Giving Users Insight into Web Fingerprinting

Yasha Mostofi
mostofi2@illinois.edu

ABSTRACT

A majority of users are not aware of the data being collected about them as they browse the web, allowing many web services to get away with collecting arbitrary data to fingerprint the user across different sites, primarily for targeted behavioral advertisements. As such, users are unaware of how certain advertisements show up, which causes public discomfort [36]. The proposed solution is to present all of the different attributes about the user that are being collected by a service the first time a user navigates to that website via a browser extension, primarily to inform users about what data is collected about them, but also secondarily, to allow users to make a more informed choice about whether or not they wish to use a certain service. With this extension, expected results show that users are less likely to continue using websites that heavily farm their machine for information, unless the user can find reasonable justification, like for banking websites wanting to detect and prevent fraudulent use.

1 INTRODUCTION

As web tracking becomes more prevalent, users are often unaware of how much different technologies are able to track them for analytic and advertisement purposes, especially across different websites [23]. Just knowing a single web page load is enough to determine plenty of information about a user, and once patterns can be analyzed, then even stronger inferences can be made [23].

Cookies are growing in popularity and sometimes are considered "critical" for site functionality. New technology developments allow for more complex tracking. When cookies fall short, some services opt for fingerprinting, a mechanism by which data about a user's environment allows a website to track them across page loads, networks, browsers, and when cookies are blocked or tampered with. These fingerprinting services are growing in popularity, and now there are even companies dedicated to providing fingerprinting for other websites [1, 11].

Studies have shown that the amount of data that can be obtained from fingerprinting users is quite large, and allows for a rather high number of bits of entropy to be collected [11]. While some, more technologically savvy users might be aware of this, other users are generally concerned about privacy but don't understand the depths of web tracking and furthermore how to protect their privacy - which can be as simple as avoiding certain services. The remaining users aren't concerned about privacy or web tracking at all - either because they don't care or have no idea how much data is collected on them. Fortunately, as web tracking grows, journalists publish more and more articles exposing some of the harsh realities to everyday Internet users. For example, the New York Times recently published an article detailing how several different mobile apps track - and often sell - user's location data. One example used was Lisa Magrin, whose location was recorded as often as every two seconds [7].

Since browser extensions are already a ubiquitous way for users to "adjust" their browsing experience to satisfy their needs [32], the proposed solution is to expose what sort of information about the user's environment is being collected on a given website. This is similar to the New York Times article but attempts to showcase fingerprinting and how common it is all across the web. The motivation behind this is the belief that educating users about how common privacy violations are on the web might lead to more widespread change. Recently, tools have been developed to bring tracking and subsequently, a lack of privacy in the real world. Taylor Swift used facial recognition hidden within a kiosk displaying videos to try to catch any of her known stalkers [13], which none of the attendees knew about or consented to, even though one could argue that private events, like concerts, don't need explicit consent. Concerts aren't the worst of it since similar facial recognition technology is being developed for the military but might be used for domestic uses too [10]. Both of these uses are very under the radar but are much worse than the privacy invasions users are subjected to on the web. Users being educated about these would prevent companies from even exploring these avenues of tracking, which keep getting worse. Shortly after acquiring Ring, the maker of a smart doorbell, Amazon plans to add facial recognition technology to the built-in camera to notify homeowners of "suspicious" people [34]. Stopping web tracking is the first frontier - educating more users about how little privacy they have on the web will hopefully prevent even worse tracking technology from being developed.

2 BACKGROUND

2.1 Fingerprinting

As the hunger for tracking users on the web has grown, companies have moved from cookies - which can be blocked or tampered with - to more complex fingerprinting-based methodologies. There are companies [1] that sell fingerprinting services to third parties. In one study done by the EFF, the researchers implemented one possible fingerprinting algorithm and observed that the distribution of their fingerprint contains at least 18.1 bits of entropy - meaning, that a random fingerprint will happen again only in 1/287,777 other browsers, at best [11]. The EFF also has a site where users can be fingerprinted with their algorithm to see how unique their fingerprint is, to measure how safe they are against tracking on the Internet.

This is because fingerprinting is a substantial threat to privacy, and is now becoming much more accessible for websites that use JavaScript (pretty much all web services) to easily incorporate fingerprinting in their code, even via open source libraries. One library even allows the end user of the library to make the tradeoff between time to fingerprint and how much data to incorporate in the fingerprinting process [38].

Fingerprinting is not limited to attributes of the environment, as one study found that using only 50 web pages in a user's browser history is sufficient to fingerprint 42% of users [27]. A lot of the data available for fingerprinting comes from standard JavaScript methods that exist in modern web browsers that allow scripts to access a user's DNS or browser cache by measuring how *long* certain operations take [14].

However, fingerprinting can be (and is) used constructively. Many services catch fraudulent logins by capturing a fingerprint of each login and comparing it to the set of previously captured. The line between constructive and destructive fingerprinting is difficult to identify since both use the same technology. The browser exposes a set of information via standard JavaScript functions on the `window`, `navigator`, and `screen` objects. Some functions allow for certain fingerprinting techniques even to identify a browser to its minor version [26]. Also, extensions that aim to tackle fingerprinting are shown to all fail to hide the browser's true identity [26] completely.

