University of Piraeus – Department of Informatics

Postgraduate Program

«Advanced Information Systems»

Master Thesis

Thesis Title	**Browser fingerprinting and countermeasures**
Student Name	**Evangelos Deirmentzoglou**
Father's Name	**Andreas Deirmentzoglou**
Student ID	**MPSP/13027**
Supervisor	**Constantinos Patsakis**

Submission Date **June 2015**

Thesis Jury

Constantinos Patsakis Lecturer	Panayiotis Kotzanikolaou Lecturer	Efthimios Alepis Lecturer

Abstract

Internet is generally considered an anonymous place where people can communicate, interact and exchange goods and ideas. Certainly, this is a common misconception and there is a lot of research focusing on the providing users with their fundamental right to privacy. Many companies have huge monetary interest in removing the veil of anonymity as for instance they are interested in targeted advertisement. Therefore, they are constantly trying to trace users traffic and depending on the web pages they visit create a profile. Tracing which websites a specific IP is visiting is does provide an inaccurate measurement as many users could be using the same IP. To counter this problem browser fingerprinting techniques have been deployed allowing someone to trace a user from his browser, which means that she can be traced even if her IP is changed. In this work we discuss current state of the art in browser fingerprinting and propose measures to combat many of them. Additionally, we present Brownymity an experimental tool we have developed to illustrate the efficacy of our methodologies.

Table of Contents

Introduction

Users navigating to the internet everyday are facing different kinds of advertisement. Each of them has his own advertisement model which will be exposed to and most likely that model will be adaptable to his likings. It's quite possible that most of the ads presented will be future or current purchases of his. Sometimes a user might wonder how come all the ads presented to her were exactly what she wanted. The answer to that is clear and it's called user tracking.

From the beginning of the advertisement world every ad company was trying to show advertisements to potential future customers. At first marketing was being conducted by exposing the same ad to huge masses of people like the television or a certain part of the town. This tactic was costly and messy, no one can predict the needs of a customer only by a show he's watching or by living on a specific neighborhood. It will certainly attract some customers but not all of them.

As time passed by ads became more personalized as the technology around us became more advanced. Today most if not all of the internet users have been a victim of user-tracking. A user can surf the internet and all of her actions will be recorded, all the clicks made will be noted and all the web history will be registered in a database somewhere around the globe for further processing. Searching the web for a particular product will force that product to be the main ad of most of the services you currently use and it will stuck there for weeks.

21st century advertisement is so personalized that sometimes it can be creepy. One of the best examples of today's advertisement is the Target pregnancy case. In order to increase sales most of the online services will perform data mining techniques to all the data they have gathered about their users and try to make sense of it. Data derived from such processes can lead to future purchases, more accurate advertisement and thus an increase in their revenue. As such if you search on Amazon for a series of products, like camping gear it will let Amazon to believe that you are planning to go for camping and it will start presenting you with the most popular to your needs camping equipment.

This happened to a young girl who was searching Target about pregnancy related stuff. It came out that according to her searches Target's data mining results believed that she was pregnant. Target had a new program running at that time helping parents to be to find the equipment they needed in lower prices. As a result Target sent a few coupons to that teenager for baby clothes and cribs. That mail infuriated her father and made complains to Target. Once Target tried to apologize to her father he responded that his daughter was indeed pregnant and she was due in August. This is just an example of how personal and creepy advertisement can be. Since then some precautions are being made from companies like Target in their advertisement.

Ads should be personal because if they weren't they would be a waste of capital and ad companies wouldn't be profitable but they can't be too personal, at least we shouldn't believe they are. Online advertisement has pushed a lot the research in user tracking, and in combination with some other suspects who will be presented below have developed tools and techniques which make it nearly impossible to opt-out from them. As it will be discussed in this research every move we're making in the web is leaving a trail of footprints which are easily

connected together and will lead back to us. Trying to avoid a detection by using various tools will not secure our anonymity and it is quite possible that it will make us more unique than before.

Currently, user tracking methods are widely being adopted by many websites. The reasons behind this trend may vary from service improvement; as the service provider wants to know how users navigate to his website, how they respond to content and changes etc., to user profiling, where the web page tries to monetize its visibility by selling usage statistics and user preferences to e.g. advertising companies or detect whether the user has recently visited the site of a competitor. Regardless of the motive of the service provider, these methods introduce many privacy issues as the browsing history reveals a lot of sensitive information and can lead to the identification of individuals.

While the most widely used method to track users is with cookies, webpages may also use third parties' services to learn about their users [1] enabling them to add advanced features such as advertising, analytics, social network integration. Features like that is possible that they will try to acquire information by the user and sent it to another service like Facebook or Google.

By including a third party service your website will probably reveal your information to that service as well. This might not affect you right away but in the long run it'll certainly do. Imagine using 15-20 services everyday where all of them are using Google Analytics and you are connected to them through Facebook. It is quite possible that both of them will know that you visited those sites and for how long, hopefully they'll only know that much. The fun fact about these examples is that those services won't even need to perform any fingerprinting technique. They already know who you are. As for all the other services they might need to perform some cool tricks, some of which will be presented in this research.

Most of the fingerprinting techniques used in today's web wouldn't be feasible 10-15 years ago. This is true due to the fact that the majority of the exploited features are fairly new like canvas in HTML5 or the computing power and bandwidth needed for these tricks would be easily noticeable. User tracking existed back then but it wasn't so robust and accurate, it was mostly done by single cookies and they were mostly used for carrying sessions.

Currently, there is a growing increase towards browser fingerprinting methods. These methods exploit the information that is provided by browsers such as installed fonts, screen resolution and available plugins, to create an identifier for the browser. What is important about these fingerprints is that even if the fingerprint changes (i.e. the user updates his software), it is fairly easy to find from which fingerprint it was derived. Some of these techniques have been recently discussed [2] [3] [4] [5] and several countermeasures have been proposed [4] [6] trying to prevent this activity, but none of them seems to be working in full capacity.

A browser fingerprint is a series of bread-crumbs which will uniquely identify a user browser. Gathering all that information about the user's browser can generate a lot of information about him. A user can be recognized along various websites without having a session in any of them or any cookies in general. No sessions, no cookies, no local storage data. Even if the user select to clear the browser history and delete its cache he will still be recognized as a unique user.

Taking it a step further, let's say that a user decides to format his system and make a clean install of his favorite operating system. This will force his current fingerprint to stop developing and will create a second one. It is very important to note that the first fingerprint will not disappear but it will simply stop developing, meaning that it will still exist for processing. Now the second fingerprint will never be the same as the first one because this is nearly impossible to accomplish but it will behave in the same way as the first one. The user might clean his system but he will continue to operate in the web with the same habits e.g. every morning he will visit google to search for his favorite newspaper and shortly after he will open Spotify to listen to his favorite music. As his habits are the same, the models for the first fingerprint and the second will be tied as one and eventually the user will be identified again as the one with the first fingerprint. It is redundant to say that connecting your devices with the major providers like Google will tie your fingerprint immediately with the ones from your other devices.

What's of equal interest as a research topic with fingerprint recreation which was discussed above is fingerprint evolvement. A fingerprint in the digital world derived by the web browser is not static but dynamic. The browser elements keep evolving and a power user might evolve his fingerprint very fast. In most cases the fingerprint evolves in a linear way but yet again it's fairly difficult to track it and merge it with another fingerprint. This being more of a data mining problem it's very difficult to identify two different fingerprints as one being evolved or distinguishing them as two completely different fingerprints.

In Panopticlick [3], researchers collected the following information: User Agent strings, other browser headers, cookie and supercookie blocking status, timezone, screen size, browser plugins (with their versions) and system fonts from approximately 1 million users. Then, using this dataset they tried to see how random this dataset was. From their findings, 94.2% of the browsers had more than 18.8 bits of entropy, which means that someone could easily be identified. More interestingly, the elements of a browser with the highest entropy are plugins, system fonts and user agent with 15.4, 13.9 and 10 bits of entropy respectively. While the above research was mainly used as a proof-of-concept, it has been shown that this has been used widely on the Internet [7] [2].

This research consists just the tip of the iceberg as a lot more elements and ways to fingerprint a web browser have been discovered. The most sophisticated of those will render an invisible image with a text or a 3D graphic and then target site will try to read the canvas and find information about the system. Rendering graphics will make use of the GPU and consist a unique fingerprint for the browser and the system itself. Another way to fingerprint a user is by enumerating its fonts. This can be done by displaying an invisible text on the user and then calculating the size of each letter displayed and since each font has its own unique measurements a list of installed fonts is created. These methods are canvas fingerprinting and font enumeration respectively.

Apparently, by minimizing the entropy of the above elements, one would make the browser fingerprint less unique and provide users more anonymization. In this regard, we introduce *Brownymity*, a tool for mitigating browser fingerprint, without breaking the usability of the target website. The concept of Brownymity is to create a pool of commonly used browser attributes

and then provide the most compatible ones to the clients who requests them. Therefore, a server keeps track of ``browser profiles'' and recommends some profiles that are used by others having obfuscated the font list, the list of plugins, user-agents and window information. Therefore, Brownymity does not learn anything about the sites that the user is visiting and provides him with a $k-$ anonymous ``browser profile''.

The rest of the paper is organized as follows. In Section 2 we provide an overview of current literature on web tracking and browser fingerprinting. Then, Section 3 introduces Brownymity, its architecture and features. Section 4 discusses the applicability and efficacy of the tool and compares it to other solutions. Finally, Section 5 concludes the article and proposes some thoughts, ideas about future work. The source code for the Brownymity will be listed in the Appendix.

