Countering Browser Fingerprinting Techniques: Constructing a Fake Profile with Google Chrome

Ugo Fiore[§]
Information Services Centre
University of Naples "Federico II"
Via Cinthia, 26 I-80125
Napoli (NA), Italy
ugo.fiore@unina.it[§]

Aniello Castiglione[*][†], Alfredo De Santis[†]
Department of Computer Science
University of Salerno
Via Giovanni Paolo II, 132
I-84084 - Fisciano (SA), Italy
castiglione@ieee.org*, ads@unisa.it[†]

Francesco Palmieri[‡]
Department of Industrial and
Information Engineering
Second University of Naples
Via Roma, 29 I-81031 - Aversa (CE), Italy
francesco.palmieri@unina.it[‡]

***Abstract*—While Web browsers are fundamental components in the Internet nowadays, the widespread availability of several techniques that can be used to detect the individual browser connected to a server raises privacy issues that need to be adequately addressed. Browser fingerprinting uses a combination of attributes, whose values are silently obtained during normal navigation, to identify, with high likelihood, a browser and, consequently, who is controlling such browser.**

Although providing contrived or random values for specific single attributes or attribute sets can be at least partly successful in interfering with the operation of fingerprinting, such approach may also have adverse effects. The objective of this work is to suggest an alternative: supplying coherent data designed in such a way as to mimick as closely as possible a different browser. A proof-of-concept implementation as an extension for the Chrome browser is presented and discussed.

***Keywords*-Web Security; Browser Fingerprinting; Privacy; Anonymity; Digital Profiling; Browser Identification; Stealth Browsing.**

I. Introduction

Device fingerprinting can be defined as the act of discovering and correlating information that can be used to identify, with high likelihood, a device. Over the last years, digital profiling [1], [2], has known extensive use and alleged abuse [3]. In particular, browsing activities and habits are an actively sought target by various parties, because that information has an enormous potential to provide valuable insights when designing marketing campaigns or conveying advertisements to specific class of users. Increasingly, content offered on the Web contains "third-party" components originated by advertising, online social network, or analytics business. The purpose of such components is to derive and collect information about the browser, the device on which the browser is operating, the browsing activity and lot of additional information. Indeed, browser fingerprinting products are commercially sold and advertised by several enterprises.

While the results of fingerprinting may be useful in protecting users and enterprises against fraud, the amount of available information and its accuracy places a significant threat on the users' privacy. The set of properties adopted for fingerprinting as well as the collection of tools used is conspicuous. It includes, for example, browser cookies [4], Flash cookies [5], IP addresses, browsing history, online social network accounts, navigation patterns [6], etc..

Today, user experience is fundamental in several application. In order to enhance the user experience, usually, browsers report to the connected server some internal information, typically concerning its available "capabilities". In fact common in browser fingerprinting is the gathering of all such information from the browser and the use of such information for profiling purposes. It is worth noting that, unlike cookie-based tracking, fingerprinting is completely passive: no persistent tagging is left on the device, making detection of fingerprinting techniques a puzzling task even when in presence of expert users. Furthermore, unlike tracking cookies, fingerprinting cannot be simply disabled by switching to private browsing modes, making it a precious resource for who are interested in reliable tracking of browsing habits. In fact, switching to private browsing modes may have devastating consequences on the user experience.

In this work, instead of planning and taking steps with the intent to avoid being profiled, the possibility of building a fake profile is addressed. Our goal is to alter the data that can be used for fingerprinting purposes in such a way that the modified data will be consistent with those available to the connected party if our browser would have been a different one. Transmitting such modified data can hinder fingerprinting or allow the user to expose a fake, but valid, profile when in presence of tracking activities by some web servers.

The remainder of this paper is organized as follows. After a brief review of relevant related work (in Section II), fingerprinting is discussed in Section III. The case study concerning the Google Chrome browser is the subject of Section IV, preceding Section V which concludes the paper and lays directions for future work.

[†] Corresponding author: Aniello Castiglione, Department of Computer Science, University of Salerno, Via Giovanni Paolo II 132, I-84084, Fisciano (SA), ITALY. E-mail: castiglione@ieee.org castiglione@acm.org, Phone: +39089969594, Fax: +39089969821

978-1-4799-4224-4/14 $31.00 © 2014 IEEE
DOI 10.1109/NBiS.2014.102

II. Related Work

Fingerprinting and profiling, as well as techniques aimed at countering both of them, have attracted considerable attention in recent literature. Tapiador *et al.* [7], in the context of profiling users starting from activity traces, introduced a notion of indistinguishability and practical schemes to obfuscate user traces. The work in [8] discussed a list of information useful for fingerprinting, and the results of a live experiment showing how a significant percentage of devices could be fingerprinted with an accuracy so high that it would allow identification.

Nikiforakis *et al.* [9] discovered that some popular web sites use fingerprinting techniques to some extent. They also showed that, in some cases, when Flash or Java plugins were unavailable, JavaScript-based font probing was used. Third-party trackers were detected and classified according to how browser information was manipulated, finding that many commercial pages are tracked by multiple parties [10]. Another work discussed stealth fingerprinting scripts that, in order to evade detection, would remove themselves from the page's DOM after collecting data [11]. Details in the implementation of the JavaScript engine were exploited in [12] to differentiate between browsers. PriVaricator [13] is targeted at attaining unlinkability between multiple subsequent contacts with the same server, rather than at reducing the identifiability of the browser. The objective is achieved through the provision of randomized values. FourthParty, a Firefox extension presented in a survey paper on Web tracking [3], has been used to gather data about information leakage, tracking technologies, and effectiveness of blocking tools.