2.2 Chrome Extensions

Browser extensions allow a user to customize many aspects of their browsing experience. There are extensions like NoScript, which can completely block all JavaScript execution on some pages, to other extensions like Stylish, which allow for custom CSS to theme websites in ways that the website developer might not want or support, like a dark mode.

The Chrome web browser allows users to install extensions from a Web Store, where developers can publish extensions with descriptions and screenshots. The Web Store also facilitates user reviews, to help combat malicious extensions. Alternatively, users can download an extension from any third-party (GitHub is relatively common) and install it in their browser themselves. Those extensions can be either packed or unpacked, and Chrome allows the user to load either type quickly. Unpacked extensions are a set of files in a folder, and can only be loaded if Chrome is set to developer mode (which any user can easily do).

Extensions must have a `manifest.json`, which informs Chrome of the behavior of the extension, as well as some metadata, like the name, version, and more. The manifest also indicates what browser permissions the extension requires, and more importantly, what files it requires. The `notifications` permission allows the extension to have access to native notifications via the `chrome` object in JavaScript, and similarly, the `tabs` permission gives support for querying what tabs the user has open.

With regards to scripts, extensions have two types - background and content. The background script represents the "run-time" of the extension. This means the script is executed when the extension is loaded, regardless of what tabs are open or if any are open at all. The background script, as a result, has no access to any information about what websites the user has open, or anything else, but does have access to extension permissions via the `chrome` run-time. Background extensions are also typically used to manage state, like storing a user identifier or connecting to datastores, but neither are used in the initial version for this study.

The other script type, content, runs alongside any tabs that are open that match a URL specification set in the manifest. The manifest defines a `content_scripts` array, and each entry defines a set of scripts, when to run them in relation to loading a specific page, and matching URL's. As an example, the following snippet defines a `content_script` that runs on every URL at `document_start` (explained 4.1) using the file `content.js`.

```
{
  "matches": [
    "<all_urls>"
  ],
  "run_at": "document_start",
  "js": ["content.js"]
}
```

Listing 1: Extension manifest snippet

Since the background script has no context about the currently open tab and the content script does not have access to the extended `chrome` run-time, Chrome allows the two scripts to communicate via JSON messages. For the manifest defined above, `content.js` and whatever scripts exist on the current page are sandboxed to prevent collisions or strange behavior. As such, the script is loaded before the DOM (document object model) is parsed by Chrome but after it has been loaded, which allows the script to modify the DOM and, as a result, modify the JavaScript environment of the tab, bypassing the isolation mechanism that Chrome provides.

3 RELATED WORKS

3.1 Users and Tracking

Many different user studies over the past decade have shown that a majority of users are uncomfortable with third parties collecting and using browsing activity, especially for advertisement purposes [23, 36]. Unfortunately, policies and legislation (at least until EU's GDPR) have not helped users with this issue. Advertising groups argue that tracking is okay and should be the default [23]. Even with legislation like GDPR being passed in the EU, many still claim that it is fundamentally incompatible with the "big data" ecosystem we have in place today [40], even though individual users still don't have a say in how their data is used, what data is even collected, or which third-parties have access.

Even though users are aware of contextual targeting, studies have found that they are unaware of how that targeting comes to be [37]. Participants in such studies had strong concerns about data collection, but unfortunately, the environment in which a user browses the web gives them little to no control over their privacy on the web. Even though some websites provide privacy policies and opt-out tools, they are challenging to understand [21], cumbersome to use [20], and unfortunately rarely read or used by users [24]. Although, one study showed that young adults were interested in changing their Facebook privacy settings when their privacy policies were contested [9]. Unfortunately, it is not scalable to rely on media scrutiny as an incentive for users to pursue better privacy settings on the web.

3.2 Browser Extensions

Since the tooling provided by websites is rarely sufficient to curb privacy violations [21, 24], many users turn to web browser extensions to either eliminate trackers via ad blockers, display tracking activities on a given site, or, in the most extreme case, eliminate JavaScript. Unfortunately, disabling JavaScript can make some sites refuse to serve the user [18]. However, scripts provide a majority of the attack vectors websites use to collect data about a user, and also as a bonus, reduces the load time and network resource usage of more complex pages. As a result, with how prevalent JavaScript is across the web, it's not feasible to expect users to disable JavaScript, especially since Electron and other similar platforms have even moved JavaScript to the desktop, and React Native to mobile.

Extensions that show users more information about whether or not they are being tracked are sometimes helpful, especially since privacy awareness allows users to model privacy risks and as a result, better manage online privacy with their expectations and privacy preferences [28]. Research has shown that when users use privacy-related extensions, even ones with limited functionality, they show an increase in concern for privacy due to increased awareness of tracking [32]. It's important to note that, with regards to online tracking, user awareness involves not only what data is being collected, but the entity collecting it and for what purpose [29]. Awareness, more importantly, leads to people without a technical background to take steps to protect their privacy [22]. This conclusion is the reason why the proposed extension might have a substantial impact on users and sites engaging in fingerprinting them.