Related Work

While cookies can easily be removed, it was shown that they could be recovered using the so-called *respawning* method, either using Adobe Flash [8], or using ETags and the HTML5 localStorage API [9].

What is more impressive is that even a passive network observer could use third-party HTTP tracking cookies to identify users [10].

Canvas fingerprinting is a technique introduced by Mowery and Shacham [11]. Depending on the underlying operating system, font library, graphics card, graphics driver and browser, a text or an image is rendered differently. Therefore, an attacker would write an invisible text and display a hidden image in order to derive a browser fingerprint. The importance of fingerprinting through font metrics was further studied recently indicating that they can offer more information than User-Agent strings, and that they can be used to uniquely identify or significantly narrow down the search for individual users [12]. Canvas fingerprinting was introduced with HTML 5 and till today there isn't an efficient way to counter it.

It has also been proven that the browser's JavaScript execution characteristics can be used to fingerprint devices [13]. JavaScript may disclose browser version, operating system and microarchitecture or by checking whether particular domains exist in a user's NoScript[1] whitelist. By timing browser performance on different operations of the JavaScript language browser version can be distinguished and microarchitectural features. Due to the architecture of each browser and version timing their performance can derive fruitful results.

A lot of work has been done for device fingerprinting and especially for smartphones [14] [15] [16]. In those works it was demonstrated that a unique fingerprint can be derived by the use of accelerometers, speakers and microphones through inaudible sounds. An attacker can theoretically force a device to play sounds via its speakers and record them using the microphone. An analysis on the captured data can provide the attacker with enough information to render a device unique. This is due to the fact that every device's components are unique in their own way due to damage, time decay, misconfigurations of the component and factory assembling. This leads to the conclusion that even a mass production of a certain product can lead to uniquely distinguishable devices.

Firegloves [6] browser extension returns randomized values when queried for certain browser attributes. However since the same attributes can be retrieved via different browser APIs, the users of Firegloves become more uniquely identifiable than users who do not install this extension. On top of that Firegloves would allow each tab to load only a certain number of fonts reducing that way the amount of identifiable information a browser would give to a service. Unfortunately the project got discontinued.

[1] https://noscript.net/

Mor et al. introduced the concept of Bloom cookies, a special type of cookies, which as the name suggests, exploits the features of Bloom filters and is used by the client every time he makes a service request [17]. The user can easily inject bits in the filter to obfuscate his profile, however, the size of the exchanged cookie is quite small.

Privad [18] a tool made based on security concerns of existing advertising tools uses a semi-trusted proxy called the dealer to which it sends limited information about the user profile and discard irrelevant results. This tool requires a specific behavior not only by the user but from the server as well. Modifying the existing plan of advertisement companies into taking into consideration a third party like the semi-trusted proxy will not work very well at least any time soon.

Besson et al. formalize the problem web tracking and fingerprinting through information theory and propose a randomization mechanism for the user configurations which can provide him specific privacy guarantees [19]. This work consists the first appearance of the *k-anonymity* and *e-differentially* concepts by the term *t-privacy* in the browser-based fingerprinting literature. Even though they investigate the distinctively between browsers their research does make a lot more sense if used against one single browser. Trying to change the settings of a Firefox browser to match an Opera will definitely create a lot of problems but if you try and make one Firefox user to match another Firefox user then the possibilities of this happening have increased by a tenfold.

Fiore et al. [20] have followed a similar trail of thought as ours, allowing the user to create fake Chrome profiles and chose which one to use each time. Fiore's et al. approach could be considered only as theoretical since they created a Chrome extension which provides the user with hardcoded settings and there is no communication with the proxy server or even an implementation of that server thus their timing results are misleading. The proposal of having a proxy on the other hand which can control the amount of information provided to a server is indeed significant. As it is going to be discussed, most of these limitations are countered from our solution.

Pan et al. have recently introduced a new web browser called TrackingFree which isolates unique identifiers into different browser principals managing to resist tracking from all the 647 trackers found in Alexa Top 500 web sites [21]. TrackingFree consists of a Chromium's modified version.

Acar et al. [5] presented in their study the first real-world study of canvas fingerprinting and discovered that this method is the most common technique used for tracking users in the web. Alongside to this contribution their study reveals solid information that evercookies and cookie respawning techniques are still being used in the wild. As outlined in their work the most responsible factor for user fingerprinting in the web via cookies is Flash. Most of these techniques can be performed with the current state of Flash and in order to avoid it the user must disable Flash altogether. Regarding the effect of opt-out only 5% reduction in the number of participants was observed at the cookie syncing technique while for techniques like canvas fingerprinting and cookie reviving the results were unmodified.

In another work of Acar et al. [7] a new framework called FPDetective was developed specially designed for web privacy studies. The primary focus of this framework is font detection. FPDetective consists of a crawler which will visit various sites and it will keep logs of any website trying to load more than a handful fonts, accessing specific browser properties or even trying to transfer Flash objects. For this purpose a proxy was used intercepting suspicious packets and decompiling Flash objects. All of this information is later stored in a database for further analysis.

Mayer and Mitchell [1] expressed their concerns on third party tracking and its privacy implications in a document with vast information ranging from theoretical to technical, from legislative to moral and from constructive advertisement to destructive. Apart from that a research tool was developed, the FourthParty tool, a FireFox extension whose purpose was analysing traffic from third party sites, mainly advertisements.

Nikiforakis et al. [4] in his work about PriVaricator developed a new web browser from a fork of Chromium which could mislead fingerprinters by providing false font measurements. Designed to spoof font measurements the specially crafted browser will have the functions *offsetHeight*, *offsetWidth* and *getBoundingClientRect* modified into providing one of the three following values. They will either result into zero, a random value between 0 and 100 or a variation of +-5% of the initial value. All of these options could disrupt font enumeration but other problems arise. By providing only the zero value as a result for those functions would indeed eliminate font enumeration but it would give the information of using PriVaricator to the attacker since this is a very distinctive characteristic of this browser. Apart from that this technique and the random output between the values 0 and 100 often resulted to website breakage. Adding noise to the result eliminates font enumeration and it's still reliable as a fingerprint which means that the attacker won't find out anytime soon that he's being spoofed and the breakage of the website is in acceptable levels.

In another work of Nikiforakis et al. [2] an extensive research on browser fingerprinting was presented. By presenting commercial fingerprinting the fact that advertisement usually ignores ethics and takes advantage of the web vulnerabilities is undeniable. Nikiforakis et al. will go in depth presenting all the web fingerprinting methods used wildly in the web from font detection to system fingerprinting plugins. Most importantly of all their work introduced the notion that the more you try to avoid fingerprinting by using third party mechanisms like plugins, extensions or system proxies the more you are being identified as a unique user. This can happen due to the incompleteness of the tools used, the breakage or the unique fingerprints they leave at the system.

The Brownymity Tool

As already discussed, Eckersley [3] through Panopticlick[2] found that the browser characteristics that had the most randomness were the user agent, and the lists of fonts and plugins. Following this trail of thought, we argue that by making this set less random, the browser is more anonymous. The question that arises from this assumption is what this less random browser fingerprint is going to be and whether this fingerprint is going to compromise user experience. In the following paragraphs, we present Brownymity and its architecture, a research tool that has been developed to resolve these issues.

Main actors and desiderata

In our model we have three main entities, the User, the WebPage and the Pool of Profiles (PoP). Initially, the user wants to visit WebPage, but he wants to remove as much traces as possible from his browser as he knows that WebPage applies several methods for browser fingerprinting. To counter this threat, the User forwards his browser fingerprint to PoP, a semi-trusted server which takes as input the browser fingerprint and outputs a compatible one with less randomization, or one that has been used several times before. The provided fingerprint bears many characteristics in common with the original and is then used by the User's browser to visit WebPage. On receiving the browser fingerprint, WebPage queries her database and finds at least $k-1$ more instances of the same fingerprint so she cannot tell whether the fingerprint belongs to a new browser or not.

In our model we assume that PoP does not collude with WebPage; telling him when User contacted him and what profile was given to him; and does not behave maliciously; it does not provide identifiable fingerprints. The general overview of Brownymity's architecture is illustrated in Figure 1.

Functionality

Once the user navigates to a website the chrome extension will stall the first packet and query its local database to determine whether the URL is blacklisted. If it is not blacklisted, the process will terminate and the packet will resume its normal operation. However, if the URL is blacklisted, Chrome will collect vital information about the client's browser; shown in Table 1 and sent them to PoP. We denote this set of information as A.

[2] https://panopticlick.eff.org/browser-uniqueness.pdf

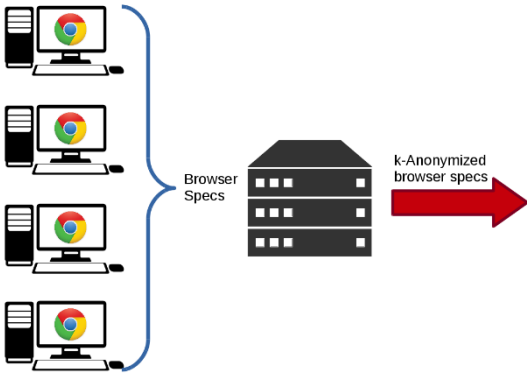

Figure 1 The general overview of our architecture.

Obfuscated elements	
Fonts	aa
Plugins	aa
Screen resolution	bb
User agent	cc

Table 1 Collected browser attributes by Brownymity.

On receiving A PoP will query its database to find similar records. PoP collects anonymous information from other clients and returns to the chrome extension the common elements that were found closer to the client's browser characteristics. This means that PoP will respond with a new set of information which is a subset of A which is denoted as A'.

