

# Internet Anonymity

**Submitted by:**  
Group 12

Mike Hoffert - mlh374

Jeff Pereyma - jdp037

Kari Vass - kdv504

Nathan Abramyk - nsa901

**Date:**  
March 28, 2014

## Abstract

Internet anonymity is important because it can protect individuals from oppressive censorship and those in positions where tying an identity to their online presence could cause harm to come to them. This project focused on one particular method of undermining anonymity: the browser fingerprint. Browsers are fingerprinted by gathering information about the browser. The combination of this information is often reasonably unique, and can thus be used to track that browser.

Our project was to undermine the harvesting of the types of information which are typically used to create the browser fingerprint. In particular, we found that preventing enumeration (and mass detection) of fonts and plugins to help generalize the browser. Setting the HTTP language headers to be as general as possible also helps reduce fingerprinting. The Panopticlick tool, provided by the EFF, was used in gauging the effects of our project.<sup>1</sup>

## Contents

<b>1</b>	<b>Genesis</b>	<b>1</b>
1.1	Overview . . . . .	1
1.2	Approach and early planning . . . . .	1
1.3	Pre-conceptions . . . . .	1
<b>2</b>	<b>Initialization</b>	<b>2</b>
2.1	Ideas . . . . .	2
2.2	Goals . . . . .	2
2.3	Obstacles . . . . .	3
2.4	Process . . . . .	3
<b>3</b>	<b>Control</b>	<b>3</b>
3.1	Documentation . . . . .	3
3.2	Planning . . . . .	4
3.3	Data gathering . . . . .	4
3.4	Analysis . . . . .	5
<b>4</b>	<b>Technical components</b>	<b>5</b>
4.1	Approaches and methods . . . . .	5
4.2	Progress and effort . . . . .	5
4.3	Difficulties and limitations . . . . .	5
<b>5</b>	<b>Results</b>	<b>6</b>
5.1	Results and outcomes . . . . .	6
5.2	Progress and failures . . . . .	6
5.3	Analysis . . . . .	6
<b>6</b>	<b>Recommendations for further study</b>	<b>6</b>

# 1 Genesis

## 1.1 Overview

Internet anonymity refers to the ability to remain anonymous on the internet, often behind a pseudonym. Internet anonymity helps support freedom of speech: governments can't censor you if they don't know who you are. Skirting surveillance, however, generally requires anonymity. Oppressive governments often attempt censorship. For example, Turkey recently blocked Twitter and Youtube, Amnesty International stated that China "has the largest recorded number of imprisoned journalists and cyber-dissidents in the world", and in Iran, internet users must promise not to access "non-Islamic" websites.

Further, whistleblowers may need anonymity to prevent retaliation, undercover military and law enforcement agents often need anonymity for their protection, and journalists may have to protect their sources. Internet anonymity is also a line of defence against targeted attacks (it's difficult to target someone whom you cannot identify).

Internet anonymity, however, is often under attack. Some modern threats to internet anonymity which are relevant in our country include how IP addresses can be tied to an ISP customer (but court rulings have found that IP addresses are insufficient to identify an individual), a number of political acts, both in the House of Commons and in Congress (and since many web sites are American, changes in American law affect Canadians, too), and tracking services (especially with online advertisers).

## 1.2 Approach and early planning

One lesser known threat to internet anonymity is browser fingerprinting, in which information about the browser and computer are combined to form a digital fingerprint. That is, information such as the list of installed fonts and plugins, screen resolution, time zone, and several others can be combined, and the result is often reasonably unique.

We chose to study this area because there has been fairly little research into browser fingerprinting. Our preliminary findings were that some web browsers are very susceptible to fingerprinting. This was discovered by using the Panopticlick<sup>1</sup> tool, provided by the EFF. This tool collects data from the browser and uses that data to estimate the uniqueness of the browser. In early testing, we found that our browsers were often highly unique, and thus, easily tracked.

For our project, we decided to minimize the degree to which the browser can be tracked via its fingerprint. That is, we desired to make the browser as generic as possible (from the viewpoint of querying websites).

**TODO:** Should we mention the other ideas that we considered (eg, Tor exit node analysis) here? I'd like to, but I'm having a hard time fitting it into the text in a fluid way.

## 1.3 Pre-conceptions

At the early stages of our project, it was unclear what restrictions we would face in preventing fingerprinting. It was decided that we would develop a browser extension, and chose to do so for the Chrome browser because it (1) was considered to be easier to develop extensions for and (2) allowed enumeration of plugins, which provided an

additional vulnerability against fingerprinting (and thus being a more optimal choice for studying how to thwart fingerprinting techniques). For comparison, the Firefox browser does not allow enumeration of plugins, but was still vulnerable to fingerprinting because it allowed mass querying of plugin data.