3.3 Tracking

Even though a majority of web tracking is for targeted behavioral advertisements, researchers continue to develop faster and more efficient ways to fingerprint users [25]. This is because websites like banking websites want to prevent fraudulent accesses. Patents [15] as well as novel ideas [17], like tracking patterns in keystrokes as biometric authentication, are being proposed to help prevent malicious attacks. Fingerprinting, for some researchers, is analogous to bio-metric authentication. Other researchers, working to prevent tracking, argue that the tracking used to prevent fraud should be limited to within a single website, rather than *cross-website* tracking [35]. As such, tools like FP-Block [35] or Intelligent Tracking Prevention 2.0 [6], exist to try to balance anti-fraud tracking and privacy violations. Furthermore, even though many advertisement-heavy services claim that full tracking is needed, a study expanded on the Do Not Track requirements and discovered that adequate and practical solutions could exist for advertisement auctions without exposing full user behavior [31]. Another design, Privad, was able to preserve the core functionality required: ads shown on pages, targeting based on keywords and other attributes, ranking via auctions, view and click accounting, and defense against click-fraud while maintaining a much higher level of user privacy [16].

On mobile platforms like Android, many third-party libraries that are used for good purposes have been shown to report back location data and other personally identifiable information. When evaluating apps made for children on Android, researchers found that a substantial portion of the six-thousand apps tested had third-party libraries that transmitted some sort of personal information [30]. One reason that is commonly cited for this lack of privacy on Android is that any library used in an app inherits the full permission set from the app, which is why new tools like FlexDROID aim to give the app developer more control over what data can be accessed from the app [33]. FlexDROID allows an Android app developer to have much finer-grained access control over any third-party libraries used in their app, and also allows the developer to mock responses.

```
<flexdroid android:name="com.third.party.library"
    android:mockOnException="true">
</flexdroid>
```

Listing 2: Simple FlexDROID Rule

However, the war between obtaining extra information and privacy-protecting tools continues. When traditional mechanisms fall short, if a third-party wants access to a user's browsing history, they can resort to using security flaws in GPUs. Both NVIDIA and AMD GPUs do not initialize newly allocated GPU memory pages, which can be used by an adversary. Researchers, in 2014, were able to apply this to Chromium and Firefox web browsers (which use GPUs for accelerated web page rendering) and discovered that both browsers leave rendered textures in memory, and as such, were able to infer which web pages a victim had visited by analyzing the remaining textures and had up to 95.4% accuracy. [19]. Blocking the easier tracking mechanisms (like cookies and fingerprinting) means that services have to resort to these kinds of tactics, which are prohibitively difficult and might result in a decrease of tracking done on the web.

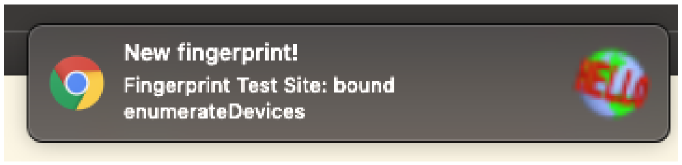
3.4 Nudging

The proposed solution is, in essence, a form of nudging. It's commonly used within research as a mechanism to change some part of human behavior. For example, researchers used a modification to the Facebook web interface to nudge users to consider what they post, how often, and who might be seeing it [39]. Two different nudges were tested, and each form targeted a different aspect of Facebook usage. Some participants found the nudges helpful, adopting better practices, while others found them annoying. Even though the results are inconsistent, nudging is still a powerful tool and has been used to help users be more informed when they decide a privacy trade-off since 2009. Researchers claimed that, once a preference for privacy is formed, it is improbable to change, but different sites take advantage of inconsistencies by framing choice alternatives in different ways and masking information [8]. This is because most users have a difficult time estimating the consequences of a lack of privacy, allowing for various things to influence the way people value their data and how they act on that perspective [8].

4 CHROME EXTENSION: IN DETAIL

This project consists of two components: one, a Chrome extension that notifies users when data is collected on them commonly used for fingerprinting, and two, a user study to see how those notifications impact user browsing behavior. As discussed in 2.2, Chrome extensions have three components. The `manifest.json` defines the extension, asks Chrome for a permission set, and defines the background and content scripts. The background scripts represent the "run-time" of the extension and typically run as long as Chrome is open. The background scripts also have access to any extra permissions granted to the extension via the `chrome` object. The content scripts run in the foreground. A new instance is launched whenever a URL match occurs. The content scripts have access to the DOM of the tab but run in an isolated JavaScript environment.

Figure 1: Example Fingerprint Notification


New fingerprint!
Fingerprint Test Site: bound enumerateDevices

4.1 Manifest

In the manifest file, a single `background.js` script is set to be the background script, and it also requests the `notification` permission. In addition, a single `content.js` script is set to be the content content script that runs on every URL. The script is set to run at `document_start`, which means `content.js` is ran after the DOM is loaded but not yet parsed by Chrome, which allows for JavaScript injection.