By the time the central authority has responded to the chrome extension's initial request the extension will make the appropriate changes according to PoP's response. The spoofing of the plugins and other various elements is currently achieved by overriding the ‘’getter". Regarding font management, the Chrome extension sends sets F and F' the font lists of A and A' respectively to the daemon using a synchronous call to handle font spoofing.

The daemon will remove F, the core operating system (OS) fonts needed for the OS to function, that we denote as C Note that each OS has a number of fonts which come with the installation of the OS. Therefore, the system will not allow the removal of those fonts. From the remaining of the F set the F' will be removed as well as the set B which is the set of the fonts needed for the target site to render successfully. To summarize, the resulting set F'' is $F'' = F - F' - C - B$. The set F'' is the set of fonts needed to be removed from the system to make it

less random, as the set F'' is the set of fonts which render a user unique while all the other fonts the client has are much more common to the target website.

The removal of the fonts will take place in the registry of the Windows operating system and the overhead added of such an operation is proportional to the size of F'', typically close to 2-3 seconds. Once this action is performed the daemon will send a message to the chrome extension that the connection to the target website can be continued.

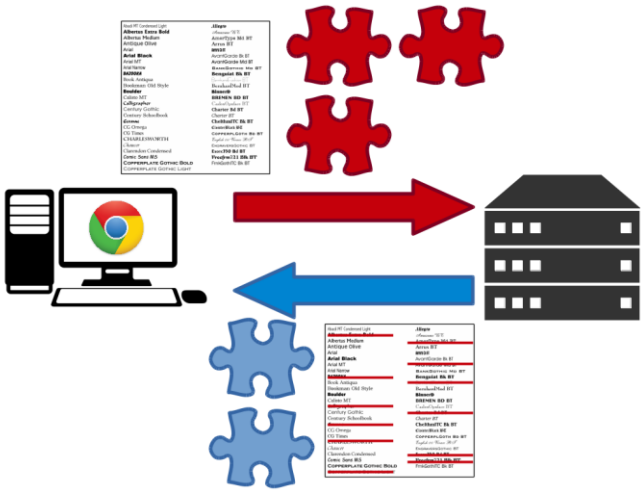

Figure 2 The general overview of our architecture

The implementation

The current version of Brownymity consists of three elements: the Brownymity extension, the Brownymity daemon and the PoP server.

The Brownymity extension is built for the Chrome browser and the extension as well as the daemon are built for Windows OSes, the reason for this last constraint is going to be discussed in the next section. The daemon as well as PoP were implemented in Python, using the Flask framework.

Discussion

Instead of spoofing the current fonts we decided to remove them completely from the system. This was done due to the appearance of many sophisticated ways to recover the font list even if the browser was trying to spoof the malicious site [5] [2]. As of that we decided that instead of spoofing those elements we could just modify them and thus make use of a more drastic measure. By using another browser like Tor [7] or a special crafted one in order to prevent font enumeration as stated in [4] might work but the use of fixed values as in Tor will alert the target site that the user is using a Tor browser and thus render the client more unique as this happens to most of the spoofing technologies [2]. Using a more dynamically and unpredictable method will make the new appearance of the client's browser reliable and will successfully spoof the target site.

Our approach is aligned with the recent findings of Fifield and Egelman [12] who propose that the whitelisting of a set of standard font files could provide further user anonymity.

Conclusions

In this work we've discussed the current literature and the problems we're faced with. Web fingerprinting is a huge area of research and most of the tools we use aren't even mature and cannot compare with the capabilities of the tools the attackers have in their possession. The attack surface is vast and research on this subject is in its infancy.

This huge difference might be that the first part of the equation, the "attackers" are motivated advertisement, governmental, military and other agencies trying to protect their products by identifying copyright violations or by providing a personalized ad model or by trying to identify potential terrorists. The funding on this side is huge and that's because their results are profitable for those organizations. On the other hand preventing fingerprinting carries little to no interest to most for-profit organizations and thus the funding for such research is limited to non-existent.

A new model for tools providing web anonymity was presented and developed. A user visiting a website would have to choose from a pool of profiles to present to the target website hiding this way its true identity. By providing a neutral intermediate to establish the communications with the website the user can hide certain elements which the browser doesn't need and shouldn't know. This way browser based fingerprinting is eliminated.

Our methodology proposes that every user visiting a certain website will share to it only his common elements with the other users and this way each user will only be present a subset of his browser's elements.

Apart from the theory a more practical conversation has been made where all the sensitive elements of our browser were presented and discussed ways to hide them. This part consisted of simply overriding their getter while a new novel method was presented as well in order to counter font enumeration. Uninstalling dynamically fonts from the system proved to be more efficient than trying to find a counter measure for every technique discovered for font enumeration.

All of those methodologies and techniques were implemented in a new experimental tool named Brownymity whose source code is available in the Appendix of this research.

Appendix

In this section the source code of Brownymity will be listed.

Chrome Extension

Manifest.js

```
{
  "name": "Brownymity",
  "description": "A browser anonymity tool - preventing others from finding out important information about your browser",
  "version": "0.1",
  "background": {
   "scripts": ["background.js"]
  },
  "browser_action": {
      "name": "Demo"

  },
  "permissions":[
    "tabs",
    "webRequest",
    "webRequestBlocking",
    "<all_urls>",
    "activeTab",
    "fontSettings",
    "unlimitedStorage"
  ],
  "manifest_version": 2
}
```