The development of a Chrome extension, however, lead to several pre-conceptions. In particular, we assumed that Chrome extensions could inject JavaScript into webpages, and that this injected JavaScript can modify JavaScript on that webpage. Traditionally, JavaScript located on a webpage exists in the same scope, meaning that you could have a variable in one script that is referenced in another.

This was a crucial assumption, as plugins are accessed via the `window.navigator` object. In JavaScript, it is possible to overwrite objects. If an injected script was run in the same scope, it would be able to modify the navigator object for the other scripts on the page.

We also had pre-conceptions regarding font detection. It was initially assumed that fonts were detected with JavaScript, but while it turns out that JavaScript does not provide such a functionality. We did, however, discover a means of detecting fonts with JavaScript by means of setting the font of a field of known proportions and checking if the proportions have changed. This unorthodox technique hindered our ability to prevent all types of fingerprinting.

## 2 Initialization

### 2.1 Ideas

The initial idea for our project was to construct our Chrome extension in a way that would detect when fingerprinting is possibly occurring (ie, when sites attempt to access too much information) and let the user decide whether or not to allow this action. To use an analogy, our extension would work akin to an anti-virus that notices suspicious activity and asks the user if they want to allow that program to run.

In our testings, Panoptick showed three major areas that made the browser the most identifiable. The HTTP language headers were sending en-CA (Canadian English) as an acceptable language, which was unnecessary because they already accepted en-US (which is sufficient for the virtually all websites). The list of installed fonts was strongly identifying. It appeared that programs often install fonts, which makes for a fairly unique font list. Finally, the list of installed plugins, which contained plugin names and versions, was approximately as strongly identifying as the font list.

We would also create a whitelist for our extension, which was a list of domains for which our extension would not be active on. This would allow sites to perform tasks that our extension blocks, provided that they had explicit user permission (as is necessary to add the domain to the whitelist).

### 2.2 Goals

The general goal of our project was to study and understand how browser fingerprinting works and its implications on internet anonymity. Our practical goal, however, was to detect fingerprinting behavior and alert the users to it. We believe that websites often

take liberties of users not being aware of the site’s actions, and desired to make those actions transparent via our extension.

TODO: Should we mention the revised goals here or later? ie, the fact we couldn’t alert the user and sites sometimes silently fail? I’m thinking just mention it later, so that the report reads more linearly.

## 2.3 Obstacles

The biggest obstacle that we had to face was the fact that we were treading into very new ground. Not only did we have to work with languages, frameworks, APIs, and fields that some of us had little or no experience in, but we were also largely doing what we believe to be a novel project. To the extent of the authors, there are no other extensions or programs that attempt to thwart browser fingerprinting. This reduced the number of resources we could draw from.

One major obstacle that we encountered mid-way through our project’s development was the fact that Chrome sandboxes its extensions. Chrome’s sandboxing means that while a content script is “injected” into a webpage, the script is run in a sandbox, separately from the other scripts on the page. The script can modify the page’s DOM, but it exists outside of the scope of other scripts. This was a contradiction to our pre-conceptions, where we assumed that we’d be able to modify JavaScript variables for other scripts.

Thankfully, we found a solution to get around Chrome’s sandboxing, namely by inserting a script element into the DOM of the page, containing the injected script that had to be run in the same scope as the rest of the page. Thus, from this injected code, we were able to globally modify the `window.navigator` object, preventing plugin enumeration.

## 2.4 Process

TODO: From an SE perspective. I guess it was mostly “concurrent engineering”. Not much of a formal process... anyone have a better description?

# 3 Control

## 3.1 Documentation

The primary “documentation” of our project is the source code of our created Chrome extension, which illustrates the techniques we used in preventing access to fingerprinting information. The source code of our project is freely available on GitHub.<sup>2</sup>

The project is also largely results based, with Panopticlick being used as the driving force in interpreting the effects of our extension. The results of our finished extension are mentioned in section 5.1.

TODO: I feel this is rather sparse, or perhaps misunderstood? Anyone have better ideas for this section?

## 3.2 Planning

TODO: Mention how we identified core tasks (plugins, fonts, etc) and divided them up. Also some sharing of tasks, meetings, and so on.

## 3.3 Data gathering

Panopticlick was the dominant method of evaluating the effects of our extension. The tool calculated two useful numbers: an estimated bits of entropy provided by some identifying factor (eg, installed fonts, time zone, HTTP accept headers) and an estimated “one in  $x$  browsers has this value”. Obviously a more generic browser is more difficult to fingerprint (as you can’t tell it apart from thousands of others), and thus, we want to minimize the value for how many browsers have a given value.

