reports/ftc-report-congress-privacy-security/report_to_congress_on_privacy_and_data_security_2021.pdf> [↑](#footnote-ref-82)
84. Djuraskovic, O. (2022, April 20). Google Search Statistics (2022): 17+ facts you need to know. FirstSiteGuide. Retrieved May 7, 2022, from <https://firstsiteguide.com/google-search-stats/#:~:text=Google%20has%20over%201%20billion,has%20the%20biggest%20customer%20base>. [↑](#footnote-ref-83)
85. Google. (n.d.). *Google terms of service – privacy & terms*. Google. Retrieved May 10, 2022, from https://policies.google.com/terms?hl=en-US [↑](#footnote-ref-84)
86. Ibid. Terms of Service. [↑](#footnote-ref-85)
87. *Calhoun v. Google LLC.*, 526 F. Supp. 3d 605 (N.D. Cal. 2021). [↑](#footnote-ref-86)
88. Terms of Service. [↑](#footnote-ref-87)
89. Google. (n.d.). *Privacy policy – privacy & terms*. Google. Retrieved May 10, 2022, from <https://policies.google.com/privacy?hl=en-US> [↑](#footnote-ref-88)
90. *Calhoun v. Google LLC.* [↑](#footnote-ref-89)
91. Id. [↑](#footnote-ref-90)
92. Terms of Service. [↑](#footnote-ref-91)
93. Ibid. Terms of Service. [↑](#footnote-ref-92)
94. Privacy Policy. [↑](#footnote-ref-93)
95. Ibid. Privacy Policy. [↑](#footnote-ref-94)
96. Ibid. [↑](#footnote-ref-95)
97. Ibid. Terms of Service. [↑](#footnote-ref-96)
98. Ibid. Terms of Service. [↑](#footnote-ref-97)