4.2 Background

As explained earlier, the `background.js` script represents the runtime of the extension. The extension is given access to any extra functionality requested in the manifest through the `chrome` object. For this study, the extra permissions used are the `notification` and `tabs` permissions, which are fairly straightforward to use [3, 5].

```
var opt = {
  type: "basic",
  title: "Notification",
  message: "Sample Notification",
  iconUrl: "icon.png"
}
chrome.notifications.create(opt);
```

Listing 3: Sending a notification from `background.js`

Because only the background script has access to the notifications object, the background script has to wait for messages from the content script before sending a notification. This is done through the `onMessage` event part of `chrome.runtime` [4], which allows the content and background script to communicate with each other. The background script sets up a callback that takes in the message and then crafts an `opt` object to use for a notification. The notification should also include the name of the website that used the fingerprint function, so the background script uses the `chrome.tabs` object to get the name of the currently selected tab.

```
chrome.tabs.getSelected(null, function(tab) {
  console.log(tab.title); // provides the current tab
     title
});
```

Listing 4: Getting the current tab

4.3 Content

The content script, unlike the background script, is launched every time a new page is visited. The content script's primary responsibility is to hook the functions and then send a message to the background script via `chrome.runtime`. However, Chrome extension's content scripts run in an isolated JavaScript environment, meaning, they cannot modify the JavaScript environment of the tab. A naive approach for hooking any JavaScript API functions is below:

```
window.navigator.mediaDevices.enumerateDevices = function
     () {
  console.log("Function was called!");
  return window.navigator.mediaDevices.enumerateDevices()
     ;
}
```

Listing 5: Naive JavaScript hook

This would work perfectly if the environments were not isolated, but unfortunately, a different approach is needed. Fortunately, this is something many extension developers wish to do, and as such, a workaround has existed publicly since 2010 [2]. A modified approach is listed below:

```
var hook = function() {
  var wrap = function(someFunction){
    var wrappedFunction = function(){
      var args = [...arguments].splice(0)
      console.log('fn: ${someFunction.toString()}\nargs:
     ${args}')
      return someFunction(args)
    }
    return wrappedFunction
  }
  window.navigator.mediaDevices.enumerateDevices = wrap(
     window.navigator.mediaDevices.enumerateDevices)
};

var script = document.createElement('script');
script.textContent = '(' + hook.toString() + ')()';
document.documentElement.appendChild(script);
```

Listing 6: Hooking JavaScript function via script injection

In the snippet, `hook` defines a function `wrap` which takes in `someFunction` and replaces it with a wrapper that logs whenever the function is called along with its arguments. As an example, hook wraps the (enumerateDevices) function, one of many that Fingerprintjs2 uses. The last three lines of the snippet perform the JavaScript injection. First, it creates a new script object on the generic document. Then, the next line inserts the body of the hook function defined above, with an immediate evaluation of it (via the `()`). Finally, the script object is added to the generic document object. An initial approach for the last line would be to add it to the body of the document, a common approach used in extension development when wanting to modify the DOM. However, since this script is executed on `document_start`, the body is not yet available, and as such, the injected code would run *after* the DOM's scripts, defeating the purpose of trying to hook the function. As a result, the content script adds the script object to the document. Because it is evaluated before the DOM is parsed by Chrome, hook runs before any other scripts would, modifying the version of `enumerateDevices` that they would use. This is why the content script is set to run at `document_start`.

While this approach allows for modifying the JavaScript API body, the scope of the function `wrap` is in the tab, not in the extension, meaning that `wrap` does not have access to `sendMessage`. This is okay in the snippet above (6), but the goal of the extension is to notify the user via a native extension, not a log in the console. And since the content script doesn't even have access to the notifications API's, the tab now has to communicate to the content script, which then has to communicate to the background script. Fortunately, each one of those pairs has a mechanism by which to communicate. The workaround is to chain multiple events until the background script can send a notification. The script injected onto the DOM sends an arbitrary event via the `document` object that the content script can set up a listener for, and then in that listener, communicate to the background script.

```
var newEvent = function(msg) {
  var event = new CustomEvent('Event', {
    "detail": {
      message: msg
    }
  });
  event.initEvent('hook');
  document.dispatchEvent(event);
}
```

Listing 7: Sending an event

```
document.addEventListener("hook", function(data) {
  doSomething(data);
});
```

Listing 8: Event listener

With both of these workarounds combined, the content script can modify the JavaScript on the DOM to hook any API functions, and then use `document` events to communicate back to the content script to then communicate with the background script.

5 USER STUDY: IN DETAIL

To test the efficacy of the proposed Chrome extension, the next component of the project is a simple user study [1].

5.1 Pre-Study Preparation

Before starting the actual study, I would select somewhere between twenty to fifty sites [2] with various levels of fingerprinting, based on data from [12]. When selecting these sites, it's important to include sites that have minimal to no fingerprinting, heavy use of fingerprinting, and a few sites in between. It's important also to consider how common they are since the goal of the study is to see how a user's browsing habits change if they are exposed to knowledge about how sites fingerprint them. In addition, every site selected would have to be tested with the extension, since the study [12] has a much broader definition of fingerprinting, but the sites used in the study would have to be using functions that the extension hooks.

5.2 Pre-Study Survey

After selecting some sites to use in the study and collecting participants [3], each participant in the study would have to complete a pre-study survey. The primary goal of the survey is to capture their browsing habits before using the extension for any period. The survey asks the user how often they visit each website on the pre-selected list of websites, using a Likert scale. If the user indicates any answer above "never visited," the survey would ask a simple followup question: "how much data do you think this website collects about you?".