Background.js

```
var min = 1;
var max = 5;
var current = min;
var currentTabId;
var currentUrl;
var fonts = [];
var intersected_fonts;
var plugins_temp = navigator.plugins; //Get plugins
var windowInfo; //Get window info
var userAgent = navigator.userAgent; //Get useragent
var date = new Date();
var timezone = date.getTimezoneOffset(); //Get timezone offset
```

```
13. var screenscreenheight = screen.height;
14. var screenscreenwidth = screen.width;
15. var screenscreendepth = screen.colorDepth;
16. var cookieEnabled = navigator.cookieEnabled;
17. var productSub = navigator.productSub;
18. var vendor = navigator.vendor;
19. var appVersion = navigator.appVersion;
20. var plugins = [];
21. var tablink;
22. var all_tabs;
23. var temp_tab;
24. var start;
25. var flag = 0;
26. var should_block = false;
27.
28. for (var i = 0; i < plugins_temp.length ; i++) {
29.   plugins.push({
30.     name: plugins_temp[i].name,
31.     filename: plugins_temp[i].filename,
32.     description: plugins_temp[i].description
33.   });
34. };
35.
36. function errorHandler(e) {
37.   console.log(e);
38. }
39.
40. function maincall(details) {
41.   if(details && details.tabId >= 0 &&
42.       details.method === 'GET' &&
43.       details.type === 'main_frame'){// tabId=-
    1 is a call from an extension
44.   chrome.tabs.getSelected(null,currentTab);
45.   chrome.windows.getLastFocused(null, WindowInfo);
46.   sendRequest();
47.   }
48. }
49.
50. function parseURL(url) { //Function to parse the url. Aquiring the hostname of the vi
    siting url.
51.     var parser = document.createElement('a'),
52.         searchObject = {},
53.         queries, split, i;
54.     // Let the browser do the work
55.     parser.href = url;
56.     // Convert query string to object
57.     queries = parser.search.replace(/^\?/, '').split('&');
58.     for( i = 0; i < queries.length; i++ ) {
59.         split = queries[i].split('=');
60.         searchObject[split[0]] = split[1];
```

```
61.     }
62.     return {
63.         protocol: parser.protocol,
64.         host: parser.host,
65.         hostname: parser.hostname,
66.         port: parser.port,
67.         pathname: parser.pathname,
68.         search: parser.search,
69.         searchObject: searchObject,
70.         hash: parser.hash
71.     };
72. }
73. function sendtoflask(){
74.   //Sending to the flask framework in order to update the fonts
75.   var xhr2 = new XMLHttpRequest();
76.   var url2 = "http://localhost:5000/fonts";
77.   xhr2.open("POST", url2, false); // False for synchronous
78.   xhr2.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
79.   xhr2.onreadystatechange = function() {
80.     if (xhr2.readyState === 4) {
81.       //Upon receiving the response from the flask framework
82.       chrome.fontSettings.getFontList(giefFonts);
83.       var res = xhr2.responseText;
84.       var success_count = JSON.parse(res).success_count;
85.       var fail_count = JSON.parse(res).fail_count;
86.       var python_elapsed_time = JSON.parse(res).elapsed_time;
87.       console.log('Fonts successfully removed: ' + success_count);
88.       console.log('Fonts failed to be removed: ' + fail_count)
89.       var end = performance.now();
90.       var time = end - start;
91.       console.log('Execution time: ' + time/1000 + ' seconds');
92.       console.log('Font removal time: ' + python_elapsed_time + ' seconds')
93.     }
94.   };
95.   //Send only the plugins and the fonts. At the moment plugins are useless / no need
    to send them.
96.   xhr2.send("plugins=" + encodeURIComponent(JSON.stringify(plugins))
97.   //TODO Change plugins to received plugins and not the initial value
98.           + "&fonts=" + encodeURIComponent(JSON.stringify(fonts))
99.           + "&common_fonts=" + encodeURIComponent(JSON.stringify(intersected_fonts))

100.                );
101.       }
102.
103.       function call_remote_server(open_ports){
104.             var xhr = new XMLHttpRequest();
105.             // Preparing request to the peer network
106.             var url = "http://localhost:3001";
107.             xhr.open("POST", url, true);
```

```
108.             xhr.setRequestHeader("Content-type", "application/x-www-form-
    urlencoded");
109.             xhr.onreadystatechange = function() {
110.               if (xhr.readyState === 4) {
111.                 var serverResponse = xhr.responseText;
112.                 //Receiving the data from the peer network and override the getters

113.                 // with the appropriate data
114.                 //Change the user agent by modifying the navigator object
115.                 navigator.__defineGetter__('userAgent', function(){
116.                   return JSON.parse(serverResponse).userAgent;
117.                 });
118.
119.                 //Change the plugins by modifying the navigator object
120.                 navigator.__defineGetter__('plugins', function(){
121.                   return JSON.parse(serverResponse).plugins;
122.                 });
123.
124.                 //Change the cookieEnabled option by modifying the navigator object

125.                 navigator.__defineGetter__('cookieEnabled', function(){
126.                   return JSON.parse(serverResponse).cookieEnabled;
127.                 });
128.
129.                 navigator.__defineGetter__('productSub', function(){
130.                   return JSON.parse(serverResponse).productSub;
131.                 });
132.
133.                 navigator.__defineGetter__('vendor', function(){
134.                   return JSON.parse(serverResponse).vendor;
135.                 });
136.
137.                 navigator.__defineGetter__('appVersion', function(){
138.                   return JSON.parse(serverResponse).appVersion;
139.                 });
140.
141.                 screen.__defineGetter__('width', function(){
142.                   return JSON.parse(serverResponse).screenwidth;
143.                 })
144.
145.                 screen.__defineGetter__('depth', function(){
146.                   return JSON.parse(serverResponse).screendepth;
147.                 })
148.                 screen.__defineGetter__('height', function(){
149.                   return JSON.parse(serverResponse).screenheight;
150.                 })
151.                 // Resetting the window name in case of window.name monitoring
152.                 window.name = ""
153.                 console.log("Initiating font call to local server");
154.                 intersected_fonts = JSON.parse(serverResponse).fonts;
```

```
155.                 sendtoflask();
156.               }
157.             };
158.             xhr.send("plugins=" + encodeURIComponent(JSON.stringify(plugins))
159.                + "&userAgent=" + encodeURIComponent(JSON.stringify(userAgent))
160.                + "&timezone=" + encodeURIComponent(JSON.stringify(timezone))
161.                + "&cookieEnabled=" + encodeURIComponent(JSON.stringify(cookieEnabled
    ))
162.                + "&vendor=" + encodeURIComponent(JSON.stringify(vendor))
163.                + "&productSub=" + encodeURIComponent(JSON.stringify(productSub))
164.                + "&appVersion=" + encodeURIComponent(JSON.stringify(appVersion))
165.               // + "&windowInfo=" + encodeURIComponent(JSON.stringify(windowInfo))

166.                + "&screendepth=" + encodeURIComponent(JSON.stringify(screendepth))

167.                + "&screenwidth=" + encodeURIComponent(JSON.stringify(screenwidth))

168.                + "&screenheight=" + encodeURIComponent(JSON.stringify(screenheight))

169.                + "&fonts=" + encodeURIComponent(JSON.stringify(fonts))
170.                + "&ports=" + encodeURIComponent(JSON.stringify(open_ports))
171.               );
172.       }
173.
174.       function sendRequest(){
175.         chrome.tabs.getSelected(null,function(tab) {
176.           tablink = parseURL(tab.url);
177.         if(tablink.hostname === 'google.com' || tablink.hostname === 'www.google.gr'
    || tablink.hostname === 'www.google.com'){
178.           flag = 0;
179.           for (var i = 0; i < aβ□□ll_tabs.length; i++) {
180.             temp_tab = parseURL(all_tabs[i].url);
181.             if( temp_tab.hostname === 'google.com' || temp_tab.hostname === 'www.goo
    gle.gr' || temp_tab.hostname === 'www.google.com')
182.               flag = 1;
183.           };
184.
185.
186.           if (!flag){ //if flag = 0 meaning that the url google.com is not opened al
    ready
187.             start = performance.now();
188.             // Removing the port scanning technique for now
189.             // open_ports will be a null variable
190.             open_ports = 0;
191.             call_remote_server(open_ports);
192.           }
193.           should_block = false;
194.
195.         }
196.         else {
```

```
            should_block = true;
            console.log(tablink.hostname);
          }

          });
            chrome.tabs.query({},function(tabs){
              all_tabs = tabs;
            });
            chrome.fontSettings.getFontList(giefFonts);
            return should_block;
        }

        function currentTab(tab){
          currentTabId = tab.id;
          currentUrl = tab.favIconUrl;
        }

        function giefFonts(FontName){
          fonts = FontName;
        }

        chrome.tabs.query({},function(tabs){
          all_tabs = tabs;
        });

        chrome.webRequest.onBeforeRequest.addListener(
            maincall,
            {urls: ["<all_urls>"]},
            ["blocking"]);
        chrome.fontSettings.getFontList(giefFonts); // Get fonts
        function WindowInfo(e){
          windowInfo = e;
        }
```