The fields that Panopticlick collected data for were:

- User agent
- HTTP\_ACCEPT headers
- Browser plugin details
- Time zone
- Screen size and color depth
- Installed fonts
- Are cookies enabled?
- Limited supercookie test

Of these, we determined that only the HTTP\_ACCEPT headers, browser plugin details, and installed fonts were significantly unique on the browsers which we tested. The other values were very common, with “one in  $x$  browsers has this value” numbers ranging from 1.35 for the supercookie test to 1547.31 for the user agent. We determined that these factors could not be reasonably improved on. The user agent number is likely biased, since the user agent contains the browser version, which frequently changes (and Panopticlick has stored many older versions of the user agent, which changes frequently).

Further, some of these fields are not practical to change. Changing the user agent could mess up sites that use the user agent to display browser unique content, while modifying the screen size could cause positioning issues for sites that dynamically position elements based on the screen size.

The HTTP\_ACCEPT headers, browser plugin details, and installed fonts fields, however, has “one in  $x$  browsers has this value” numbers ranging from 22,124.09 for the HTTP\_ACCEPT headers to 1,334,820 for installed fonts. From this, we can almost identify the browser from the fonts alone (Panopticlick only has a sample size of approximately 4 million).

Thus, Panopticlick was the main tool we used in collecting data and determining the effect of our efforts. In particular, our goal was to minimize the “one in  $x$  browsers has this value” number, painting the browser as generic as possible.

### 3.4 Analysis

Panopticlick proved to be a formidable tool, allowing us to establish a control group (the unmodified browser) versus our experimental group (the browser with our extension enabled), and compare the change in “browser generality”. This required that we make the assumption that a unique browser is easily identified and thus easily fingerprinted, while a common browser “blends in” with other browsers, making it difficult to track by fingerprinting alone.

In our implementation, we realized that JavaScript conventions typically stored information as object properties, as opposed to using getters and setters, as with other languages that the authors were used to. Because of this and the lack of visibility modifiers in JavaScript, combined with time constraints, the authors initially erroneously assumed that we could only remove JavaScript properties and could not detect their access.

This was later found to be incorrect, and in fact JavaScript defines an approach to creating automatic getters and setters for accessing and setting object properties. It was due to this error that the project was ultimately unsuccessful in alerting the user to fingerprinting techniques and instead opted to silently block access to the identifying information. A more rigid implementation could take advantage of JavaScript’s ability to define getters in such a way that the implementation could keep track of and only block sites that attempt to mass collect such information.

For the purpose of this research project, however, we believe that our implementation is sufficiently useful, particularly as a proof-of-concept. A practical implementation could expand on this using the previously outlined techniques to create a more user-friendly experience.

TODO: I feel that we can expand on our analysis here. The focus should be a meta-analysis of the previous sections.

## 4 Technical components

### 4.1 Approaches and methods

TODO: How did we approach the project. Code, browser, extension, etc. The tools we used. Mention the use of panopticlick as a driving tool, particularly in checking our results.

### 4.2 Progress and effort

TODO: How many of our goals did we meet? What places failed but we applied effort on? Particular parts we’re proud of and are central to project, etc?

### 4.3 Difficulties and limitations

TODO: Hard parts, stuff we couldn’t do, and limitations (particularly those created by Chrome and JS).

## 5 Results

### 5.1 Results and outcomes

TODO: Put the numbers from the slides here. Also mention the overall effectiveness and other approaches.

### 5.2 Progress and failures

TODO: Progress stuff from previous section, with emphasis on the implications our progress had on the results. And failures are straightforward enough (in particular, inability to truly let the user know when fingerprinting is going on. Adding sites to the whitelist is entirely up to the user, we couldn't provide advice. As a result, some sites fail silently. I, for example, cannot load gmail's inbox without adding the site to the whitelist.

### 5.3 Analysis

TODO: General summary of results. Perhaps deeper look at other areas? Ideas here?

## 6 Recommendations for further study

TODO: Recommendations for how browsers could implement our functionality (but better). Chrome shouldn't allow plugin enumeration, and should warn users if too many plugin details are collected. Also, Flash shouldn't be able to enumerate fonts without user permission. Basically, ask the user more.

TODO: Read through this document VERY carefully and find more citations for dubious claims.

## References

- [1] Panopticlick: <https://panopticlick.eff.org/>
- [2] Fingerprint Anonymizer repository on GitHub: <https://github.com/MikeHoffert/FingerprintAnonymizer>