Each question on the survey serves a significant purpose. First, knowing how often a user visits a website allows us to understand how their browsing habits change throughout the study. If a user never visits a website that already heavily fingerprints users, then the extension exposing that information to the user has no impact on their browsing.

Second, asking users how much data they think is collected on them gives researchers insight into how effective the extension is, and if the user even cares about data collection. If they indicate that many websites collect lots of data but still visit those websites frequently, then the extension is not going to have an impact on their browsing habits, even for different websites.

5.3 Study

For the actual study, participants will be sent instructions for how to download the extension. After installation is complete, the user would just be instructed to use the Internet, and that's it. The study does not collect any data on participant browsing habits, because while that data might show how browsing habits change, it doesn't include any information as to why a particular user reduced usage of a site. For the first version of the study, participants will use the extension while browsing for one month. Unfortunately, one month might be too short to see a statistically significant change in browsing habits, but the data from the first run will be enough to decide if the study should be run again with a longer time frame or with an extension described in 6.

During the study, users may want to see a privacy policy from the actual extension, since it might appear that the extension is collecting sites they visit (which is relatively common in studies that are extension-based). As such, users should have very easy access (either in an email sent to them or part of the extension) to a privacy policy outlining the lack of data collection in this version of the study, and how the extension only serves as a mechanism to alert the user of information based on their browsing habits. This privacy policy should also include relevant source code for participants that are more technically savvy. The primary purpose behind this is that often people who are familiar with what an adversary could do through a Chrome extension might choose not to participate in a study like this, and including this type of privacy policy would help retain some of those users.

In addition, the setup information for the study should also include instructions on how to quit the study early, since some users might feel overwhelmed with the notifications and want to quit. The instructions would include how to uninstall the extension and also to notify the researchers to not follow-up with the post-study survey. Furthermore, the Chrome extension icon will be visible in the toolbar [4] so that participants always know that they are in the study.

5.4 Post-Study Survey

After the period has elapsed, all users would be sent a final post-study survey as well as instructions on how to uninstall the Chrome extension. The post-study survey serves a few different purposes

[1]Due to a lack of IRB, the study was not completed.
[2]This number depends on how many sites fit in each category that people use.
[3]There's no strict requirement on participant demographics, as long as its representative of a variety of Internet users.
[4]This is an explicit choice and is off by default.

and collects the actual data that would be used to make any conclusions about the efficacy of the Chrome extension and user behavior when exposed to data collection practices.

First, the survey will show users the same list of websites from the pre-study survey except without any of the sites they said they never visit. The survey would ask them to rate how often they will visit the site from now on, and how much data they believe the website collects about them. This is the most critical part of the study since it provides researchers with valuable data regarding any change in browsing habits. In addition, there might be some users that still visit websites frequently, but now have a much better idea about that site's data collection and fingerprint practices, which is still a successful outcome for the extension.

Based on the post-study survey data, two conclusions could be made. The one that would mark a successful study would be if a statistically significant proportion of users claim they will visit sites less and also indicate an increase in perceived data collection for those same sites. The other potential successful outcome of the study would be to see participants mark an increase in perceived data collection in general, since then the goal of educating users about hidden tracking mechanisms like fingerprinting was accomplished. Another potential outcome that might be an indicator of success would be an uptick in visits to sites that have less fingerprinting, since this would mean that the participants started exploring other sites - potentially as replacements for sites with poor privacy - but didn't have enough time to completely switch to the other site.

After participants have recorded data about the websites, they will be asked standard demographic questions. The reason for asking demographic questions after the study is that participants might alter their browsing habits during the study to conform with a stereotype that matches their demographics, especially if questions like income or race are asked in the demographic portion. Asking demographics after allows participants to not feel any pressure to alter their browsing habits during the study, and subsequently make the results not as valuable.

6 DISCUSSION & FUTURE WORK

6.1 Website Selection

In the study described, there's a chance that from the websites chosen, many of the study participants wouldn't actually use them enough for the data to have a statistically significant change in browsing habits. One way to fix this issue would be to run the extension "silently" for the same time period as the actual study, but before, to collect what websites the users visit and how much those websites use the functions hooked by the extension.

This simple modification allows for the pre-study to only focus on perceived data collection, and also provides much more fine-grained data regarding browsing habits, since now the extension has collected real browsing before the notifications.

Furthermore, as described below, the hooking mechanism used breaks fingerprinting, so some users might also experience a change in basic website functionality during the study, which would have an impact on the results. The best way to solve this problem would be to merely hardcode the extension to not hook functions on websites that are known to break. Obviously, given the size of the Internet, it's not possible to get all sites, but it would drastically reduce the change of broken website functionality even existing in the study.

6.2 Non-destructive Extension

Currently, the hooking mechanism that is shown in 2.2 breaks the hooked function. This is okay for this study since the purpose is only to inform the user when those functions are being used. However, there are two areas where this could be improved.