LocalServer

Daemon2.py

```
from flask import Flask
from flask import request
from flask import jsonify
app = Flask(__name__)
import ast
import time
import csv
import os
from font_helper import f_helper
import shutil
```

```
import timeit
first_key = 'name'
second_key = 'registry_name'
cheat = [{first_key: 'Bodoni MT Condensed', second_key: ['Bodoni MT Condensed (TrueType)',
                                                         'Bodoni MT Condensed Bold (TrueType)',
                                                         'Bodoni MT Condensed Bold Italic (TrueType)',
                                                         'Bodoni MT Condensed Italic (TrueType)']},
         {first_key: 'Stencil', second_key: ['Stencil (TrueType)']},
         {first_key: 'Arial Unicode MS', second_key: ['Arial Unicode MS (TrueType)']},
         {first_key: 'Perpetua Titling MT', second_key: ['Perpetua Titling MT Bold (TrueType)',
                                                         'Perpetua Titling MT Light (TrueType)']},
         {first_key: 'Haettenschweiler', second_key: ['Haettenschweiler (TrueType)']},
         {first_key: 'Matura MT Script Capitals', second_key: ['Matura MT Script Capitals (TrueType)']},
         {first_key: 'Lucida Sans Typewriter', second_key: ['Lucida Sans Typewriter Bold (TrueType)',
                                                            'Lucida Sans Typewriter Bold Oblique (TrueType)',
                                                            'Lucida Sans Typewriter Oblique (TrueType)',
                                                            'Lucida Sans Typewriter Regular (TrueType)']},
         {first_key: 'Brush Script Std', second_key: ['BrushScriptStd (OpenType)']},
         {first_key: 'Trajan Pro', second_key: ['Trajan Pro',
                                                'TrajanPro-Bold (TrueType)',
                                                'TrajanPro-Regular (OpenType)',
                                                'TrajanPro-Regular (TrueType)']},
         {first_key: 'Arial Narrow', second_key: ['Arial Narrow (TrueType)',
                                                  'Arial Narrow Bold (TrueType)',
                                                  'Arial Narrow Bold Italic (TrueType)',
                                                  'Arial Narrow Italic (TrueType)']},
         {first_key: 'Nueva Std', second_key: ['NuevaStd-Bold (OpenType)',
                                               'NuevaStd-BoldCond (OpenType)',
                                               'NuevaStd-BoldCondItalic (OpenType)',
                                               'NuevaStd-Cond (OpenType)',
                                               'NuevaStd-CondItalic (OpenType)',
                                               'NuevaStd-Italic (OpenType)']},
         {first_key: 'Adobe Arabic', second_key: ['AdobeArabic-Bold (OpenType)',
```

```
                                                    'AdobeArabic-Bold (TrueType)',
                                                    'AdobeArabic-
BoldItalic (OpenType)',
                                                    'AdobeArabic-
BoldItalic (TrueType)',
                                                    'AdobeArabic-Italic (OpenType)',
                                                    'AdobeArabic-Italic (TrueType)',
                                                    'AdobeArabic-Regular (OpenType)',
                                                    'AdobeArabic-
Regular (TrueType)']},
        {first_key: 'Rosewood Std Regular', second_key: ['RosewoodStd-
Regular (OpenType)']},
        {first_key: 'Elephant', second_key: ['Elephant (TrueType)',
                                             'Elephant Italic (TrueType)']},
        {first_key: 'HelvLight', second_key: ['HelvLight Regular (TrueType)']},
        {first_key: 'Open Sans', second_key: ['Open Sans (TrueType)',
                                              'Open Sans Bold (TrueType)',
                                              'Open Sans Bold Italic (TrueType)',
                                              'Open Sans Extrabold (TrueType)',
                                              'Open Sans Extrabold Italic (TrueType)
',
                                              'Open Sans Italic (TrueType)',
                                              'Open Sans Light (TrueType)',
                                              'Open Sans Light Italic (TrueType)',
                                              'Open Sans Semibold (TrueType)',
                                              'Open Sans Semibold Italic (TrueType)'
]},
        {first_key: 'Hobo Std', second_key: ['HoboStd (OpenType)']},
        {first_key: 'Perpetua', second_key: ['Perpetua (TrueType)',
                                             'Perpetua Bold (TrueType)',
                                             'Perpetua Bold Italic (TrueType)',
                                             'Perpetua Italic (TrueType)',
                                             'Perpetua Titling MT Bold (TrueType)',

                                             'Perpetua Titling MT Light (TrueType)']
},
        {first_key: 'Myriad Hebrew', second_key: ['Myriad Hebrew (OpenType)',
                                                  'Myriad Hebrew Bold (OpenType)',
                                                  'Myriad Hebrew Bold Italic (OpenTy
pe)',
                                                  'Myriad Hebrew Italic (OpenType)']
},
        {first_key: 'Adobe Caslon Pro', second_key: ['ACaslonPro-
Italic (OpenType)',
                                                     'ACaslonPro-
Regular (OpenType)',
                                                     'ACaslonPro-
Semibold (OpenType)',
                                                     'ACaslonPro-
SemiboldItalic (OpenType)']},
```

```
80.          {first_key: 'Mesquite Std', second_key: ['MesquiteStd (OpenType)']},
81.          {first_key: 'Adobe Kaiti Std R', second_key: ['AdobeKaitiStd-
    Regular (OpenType)']},
82.          {first_key: 'Kozuka Gothic Pr6N R', second_key: ['KozGoPr6N-
    Regular (OpenType)']},
83.          {first_key: 'Tekton Pro', second_key: ['TektonPro-Bold (OpenType)',
84.                                                 'TektonPro-BoldCond (OpenType)',
85.                                                 'TektonPro-BoldExt (OpenType)',
86.                                                 'TektonPro-BoldObl (OpenType)']},
87.          {first_key: 'Bell MT', second_key: ['Bell MT (TrueType)',
88.                                              'Bell MT Bold (TrueType)',
89.                                              'Bell MT Italic (TrueType)']},
90.          {first_key: 'Lucida Sans', second_key: ['Lucida Sans Demibold Italic (TrueTy
    pe)',
91.                                                 'Lucida Sans Demibold Roman (TrueTyp
    e)',
92.                                                 'Lucida Sans Italic (TrueType)',
93.                                                 'Lucida Sans Regular (TrueType)',
94.                                                 'Lucida Sans Typewriter Bold (TrueTy
    pe)',
95.                                                 'Lucida Sans Typewriter Bold Oblique
     (TrueType)',
96.                                                 'Lucida Sans Typewriter Oblique (Tru
    eType)',
97.                                                 'Lucida Sans Typewriter Regular (Tru
    eType)',
98.                                                 'Lucida Sans Unicode (TrueType)']},

99.          # Gothic
100.                 {first_key: 'Kozuka Gothic Pr6N B', second_key: ['KozGoPr6N-
    Bold (OpenType)']},
101.                 {first_key: 'Kozuka Gothic Pr6N M', second_key: ['KozGoPr6N-
    Medium (OpenType)']},
102.                 {first_key: 'Kozuka Gothic Pr6N L', second_key: ['KozGoPr6N-
    Light (OpenType)']},
103.                 {first_key: 'Kozuka Gothic Pr6N H', second_key: ['KozGoPr6N-
    Heavy (OpenType)']},
104.                 {first_key: 'Kozuka Gothic Pr6N EL', second_key: ['KozGoPr6N-
    ExtraLight (OpenType)']},
105.                 # Mincho
106.                 {first_key: 'Kozuka Mincho Pr6N M', second_key: ['KozMinPr6N-
    Medium (OpenType)']},
107.                 {first_key: 'Kozuka Mincho Pr6N L', second_key: ['KozMinPr6N-
    Light (OpenType)']},
108.                 {first_key: 'Kozuka Mincho Pr6N B', second_key: ['KozMinPr6N-
    Bold (OpenType)']},
109.                 {first_key: 'Kozuka Mincho Pr6N R', second_key: ['KozMinPr6N-
    Regular (OpenType)']},
110.                 {first_key: 'Kozuka Mincho Pr6N EL', second_key: ['KozMinPr6N-
    ExtraLight (OpenType)']},
```

```
111.                {first_key: 'Kozuka Mincho Pr6N H', second_key: ['KozMinPr6N-
    Heavy (OpenType)']},
112.                # mincho PRO
113.                {first_key: 'Kozuka Mincho Pro B', second_key: ['KozMinPro-
    Bold (TrueType)']},
114.                {first_key: 'Kozuka Gothic Pro L', second_key: ['KozGoPro-
    Light (OpenType)']},
115.                {first_key: 'Kozuka Gothic Pro R', second_key: ['KozGoPro-
    Regular (OpenType)']},
116.                {first_key: 'Kozuka Gothic Pro B', second_key: ['KozGoPro-
    Bold (OpenType)']},
117.                {first_key: 'Kozuka Gothic Pro M', second_key: ['KozGoPro-
    Medium (OpenType)']},
118.                {first_key: 'Kozuka Gothic Pro H', second_key: ['KozGoPro-
    Heavy (OpenType)']},
119.                {first_key: 'Franklin Gothic Heavy', second_key: ['Franklin Gothic He
    avy (TrueType)',
120.                                                                  'Franklin Gothic He
    avy Italic (TrueType)']},
121.                {first_key: 'Franklin Gothic Demi', second_key: ['Franklin Gothic Dem
    i (TrueType)',
122.                                                                  'Franklin Gothic De
    mi Cond (TrueType)',
123.                                                                  'Franklin Gothic De
    mi Italic (TrueType)']},
124.                {first_key: 'Franklin Gothic Book', second_key: ['Franklin Gothic Boo
    k (TrueType)',
125.                                                                  'Franklin Gothic Bo
    ok Italic (TrueType)']},
126.                {first_key: 'Vivaldi', second_key: ['Vivaldi Italic (TrueType)']},
127.                {first_key: 'Myriad Pro Light', second_key: ['MyriadPro-
    Light (OpenType)']},
128.                {first_key: 'Adobe Gothic Std B', second_key: ['AdobeGothicStd-
    Bold (OpenType)']},
129.                {first_key: 'Agency FB', second_key: ['Agency FB (TrueType)',
130.                                                      'Agency FB Bold (TrueType)']},

131.                {first_key: 'Adobe Naskh Medium', second_key: ['Adobe Naskh Medium (O
    penType)']},
132.                {first_key: 'Orator Std', second_key: ['OratorStd (OpenType)',
133.                                                       'OratorStd-
    Slanted (OpenType)']},
134.                {first_key: 'Adobe Song Std L', second_key: ['AdobeSongStd-
    Light (OpenType)']},
135.                {first_key: 'Lucida Calligraphy', second_key: ['Lucida Calligraphy It
    alic (TrueType)']},
136.                {first_key: 'Tekton Pro Cond', second_key: ['TektonPro-
    BoldCond (OpenType)']},
137.                {first_key: 'Adobe Devanagari', second_key: ['AdobeDevanagari-
    Bold (OpenType)',
```

```
138.                                                    'AdobeDevanagari-
    BoldItalic (OpenType)',
139.                                                    'AdobeDevanagari-
    Italic (OpenType),'
140.                                                    'AdobeDevanagari-
    Regular (OpenType)']},
141.                {first_key: 'Gill Sans MT', second_key: ['Gill Sans MT (TrueType)',
142.                                                        'Gill Sans MT Bold (TrueType
    )',
143.                                                        'Gill Sans MT Bold Italic (T
    rueType)',
144.                                                        'Gill Sans MT Condensed (Tru
    eType)',
145.                                                        'Gill Sans MT Ext Condensed
    Bold (TrueType)',
146.                                                        'Gill Sans MT Italic (TrueTy
    pe)',
147.                                                        'Gill Sans Ultra Bold (TrueT
    ype)',
148.                                                        'Gill Sans Ultra Bold Conden
    sed (TrueType)']},
149.                {first_key: 'Charlemagne Std', second_key: ['CharlemagneStd-
    Bold (OpenType)']},
150.                {first_key: 'Tekton Pro Ext', second_key: ['TektonPro-
    BoldExt (OpenType)']},
151.                {first_key: 'Letter Gothic Std', second_key: ['LetterGothicStd (OpenT
    ype)',
152.                                                          'LetterGothicStd-
    Bold (OpenType)',
153.                                                          'LetterGothicStd-
    BoldSlanted (OpenType)',
154.                                                          'LetterGothicStd-
    Slanted (OpenType)']},
155.                {first_key: 'Nueva Std Cond', second_key: ['NuevaStd-
    BoldCond (OpenType)',
156.                                                        'NuevaStd-
    BoldCondItalic (OpenType)',
157.                                                        'NuevaStd-
    Cond (OpenType)',
158.                                                        'NuevaStd-
    CondItalic (OpenType)']},
159.                {first_key: 'Adobe Myungjo Std M', second_key: ['AdobeMyungjoStd-
    Medium (OpenType)']},
160.                {first_key: 'Stencil Std', second_key: ['StencilStd (OpenType)']},
161.                {first_key: 'Berlin Sans FB Demi', second_key: ['Berlin Sans FB Demi
    Bold (TrueType)']},
162.                {first_key: 'Minion Pro', second_key: ['MinionPro-Bold (OpenType)',
163.                                                      'MinionPro-
    BoldIt (OpenType)',
164.                                                      'MinionPro-It (OpenType)',
```

```
165.                                                    'MinionPro-
    Regular (OpenType)']},
166.                 {first_key: 'Minion Pro Med', second_key: ['MinionPro-
    Medium (OpenType)',
167.                                                               'MinionPro-
    MediumIt (OpenType)']},
168.                 {first_key: 'Minion Pro SmBd', second_key: ['MinionPro-
    Semibold (OpenType)',
169.                                                               'MinionPro-
    SemiboldIt (OpenType)']},
170.                 {first_key: 'Minion Pro Cond', second_key: ['MinionPro-
    BoldCn (OpenType)',
171.                                                               'MinionPro-
    BoldCnIt (OpenType)',]},
172.                 {first_key: 'Chaparral Pro Light', second_key: ['ChaparralPro-
    LightIt (OpenType)']},
173.                 {first_key: 'Cooper Std Black', second_key: ['CooperBlackStd (OpenTyp
    e)']},
174.                 {first_key: 'Lithos Pro Regular', second_key: ['LithosPro-
    Regular (OpenType)']},
175.                 {first_key: 'Myriad Arabic', second_key: ['Myriad Arabic Bold (OpenTy
    pe)',
176.                                                              'Myriad Arabic Bold Italic
    (OpenType)',
177.                                                              'Myriad Arabic Italic (Open
    Type)',
178.                                                              'Myriad Arabic (OpenType)']
    },
179.                 {first_key: 'High Tower Text', second_key: ['High Tower Text (TrueTyp
    e)',
180.                                                               'High Tower Text Italic (
    TrueType)']},
181.                 {first_key: 'Salina', second_key: ['Salina (TrueType)',
182.                                                    'Salina Regular (TrueType)']}, {fi
    rst_key: ''},
183.                 {first_key: 'Bodoni MT Black', second_key: ['Bodoni MT Black (TrueTyp
    e)',
184.                                                               'Bodoni MT Black Italic (
    TrueType)']},
185.                 {first_key: 'Lucida Handwriting', second_key: ['Lucida Handwriting It
    alic (TrueType)']},
186.                 {first_key: 'Myriad Pro Cond', second_key: ['MyriadPro-
    BoldCond (OpenType)',
187.                                                               'MyriadPro-
    BoldCondIt (OpenType)',
188.                                                               'MyriadPro-