First, a new data point of the study could be recording what sites break when the functions used for fingerprinting don't work correctly anymore. While not directly related to understanding how browsing habits would change, this would be a strong indicator that the functions used for fingerprinting aren't being used for behavioral advertising or malicious purposes, but could also be websites simply disabling functionality due to decreased revenue. A lot of financial services have recently begun to fingerprint visitors to prevent fraudulent accesses. Websites that break with the current extension would be most likely using fingerprinting to detect and prevent fraud.

Second, preserving fingerprinting functionality may impact browsing habits, as then users would be exposed to potentially more privacy-invasive advertisements. For example, if a website uses fingerprinting for behavioral advertisements, if the functions fail when called, the service would be forced to show standard advertisements, and the user might not feel that their privacy is being invaded even if they are getting notifications about various data being collected. As a result, retaining fingerprint functionality might have a substantial impact on the results of the study.

6.3 Mocked Fingerprint Data

As discussed in 3.3, FlexDROID allows Android developers to mock responses for privacy-sensitive Android API calls, which inspires an exciting set of additions for the extension described in this study. The extension could expose an interface for the user to be able to see what the fingerprinting functions return and modify their return values to be whatever they want. For example, the user could remove any custom fonts they have installed from the font list returned, which would, therefore, decrease the uniqueness of their fingerprint. Another interesting avenue to explore would be for the extension to actually collect top return values for the fingerprinting functions and then allow users to pick from one of those instead of having to customize their own. This would both allow them to change a single value, for example returning the default Windows 10 installed fonts, or to completely mirror a different system by picking a whole set of return values, like a fresh install 13 inch MacBook Pro running MacOS Mojave. Or, the user could configure to return the most common fingerprint set at that time, which would be essentially removing all value from the fingerprinting result.

Giving users more control over fingerprinting has the most exciting potential for this project because users can take a more active stance against fingerprinting by choosing how much data to give to trackers, if any at all. Users could also make fingerprinting much less useful for tracking and advertisement companies by returning artificial data, paving the way for a less privacy-invasive future.

A SAMPLE PRE-STUDY SURVEY

Thanks for participating in this study! In the pre-study survey, you'll have to answer two questions about a handful of sites. For each site, please indicate how much you use that site, ranging from never (have never visited it) to every day (or multiple times per day). If, for a given site, you have visited it at least once, please indicate how much data you think that website collects about you.

(1) **facebook.com**:
  (a) How often do you visit **facebook.com**?
  O Never O Rarely O Sometimes O Regularly O Every day
  (b) If more than *never*, how much data do you think this website collects about you?
  O Nothing O A Few Items O A Couple Items O A Lot of Items O Everything About Me
(2) **google.com**:
  (a) How often do you visit **google.com**?
  O Never O Rarely O Sometimes O Regularly O Every day
  (b) If more than *never*, how much data do you think this website collects about you?
  O Nothing O A Few Items O A Couple Items O A Lot of Items O Everything About Me
(3) **bbc.com**:
  (a) How often do you visit **bbc.com**?
  O Never O Rarely O Sometimes O Regularly O Every day
  (b) If more than *never*, how much data do you think this website collects about you?
  O Nothing O A Few Items O A Couple Items O A Lot of Items O Everything About Me
(4) **newyorktimes.com**:
  (a) How often do you visit **newyorktimes.com**?
  O Never O Rarely O Sometimes O Regularly O Every day
  (b) If more than *never*, how much data do you think this website collects about you?
  O Nothing O A Few Items O A Couple Items O A Lot of Items O Everything About Me
(5) **news.ycombinator.com**:
  (a) How often do you visit **news.ycombinator.com**?
  O Never O Rarely O Sometimes O Regularly O Every day
  (b) If more than *never*, how much data do you think this website collects about you?
  O Nothing O A Few Items O A Couple Items O A Lot of Items O Everything About Me

B SAMPLE POST-STUDY SURVEY

Thanks for completing the study! First, if you haven't already, please navigate to chrome://extensions/ and click **Remove** under `Fingerprint Notifier` to remove the Chrome extension from your browser. After you've removed it, you won't receive any notifications any more. If you have any trouble, please let us know! We would be happy to help you with any technical difficulties you may encounter.

In this post-study survey, you'll be asked a series of questions similar to the pre-study survey. There will be a list of sites, and for each site, please indicate how much you plan to use that site, ranging from never (not going to visit it again) to every day (or multiple times per day). If, for a given site, you have visited it at least once (both during and before the study), please indicate how much data you think that website collects about you.

(1) **facebook.com**:
  (a) How often do you visit **facebook.com**?
  O Never O Rarely O Sometimes O Regularly O Every day
  (b) If more than *never*, how much data do you think this website collects about you?
  O Nothing O A Few Items O A Couple Items O A Lot of Items O Everything About Me
(2) **google.com**:
  (a) How often do you visit **google.com**?
  O Never O Rarely O Sometimes O Regularly O Every day
  (b) If more than *never*, how much data do you think this website collects about you?
  O Nothing O A Few Items O A Couple Items O A Lot of Items O Everything About Me
(3) **bbc.com**:
  (a) How often do you visit **bbc.com**?
  O Never O Rarely O Sometimes O Regularly O Every day
  (b) If more than *never*, how much data do you think this website collects about you?
  O Nothing O A Few Items O A Couple Items O A Lot of Items O Everything About Me
(4) **newyorktimes.com**:
  (a) How often do you visit **newyorktimes.com**?
  O Never O Rarely O Sometimes O Regularly O Every day
  (b) If more than *never*, how much data do you think this website collects about you?
  O Nothing O A Few Items O A Couple Items O A Lot of Items O Everything About Me
(5) **news.ycombinator.com**:
  (a) How often do you visit **news.ycombinator.com**?
  O Never O Rarely O Sometimes O Regularly O Every day
  (b) If more than *never*, how much data do you think this website collects about you?
  O Nothing O A Few Items O A Couple Items O A Lot of Items O Everything About Me
(6) Demographic Questions
  (a) Your age:
  (b) Your gender:
  (c) How many times a week do you go on the web?
  O Less than once a week O Between once and three times O Between three times a week and every day O Every day

C LIST OF FUNCTIONS HOOKED

```
window.navigator.devices.enumerateDevices
window.OfflineAudioContext
window.webkitOfflineAudioContext
navigator.userAgent
navigator.language
navigator.userLanguage
navigator.browserLanguage
navigator.systemLanguage
window.screen.colorDepth
navigator.deviceMemory
window.devicePixelRatio
window.screen.width
window.screen.height
window.screen.availWidth
window.screen.availHeight
window.Intl
window.Intl.DateTimeFormat
window.sessionStorage
window.localStorage
window.indexedDB
window.openDatabase
navigator.cpuClass
navigator.platform
navigator.doNotTrack
navigator.msDoNotTrack
window.doNotTrack
window.swfobject
navigator.plugins
navigator.maxTouchPoints
navigator.hardwareConcurrency
```

REFERENCES

[1] 2008. The 41st Parameter: PCPrint. (2008). http://www.the41st.com/land/DeviceID.asp

[2] 2010. content scripts that want to hijack Javascript functions in the main browser page. https://groups.google.com/a/chromium.org/forum/#!topic/chromium-extensions/LCZCkraIBdg. (2010). Accessed: 2018-11-28.

[3] 2018. chrome.notifications. https://developer.chrome.com/apps/notifications. (2018). Accessed: 2010-12-12.

[4] 2018. chrome.runtime. https://developer.chrome.com/extensions/runtime. (2018). Accessed: 2018-12-12.

[5] 2018. chrome.tabs. https://developer.chrome.com/extensions/tabs. (2018). Accessed: 2018-12-13.

[6] 2018. Intelligent Tracking Prevention 2.0. (2018). https://webkit.org/blog/8311/intelligent-tracking-prevention-2-0/

[7] 2018. Your Apps Know Where You Were Last Night, and They're Not Keeping It Secret. (Dec 2018). https://www.nytimes.com/interactive/2018/12/10/business/location-data-privacy-apps.html

[8] Alessandro Acquisti. 2009. Nudging privacy: The behavioral economics of personal information. *IEEE security & privacy* 7, 6 (2009).

[9] Danah Boyd and Eszter Hargittai. 2010. Facebook privacy settings: Who cares? *First Monday* 15, 8 (2010). https://doi.org/10.5210/fm.v15i8.3086

[10] Matt Cagle and Nicole Ozer. 2018. Amazon Teams Up With Government to Deploy Dangerous New Facial Recognition Technology. (2018). https://www.aclu.org/blog/privacy-technology/surveillance-technologies/amazon-teams-government-deploy-dangerous-new

[11] Peter Eckersley. 2010. How Unique Is Your Web Browser?. In *Privacy Enhancing Technologies*, Mikhail J. Atallah and Nicholas J. Hopper (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–18.

[12] Steven Englehardt and Arvind Narayanan. 2016. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security.* ACM, 1388–1401.

[13] Stefan Etienne. 2018. Taylor Swift tracked stalkers with facial recognition tech at her concert. (Dec 2018). https://www.theverge.com/2018/12/12/18137984/taylor-swift-facial-recognition-tech-concert-attendees-stalkers

[14] Edward W Felten and Michael A Schneider. 2000. Timing attacks on web privacy. In *Proceedings of the 7th ACM conference on Computer and communications security.* ACM, 25–32.

[15] Richard J Gray and Dominic V Bennett. 2008. Method for Detecting and Preventing Fraudulent Internet Advertising Activity. (Nov. 20 2008). US Patent App. 11/688,160.

[16] Saikat Guha, Bin Cheng, and Paul Francis. 2011. Privad: Practical privacy in online advertising. In *USENIX conference on Networked systems design and implementation.* 169–182.

[17] Danish Jamil and Muhammad Numan Ali Khan. 2011. Keystroke pattern recognition preventing online fraud. *International Journal of Engineering Science and Technology (IJEST)* 3, 3 (2011), 1953–1958.

[18] Balachander Krishnamurthy, Delfina Malandrino, and Craig E Wills. 2007. Measuring privacy loss and the impact of privacy protection in web browsing. In *Proceedings of the 3rd symposium on Usable privacy and security.* ACM, 52–63.