    Cond (OpenType)',
189.                                                               'MyriadPro-
    CondIt (OpenType)']},
190.                 {first_key: 'Myriad Pro', second_key: ['MyriadPro-Bold (OpenType)',
```

```
191.                                                    'MyriadPro-
    BoldIt (OpenType)',
192.                                                    'MyriadPro-It (OpenType)',
193.                                                    'MyriadPro-
    Regular (OpenType)',
194.                                                    'MyriadPro-
    Semibold (OpenType)',
195.                                                    'MyriadPro-
    SemiboldIt (OpenType)']},
196.                {first_key: 'Adobe Heiti Std R', second_key: ['AdobeHeitiStd-
    Regular (OpenType)']},
197.                {first_key: 'Garamond', second_key: ['Garamond (TrueType)',
198.                                                     'Garamond Bold (TrueType)',
199.                                                     'Garamond Italic (TrueType)']},

200.                {first_key: 'Brush Script MT', second_key: ['Brush Script MT Italic (
    TrueType)']},
201.                {first_key: 'Blackoak Std', second_key: ['BlackoakStd (TrueType)',
202.                                                         'BlackoakStd (OpenType)']},

203.                {first_key: 'Adobe Ming Std L', second_key: ['AdobeMingStd-
    Light (OpenType)']},
204.                {first_key: 'Adobe Caslon Pro Bold', second_key: ['ACaslonPro-
    Bold (OpenType)',
205.                                                              'ACaslonPro-
    BoldItalic (OpenType)']},
206.                {first_key: 'Tw Cen MT Condensed', second_key: ['Tw Cen MT Condensed
    (TrueType)',
207.                                                            'Tw Cen MT Condensed
    Bold (TrueType)',
208.                                                            'Tw Cen MT Condensed
    Extra Bold (TrueType)']},
209.                {first_key: 'Open Sans Light', second_key: ['Open Sans Light (TrueTyp
    e)',
210.                                                          'Open Sans Light Italic (
    TrueType)']},
211.                {first_key: 'Adobe Fangsong Std R', second_key: ['AdobeFangsongStd-
    Regular (OpenType)']},
212.                {first_key: 'Giddyup Std', second_key: ['GiddyupStd (OpenType)',
213.                                                       'GiddyupStd (TrueType)']},
214.                {first_key: 'Magneto', second_key: ['Magneto Bold (TrueType)']},
215.                {first_key: 'Tw Cen MT', second_key: ['Tw Cen MT (TrueType)',
216.                                                     'Tw Cen MT Bold (TrueType)',
217.                                                     'Tw Cen MT Bold Italic (TrueTyp
    e)',
218.                                                     'Tw Cen MT Italic (TrueType)']}
    ,
219.                {first_key: 'Lucida Bright', second_key: ['Lucida Bright (TrueType)',
```

```
220.                                                    'Lucida Bright Demibold (Tr
    ueType)',
221.                                                    'Lucida Bright Demibold Ita
    lic (TrueType)',
222.                                                    'Lucida Bright Italic (True
    Type)']},
223.                {first_key: 'Open Sans Extrabold', second_key: ['Open Sans Extrabold
    (TrueType)',
224.                                                                 'Open Sans Extrabold
    Italic (TrueType)']},
225.                {first_key: 'Poplar Std', second_key: ['PoplarStd (OpenType)']},
226.                {first_key: 'Adobe Garamond Pro Bold', second_key: ['AGaramondPro-
    Bold (OpenType)',
227.                                                                    'AGaramondPro-
    BoldItalic (OpenType)']},
228.                {first_key: 'Rockwell Condensed', second_key: ['Rockwell Condensed (T
    rueType)',
229.                                                               'Rockwell Condensed Bo
    ld (TrueType)']},
230.                {first_key: 'Adobe Hebrew', second_key: ['AdobeHebrew-
    Bold (OpenType)',
231.                                                        'AdobeHebrew-
    BoldItalic (OpenType)',
232.                                                        'AdobeHebrew-
    Italic (OpenType)',
233.                                                        'AdobeHebrew-
    Regular (OpenType)']},
234.                {first_key: 'Chaparral Pro', second_key: ['ChaparralPro-
    Bold (OpenType)',
235.                                                         'ChaparralPro-
    BoldIt (OpenType)',
236.                                                         'ChaparralPro-
    Italic (OpenType)',
237.                                                         'ChaparralPro-
    LightIt (OpenType)',
238.                                                         'ChaparralPro-
    Regular (OpenType)']},
239.                {first_key: 'Prestige Elite Std', second_key: ['PrestigeEliteStd-
    Bd (OpenType)']},
240.                {first_key: 'FlemishScript BT', second_key: ['Flemish Script BT (True
    Type)']},
241.                {first_key: 'Tiranti Solid LET', second_key: ['Tiranti Solid LET (Tru
    eType)',
242.                                                              'Tiranti Solid LET Plai
    n:1.0 (TrueType)']},
243.                {first_key: 'Rockwell', second_key: ['Rockwell (TrueType)',
244.                                                    'Rockwell Bold (TrueType)',
245.                                                    'Rockwell Bold Italic (TrueType)
    ',
```

```
246.                                                    'Rockwell Condensed (TrueType)',
247.                                                    'Rockwell Condensed Bold (TrueTy
    pe)',
248.                                                    'Rockwell Extra Bold (TrueType)'
    ,
249.                                                    'Rockwell Italic (TrueType)']},
250.                 {first_key: 'Bodoni MT', second_key: ['Bodoni MT (TrueType)',
251.                                                       'Bodoni MT Black (TrueType)',
252.                                                       'Bodoni MT Black Italic (TrueTy
    pe)',
253.                                                       'Bodoni MT Bold (TrueType)',
254.                                                       'Bodoni MT Bold Italic (TrueTyp
    e)',
255.                                                       'Bodoni MT Condensed (TrueType)
    ',
256.                                                       'Bodoni MT Condensed Bold (True
    Type)',
257.                                                       'Bodoni MT Condensed Bold Itali
    c (TrueType)',
258.                                                       'Bodoni MT Condensed Italic (Tr
    ueType)',
259.                                                       'Bodoni MT Italic (TrueType)',
260.                                                       'Bodoni MT Poster Compressed (T
    rueType)']},
261.                 {first_key: 'Bookman Old Style', second_key: ['Bookman Old Style (Tru
    eType)',
262.                                                               'Bookman Old Style Bold
     (TrueType)',
263.                                                               'Bookman Old Style Bold
     Italic (TrueType)',
264.                                                               'Bookman Old Style Ital
    ic (TrueType)']},
265.                 {first_key: 'Californian FB', second_key: ['Californian FB (TrueType)
    ',
266.                                                            'Californian FB Bold (True
    Type)',
267.                                                            'Californian FB Italic (Tr
    ueType)']},
268.                 {first_key: 'Goudy Old Style', second_key: ['Goudy Old Style (TrueTyp
    e)',
269.                                                             'Goudy Old Style Bold (Tr
    ueType)',
270.                                                             'Goudy Old Style Italic (
    TrueType)']},
271.                 {first_key: 'OCR A Std', second_key: ['OCRAStd (OpenType)']},
272.                 {first_key: 'Adobe Garamond Pro', second_key: ['AGaramondPro-
    Italic (OpenType)',
```

```
273.                                                                'AGaramondPro-
    Regular (OpenType)']},
274.                {first_key: 'Open Sans Semibold', second_key: ['Open Sans Semibold (T
    rueType)',
275.                                                                'Open Sans Semibold It
    alic (TrueType)']},
276.                {first_key: 'Adobe Fan Heiti Std B', second_key: ['AdobeFanHeitiStd-
    Bold (OpenType)']},
277.                {first_key: 'Berlin Sans FB', second_key: ['Berlin Sans FB (TrueType)
    ',
278.                                                           'Berlin Sans FB Bold (True
    Type)',
279.                                                           'Berlin Sans FB Demi Bold
    (TrueType)']},
280.                {first_key: 'Lucida Fax', second_key: ['Lucida Fax Demibold (TrueType
    )',
281.                                                       'Lucida Fax Demibold Italic (T
    rueType)',
282.                                                       'Lucida Fax Italic (TrueType)'
    ,
283.                                                       'Lucida Fax Regular (TrueType)
    ']},
284.                {first_key: 'Century Schoolbook', second_key: ['Century Schoolbook (T
    rueType)',
285.                                                                'Century Schoolbook Bo
    ld (TrueType)',
286.                                                                'Century Schoolbook Bo
    ld Italic (TrueType)',
287.                                                                'Century Schoolbook It
    alic (TrueType)']}]
288.       @app.route('/ports', methods=['POST'])
289.       def get_ports():
290.           # Find open ports from this system
291.           import psutil
292.
293.           connections = psutil.net_connections()
294.           temp_connections = []
295.           open_ports = []
296.           print len(connections)
297.           for index, connection in enumerate(connections):
298.               if connection[5] != 'NONE' \
299.                       and connection[5] != 'CLOSE_WAIT' \
300.                       and connection[5] != 'TIME_WAIT' \
301.                       and connection[3][0] != '127.0.0.1' \
302.                       and connection[3][0] != '::' \
303.                       and connection[3][0] != '::1' \
304.                       and connection[3][0] != '0.0.0.0':
305.                   temp_connections.append(connection)
306.                   open_ports.append(connection[3][1])
307.
```

```
308.             # Convert to set to avoid duplicate values
309.             # Then back to a list and sort it
310.             sorted_ports = sorted(list(set(open_ports)))
311.             return jsonify(ports=sorted_ports)
312.
313.
314.         @app.route('/fonts', methods=['POST'])
315.         def modify_fonts():
316.             # Start the timer
317.             start_time = timeit.default_timer()
318.             # Receive data from POST request
319.             fonts = request.form.getlist('fonts')
320.             common_fonts = request.form.getlist('common_fonts')
321.             # Transform fonts to a literal form in order to be accessible
322.             fonts = ast.literal_eval(fonts[0])
323.             # Do the same for the common fonts
324.             common_fonts = ast.literal_eval(common_fonts[0])
325.             temp = []
326.             # Transform the fonts into a list containing only the name of the font
327.             for x in range(len(fonts)):
328.                 temp.append(fonts[x]['fontId'])
329.             fonts = temp
330.             # Read the core OS fonts from the CSV
331.             core_fonts = f_helper.read_csv()
332.             # Remove the common and core fonts from the initial set of fonts.
333.             # The remaining fonts are the surplus and should be removed
334.             fonts_to_remove = list(set(fonts) - set(common_fonts) - set(core_fonts))
335.
336.             # At this moment we have to extract the list of fonts from the user's regi
    stry
337.             # Using the find_fonts function a list of tuples containing the font filen
    ame, displayname
338.             # and a value identifying the value data
339.             # Temp font list will hold the triplet font name, file name and the other
    one
340.             registry_fonts = f_helper.find_fonts()
341.
342.
343.             font_list = []
344.             for i in range(len(registry_fonts)):
345.                 font_list.append(registry_fonts[i][0])
346.             temp = []
347.
348.             for i in range(len(fonts_to_remove)):
349.                 temp1 = [y for y in font_list if fonts_to_remove[i] + " (TrueType)" in
     y]
350.                 if temp1:
351.                     temp.append(temp1)
352.             small_list = [item for sublist in temp for item in sublist]
353.             small_list_without_truetype = []
```

```
354.
355.           # Filter the font_list list and keep the tuple with the fonts to delete
356.           # At this moment we can only delete the TrueType fonts
357.           fonts_to_go_with_paths = []
358.           for i in range(len(registry_fonts)):
359.               for y in range(len(small_list)):
360.                   if str(small_list[y]) == str(registry_fonts[i][0]):
361.                       fonts_to_go_with_paths.append(registry_fonts[i])
362.
363.           list_to_fuzzy_search = list(set(fonts_to_remove) - set(small_list_without_
    truetype))
364.
365.           # Associate the fuzzy list ( the non Truetype or the list of fonts with mu
    ltiple files)
366.           # With their equivalent on the cheat table
367.           fonts_to_go_with_paths_fuzzy_part = []
368.           for i in range(len(list_to_fuzzy_search)):
369.               for j in range(len(cheat)):
370.                   if list_to_fuzzy_search[i] == cheat[j][first_key]:
371.                       fonts_to_go_with_paths_fuzzy_part.append(cheat[j])
372.
373.           # Now we need to associate the cheat values with the appropriate value nam
    es from the registry table
374.           for i in range(len(fonts_to_go_with_paths_fuzzy_part)):
375.               for y in range(len(fonts_to_go_with_paths_fuzzy_part[i][second_key])):

376.                   for j in range(len(registry_fonts)):
377.                       if fonts_to_go_with_paths_fuzzy_part[i][second_key][y] == regi
    stry_fonts[j][0]:
378.                           fonts_to_go_with_paths.append(registry_fonts[j])
379.
380.           # Remove duplicate values in the list
381.           fonts_to_go_with_paths = list(set(fonts_to_go_with_paths))
382.
383.           # Remove the unnecessary fonts
384.           success_count = 0
385.           fail_count = 0
386.           # Uncomment the following snippet to make a backup of your fonts before de
    leting them
387.           # make notice that this will slow down the process!
388.           # for i in range(len(fonts_to_go_with_paths)):
389.           #     shutil.copy2('C:\Windows\Fonts\\' + fonts_to_go_with_paths[i][1], 'C
    :\Users\Beast\Desktop\la\\')
390.           for i in range(len(fonts_to_go_with_paths)):
391.               try:
392.                   result = f_helper.delete_font(fonts_to_go_with_paths[i][1])
393.
394.                   if result:
395.
396.                       success_count += 1
```

```
397.                     else:
398.                         # print 'The font ' + fonts_to_go_with_paths[i][1] + ' failed
    to be removed'
399.                         fail_count += 1
400.                 except WindowsError as e:
401.                     print 'Font ' + fonts_to_go_with_paths[i][1] + ' failed to copy'
402.                     print e
403.
404.             # print 'Total fonts removed ' + str(success_count)
405.             # print 'Fonts not removed  ' + str(fail_count)
406.             # Calculate elapsed time and print it to the console
407.             elapsed = timeit.default_timer() - start_time
408.             # print "Elapsed time: " + str(elapsed) + " seconds"
409.             return jsonify(success_count=success_count,
410.                            fail_count=fail_count,
411.                            elapsed_time=elapsed)
412.
413.
414.         if __name__ == '__main__':
415.             app.run()
```