[19] Sangho Lee, Youngsok Kim, Jangwoo Kim, and Jong Kim. 2014. Stealing webpages rendered on your browser by exploiting GPU vulnerabilities. In *2014 IEEE Symposium on Security and Privacy (SP).* IEEE, 19–33.

[20] Pedro Leon, Blase Ur, Richard Shay, Yang Wang, Rebecca Balebako, and Lorrie Cranor. 2012. Why Johnny can't opt out: a usability evaluation of tools to limit online behavioral advertising. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* ACM, 589–598.

[21] Pedro Giovanni Leon, Justin Cranshaw, Lorrie Faith Cranor, Jim Graves, Manoj Hastak, Blase Ur, and Guzi Xu. 2012. What do online behavioral advertising privacy disclosures communicate to users?. In *Proceedings of the 2012 ACM workshop on Privacy in the electronic society.* ACM, 19–30.

[22] Delfina Malandrino, Vittorio Scarano, and Raffaele Spinelli. 2013. How increased awareness can impact attitudes and behaviors toward online privacy protection. In *Social Computing (SocialCom), 2013 International Conference on.* IEEE, 57–62.

[23] J. R. Mayer and J. C. Mitchell. 2012. Third-Party Web Tracking: Policy and Technology. In *2012 IEEE Symposium on Security and Privacy.* 413–427. https://doi.org/10.1109/SP.2012.47

[24] Aleecia M McDonald and Lorrie Faith Cranor. 2008. The cost of reading privacy policies. *ISJLP* 4 (2008), 543.

[25] Martin Mulazzani, Philipp Reschl, Markus Huber, Manuel Leithner, Sebastian Schrittwieser, Edgar Weippl, and FC Wien. 2013. Fast and reliable browser identification with javascript engine fingerprinting. In *Web 2.0 Workshop on Security and Privacy (W2SP)*, Vol. 5. Citeseer.

[26] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. 2013. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Security and privacy (SP), 2013 IEEE symposium on.* IEEE, 541–555.

[27] Lukasz Olejnik, Claude Castelluccia, and Artur Janc. 2012. Why johnny can't browse in peace: On the uniqueness of web browsing history patterns. In *5th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2012).*

[28] Leysia Palen and Paul Dourish. 2003. Unpacking privacy for a networked world. In *Proceedings of the SIGCHI conference on Human factors in computing systems.* ACM, 129–136.

[29] Stefanie Pötzsch. 2008. Privacy awareness: A means to solve the privacy paradox?. In *IFIP Summer School on the Future of Identity in the Information Society.* Springer, 226–236.

[30] Irwin Reyes, Primal Wijesekera, Joel Reardon, Amit Elazari Bar On, Abbas Razaghpanah, Narseo Vallina-Rodriguez, and Serge Egelman. 2018. "Won't Somebody Think of the Children?" Examining COPPA Compliance at Scale. *Proceedings on Privacy Enhancing Technologies* 2018, 3 (2018), 63–83.

[31] Alexey Reznichenko, Saikat Guha, and Paul Francis. 2011. Auctions in do-not-track compliant internet advertising. In *Proceedings of the 18th ACM conference on Computer and communications security.* ACM, 667–676.

[32] Florian Schaub, Aditya Marella, Pranshu Kalvani, Blase Ur, Chao Pan, Emily Forney, and Lorrie Faith Cranor. 2016. Watching them watching me: Browser extensions' impact on user privacy awareness and concern. In *NDSS Workshop on Usable Security.*

[33] Jaebaek Seo, Daehyeok Kim, Donghyun Cho, Insik Shin, and Taesoo Kim. 2016. FLEXDROID: Enforcing In-App Privilege Separation in Android.. In *NDSS.*

[34] Jacob Snow. 2018. Amazons Disturbing Plan to Add Face Surveillance to Your Front Door. (Dec 2018). https://www.aclu.org/blog/privacy-technology/surveillance-technologies/amazons-disturbing-plan-add-face-surveillance-yo-0

[35] Christof Ferreira Torres, Hugo Jonker, and Sjouke Mauw. 2015. FP-block: Usable web privacy by controlling browser fingerprinting. In *European Symposium on Research in Computer Security.* Springer, 3–19.

[36] Joseph Turow, Jennifer King, Chris Jay Hoofnagle, Amy Bleakley, and Michael Hennessy. 2009. Americans reject tailored advertising and three activities that enable it. (2009).

[37] Blase Ur, Pedro Giovanni Leon, Lorrie Faith Cranor, Richard Shay, and Yang Wang. 2012. Smart, useful, scary, creepy: perceptions of online behavioral advertising. In *proceedings of the eighth symposium on usable privacy and security.* ACM, 4.

[38] Valve. 2018. fingerprintjs2. (2018). https://github.com/Valve/fingerprintjs2

[39] Yang Wang, Pedro Giovanni Leon, Alessandro Acquisti, Lorrie Faith Cranor, Alain Forget, and Norman Sadeh. 2014. A field trial of privacy nudges for facebook. In *Proceedings of the SIGCHI conference on human factors in computing systems.* ACM, 2367–2376.

[40] Tal Z Zarsky. 2016. Incompatible: The GDPR in the Age of Big Data. *Seton Hall L. Rev.* 47 (2016), 995.