F_helper.py

```
1.  import os
2.  import csv
3.  import json
4.  import win32api
5.  import win32con
6.  from _winreg import *
7.  from fuzzywuzzy import process
8.  import ctypes
9.  import platform
10.
11.
12. def csv2list(file2read):
13.     with open(file2read, 'rb') as f:
14.         reader = csv.DictReader(f)
15.         your_list = list(reader)
16.     return your_list
17.
18.
19. def read_csv():
20.     try:
21.         if  platform.system() == 'Windows':
22.             vanilla_windows = csv2list('windows-vanilla_fonts.csv')
23.         else:
24.             #temporary solution
```

```
25.             vanilla_fonts = csv2list('windows-vanilla_fonts.csv')
26.     except IOError as e:
27.         print "I/O error({0}): {1}".format(e.errno, e.strerror)
28.     except ValueError:
29.         print "Could not convert data to an integer."
30.
31.
32.
33.     vanilla_fonts_windows = []
34.     for x in range(len(vanilla_windows)):
35.         vanilla_fonts_windows.append(vanilla_windows[x]['Typeface'])
36.     return vanilla_fonts_windows
37.
38.
39. def find_fonts():
40.     font_list = []
41.     if platform.system() == 'Windows':
42.         aReg = ConnectRegistry(None, HKEY_LOCAL_MACHINE)
43.         aKey = OpenKey(aReg, r"SOFTWARE\Microsoft\Windows NT\CurrentVersion\Fonts")
44.
45.         for i in range(10240):
46.             try:
47.                 asubkey_name=EnumValue(aKey, i)
48.                 font_list.append(asubkey_name)
49.                 if "" == asubkey_name:
50.                     raise WindowsError
51.             except WindowsError:
52.                 break
53.         CloseKey(aKey)
54.     return font_list
55.
56.
57. def delete_font(font):
58.
59.     if platform.system() == 'Windows':
60.         path = 'C:\Windows\Fonts\\' + font
61.         result = ctypes.windll.gdi32.RemoveFontResourceA(path)
62.         if not result:
63.             result = ctypes.windll.gdi32.RemoveFontResourceW(path)
64.         win32api.SendMessage(win32con.HWND_BROADCAST, win32con.WM_FONTCHANGE)
65.     return result
```

Network Server

Daemon.py

```
1.  from flask import Flask
2.  from flask import request
3.  from flask import jsonify
4.  from intersect_lists import intersect_lists
5.  import flask
6.  import json
7.  import ast
8.  import sys
9.  app = Flask(__name__)
10.
11.
12. @app.route('/', methods=['POST'])
13. def hello_world():
14.     userAgent = request.form.get('userAgent')
15.     cookieEnabled = request.form.get('cookieEnabled')
16.     screendepth = request.form.get('screendepth')
17.     screenwidth = request.form.get('screenwidth')
18.     screenheight = request.form.get('screenheight')
19.     plugins = request.form.getlist('plugins')
20.     fonts = request.form.getlist('fonts')
21.     vendor = request.form.get('vendor')
22.     productSub = request.form.get('productSub')
23.     appVersion = request.form.get('appVersion')
24.     open_ports = request.form.get('ports')
25.
26.     fonts = ast.literal_eval(fonts[0])
27.     plugins = ast.literal_eval(plugins[0])
28.
29.     userAgent = ast.literal_eval(userAgent)
30.
31.     # Set new user agent according to http://techblog.willshouse.com/2012/01/03/most-
    common-user-agents/
32.     # Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chro
    me/39.0.2171.95 Safari/537.36
33.     # is the most common user agent evah for windows 64 bit
34.
35.
36.
37.     # Get the common fonts
38.     common_fonts, count = intersect_lists.simple_intersect(fonts)
39.     print ('all is good')
40.     # print common_fonts
41.     return jsonify(userAgent=userAgent, screenwidth=screenwidth,
42.                    screenheight=screenheight, screendepth=screendepth,
43.                    cookieEnabled=cookieEnabled,
```

```
                    fonts=common_fonts['fonts'],
                    plugins=plugins,
                    vendor=vendor,
                    productSub=productSub,
                    appVersion=appVersion)


if __name__ == '__main__':
    app.run(port=3001)
```

Intersect_lists.py

```
import json
import ast
import csv

def get_data(filename):
    fin_mac = open(filename)
    data = json.loads(fin_mac.read())
    plugins_data = json.loads(data["plugins"])
    plugins = []
    for pl in plugins_data:
        plugins.append(pl["filename"])
    fnts = json.loads(data["fonts"])
    fonts = []
    for f in fnts:
        fonts.append(f["fontId"])
    res = {}
    res.update({"plugins": plugins})
    res.update({"fonts": fonts})
    res.update({"cookieEnabled": data["cookieEnabled"]})
    res.update({"screenheight": data["screenheight"]})
    res.update({"screenwidth": data["screenwidth"]})
    res.update({"screendepth": data["screendepth"]})
    res.update({"timezone": data["timezone"]})
    res.update({"userAgent": data["userAgent"]})
    return res


def simple_intersect(res1):
    try:
        res2 = get_data("demo-fonts.json")
    except ValueError as e:
        print e
    r = {}
    cnt = 0
    res2 = remove_unicode(res2["fonts"])

```

```
37.     new_res1 = []
38.     for x in range(len(res1)):
39.         new_res1.append(res1[x]['fontId'])
40.     r.update({"fonts": list(set(new_res1).intersection(res2))})
41.
42.     cnt += len(r["fonts"])
43.     return r, cnt
44.
```

References

[1] J. R. Mayer and J. C. Mitchell, "Third-party web tracking: Policy and technology," in *Security and Privacy (SP), 2012 IEEE Symposium on*, IEEE, 2012, pp. 413-427.

[2] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens and G. Vigna, "Cookieless monster: Exploring the ecosystem of web-based device fingerprinting," in *Security and privacy (SP), 2013 IEEE symposium on*, 2013.

[3] P. Eckersley, "How unique is your web browser?," in *Privacy Enhancing Technologies*, Springer, 2010, pp. 1-18.

[4] N. Nikiforakis, W. Joosen and B. Livshits, "Privaricator: Deceiving fingerprinters with little white lies," in *International World Wide Web Conference (WWW), May 2015*, 2015.

[5] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan and C. Diaz, "The Web never forgets: Persistent tracking mechanisms in the wild," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014.

[6] K. Boda, "Firegloves," [Online]. Available: http://fingerprint.pet-portal.eu/menu=6.

[7] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gurses, F. Piessens and B. Preneel, "FPDetective: dusting the web for fingerprinters," in *ACM SIGSAC conference on Computer & communications security*, Berlin, 2013.

[8] A. Soltani, S. Canty, Q. Mayo, L. Thomas and C. J. Hoofnagle, "Flash Cookies and Privacy," in *2010 AAAI Spring Symposium Series*, 2010.

[9] M. Ayenson, D. J. Wambach, A. Soltani, N. Good and C. J. Hoofnagle, "Flash cookies and privacy II: Now with HTML5 and ETag respawning," in *World Wide Web Internet and Web Information Systems*, 2011.

[10] S. Englehardt, D. Reisman, C. Eubank, P. Zimmerman, J. Mayer, A. Narayanan and E. W. Felten, "Cookes that give you away: The surveillance implications of web tracking," [Online]. Available: http://randomwalker.info/publications/cookie-surveillance-v2.pdf.

[11] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in HTML5," in *Workshop on Web 2.0 Security and Privacy*, 2012.

[12] D. Fifield and S. Egelman, "Fingerprinting web users through font metrics," in *Financial Cryptography and Data Security 2015*, 2015.

[13] K. Mowery, D. Bogenreif, S. Yilek and H. Shacham, "Fingerprinting information in JavaScript implementations," *Proceedings of W2SP,* 2011.

[14] S. Dey, N. Roy, W. Xu, R. R. Choudhury and S. Nelakuditi, "Accelprint: Imperfections of accelerometers make smartphones trackable," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2014.

[15] Z. Zhou, W. Diao, X. Liu and K. Zhang, "Acoustic fingerprinting revisited: Generate stable device id stealthily with inaudible sound," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014.

[16] H. Bojinov, Y. Michalevsky, G. Nakibly and D. Boneh, "Mobile Device Identification via Sensor Fingerprinting," *arXiv preprint arXiv:1408,1416,* 2014.

[17] N. Mor, O. Riva, S. Nath and J. Kubiatowicz, "Bloom Cookies: Web Search Personalization without User Tracking," 2015.

[18] S. Guha, B. Cheng and P. Francis, "Privad: Practical Privacy in Online Advertising," in *Proceedings of NSDI'11: 8th USENIX Symposiu on Networked Systems Design and Implementation*, 2011.

[19] F. Besson, N. Bielova and T. Jensen, "Browser Randomisation against Fingerprinting: A Quantitative Information Flow Approach," in *Secure IT Systems*, Springer, 2014, pp. 181-196.

[20] U. Fiore, A. Castiglione, A. D. Santis and F. Palmieri, "Countering Browser Fingerprinting Techiniques: Constructing a Fake Profile with Google Chrome," in *Network-Based Information Systems (NBiS), 2014 7th International Conference on*, 2014.

[21] X. Pan, Y. Cao and Y. Chen, "I Do Not Know What You Visited Lat Summer: Protecting Users from Third-Party Web Tracking with TrackingFree Browser," in *22nd Annual Network and Distributed System Security Symposium, NDSS*, 2015.

[22] U. Fiore, A. Castiglione, A. De Santis and F. Palmieri, "Countering Browser Fingerprinting Techniques: Constructing a Fake Profile with Google Chrome," in *Network-Based Information Systems (NBiS), 2014 17th International Conference on*, 2014, pp. 355-360.

[23] M. Rausch, A. Bakke, S. Patt, B. Wegner and D. Scott, "Demonstrating a Simple Device Fingerprinting System," in *Midewest Instruction and Computing Symposium*, 2014.

[24] E. Rader, "Awareness of behavioral tracking and information privacy concern in facebook and google," in *Proc. of Symposium on Usable Privacy and Security (SOUPS)*, Meno Part, CA, USA, 2014.

[25] F. Besson, N. Bielova and T. Jensen, "Enforcing Browser Anonymity with Quantitative Information Flow," 2014. [Online]. Available: https://hal.inria.fr/hal-00984654.

[26] A. Guellier, C. Bidan and N. Prigent, "Homomorphic Cryptography-Based Privacy-Preserving Network Communications," in *Application and Techinique in Information Security*, vol. 490, Springer, 2014, pp. 159-170.

[27] M. Rausch, N. Good and C. J. Hoofnagle, "Searching for Indicators of Device Fingerprinting in the JavaScript Code of Popular Websites".

[28] E. Chan-Tin and et al, "Identifying Webbrowsers in Encrypted Communications," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, 2014.

[29] T. Hupperich and M. Kuhrer, "Mobile Device Fingerprinting," in *9. GI FG SIDAR Graduierten-Workshop uber Reaktive Sicherheit*, p. 10.

[30] G. Acar, "Obfuscation for and against device fingerprinting Position Paper for Symposium on Obfuscation New York University," 2014.

[31] B. Hayes, "Uniquely Me!," *AMERICAN SCIENTIST,* vol. 102, pp. 106-109, 2014.

[32] W. De Groef, D. Devriese , M. Vanhoef and F. Piessens, "Information Flow Control for Web Scripts," in *Foundations of Security Analysis and Design VII*, Springer, 2014, pp. 124-145.

[33] S. Han, V. Liu, Q. Pu, S. Peter, T. Anderson, A. Krishnamurthy and D. Wetherall, "Expressive privacy control with pseudonyms," in *ACM SIGCOMM Computer Communications Review*, vol. 43, ACM, 2013, pp. 291-302.

[34] D. I. Wolinksy, D. Jackowitz and B. Ford, "Managing NymBoxes for identity and tracking protection," in *USENIX Conference on Timely Results in Operating Systems*, 2014.

[35] M. Stopczynski and M. Zugelder, "Reducing User Tracking through Automatic Web Site State Isolations," in *Information Security*, Springer, 2014, pp. 309-327.

[36] M. Juarez and V. Torra, "DisPA: An Intelligent Agent for Private Web Search," in *Advanced Research in Data Privacy*, Springer, 2015, pp. 389-405.

[37] D. Kim, "Poster: Detection and Prevention of Web-based Device Fingerprinting".

[38] F. Shirazi and M. Volkmaer, "What Deters Jane from Preventing Identification and Tracking on the Web?," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, 2014.