The key point of the proposed work is very simple: instead of forging information (this task is already accomplished by the majority of individuals who try to back out of the fingerprinting activity) it proposes the creation of a fingerprint that is exactly the same to the one that would have been left by someone else. The principle is quite similar to what happens when building a series of actions to frame someone. Such actions have not been done by the individual that seems to have produced them but by a third party whose aim is to frame an innocent without his awareness. This, in turn, adopt techniques that are present and well-known in the field of digital alibi [14], [15], [16], [17].

III. Browser Fingerprinting

The basic idea behind browser fingerprinting is simple. How rare and unique is the configuration that a browser exposes to servers? To what extent can such information be used to identify a group of machines or even a single computer? [8] In order to acquire an awareness of the scope, methods and issues related to browser and device fingerprinting, besides browsing relevant research literature, the interested reader may refer to some resources available online.

A. Resources

Several resources are available on the Internet in order to support browser fingerprinting, ranging from dedicated sites to specific standalone ad-hoc tools.

a) Profiling Sites: A list of the most significant sites providing explicit profiling services across the Internet follows:

- What's my User Agent [1]
- What is my Browser [2]
- About my Browser [3]
- BrowserSpy [4]
- EFF Panopticlick [5]

Techniques used in these sites cover a wide variety of cases: they range from simple header analysis (thanks to "What's my User Agent") to an assessment of the degree of uniqueness of an aggregate of information made available by the browser ("Panopticlick"). BrowserSpy offers a variety of tests, including tests targeted to specific applications, plugins, or components (e.g., Silverlight or Adobe Acrobat Reader).

b) Tools: Complementing the sites listed above, several software tools are available. Broadly speaking, the purpose of these tools is to allow the user to protect their privacy, neutralize the fingerprinting methods or sometimes supply contrived information. This goal is accomplished by either altering the behavior of the browser or modifying the information that is sent over in HTTP headers. A selection of tools is:

- NoScript. Open-source add-on for Firefox allows preemptive script blocking and selective control over plugins. Disabling JavaScript altogether, however, has a significant impact on the user experience.
- User Agent Switcher. Firefox add-on than can manipulate the User-Agent HTTP header.
- HTTP Header Changer. An extension for the Chrome browser designed to change HTTP request headers.
- ShareMeNot[6]. A browser extension designed to keep embedded third-party buttons (in particular, those related to online social networks) from tracking browsing activity unless the user actually clicks on them. ShareMeNot does not completely remove the buttons.
- TorBrowser. The same values are reported to potential fingerprinters for all Tor users, that should be thus indistinguishable. Notably, a limit of it is also set on the number of fonts that can be used in a page.
- Firegloves. Firegloves is a proof-of-concept extension for Mozilla Firefox that returns randomized values when queried for fingerprinting-relevant attributes. Unfortunately, Firegloves is no longer supported by their creators.

B. Web Profiling Techniques

Several details about internal characteristics are exposed by a browser to the connected server, because they are useful to build a tailored environment and attain an improved user experience. Specific mechanisms and interfaces are, therefore, in place to enable querying of such attributes. However, the gathered results may provide, when combined, sufficient

[1] http://www.whatsmyuseragent.com/
[2] http://www.whatismybrowser.com/
[3] https://aboutmybrowser.com/
[4] http://browserspy.dk/
[5] https://panopticlick.eff.org/
[6] http://sharemenot.cs.washington.edu

information to profile and even identify a browser or device. Attributes include system information, time zone, language, and browser configuration. Representatives of such attributes that are being widely used in fingerprinting include:

- Screen resolution
- Navigator
- Timezone
- Browser plugins & MIME types
- System fonts

Clearly, characteristics that take the same value for many devices (e.g., OS version) are less distinctive than attributes than can take more diverse values. Given the variety of fonts that can be installed, the font list is likely to be the most accurate test, i.e., the one which provides the highest amount of information. Some data can be directly read or inferred from the HTTP headers. Other characteristics need a more sophisticated collection method.

As fas as the transfer of fingerprinting data to the connected server is concerned, several "vehicles" can be used for this purpose:

c) HTTP headers: HTTP header fields are control structures for the HTTP protocol. Headers used in a HTTP response naturally deliver information to the connected server. The most significant example comes from the User Agent string that is a string field in the HTTP Request Header properly set by the client to report the user agents' main features, in terms of name, operating system and browser version.

d) JavaScript: ubiquitously available in browsers, is the *de-facto* client-side Web scripting language. With JavaScript, elements of the Document Object Model (DOM) of a page can be accessed and manipulated. JavaScript is, moreover, supported by all major browsers, and is available on mobile devices. All this makes JavaScript a very powerful and popular tool to be used for fingerprinting purposes. Notably, the presence of a specific font on the system where the browser is running can be checked with JavaScript by surreptitiously measuring and then comparing the dimensions of text rendered with different fonts (unavailable fonts are rendered with fallback fonts).

e) Flash/Java: both plugins offer APIs for font enumeration. The scripting language of Flash, ActionScript, does include methods for discovering the list of installed fonts.

An important issue to be considered is whether fingerprinting is active or passive. From the standpoint of the information gatherer, fingerprinting methods that can operate without the target user's explicit consent or awareness are preferable to techniques requiring user interaction. In particular, the Flash Player transmits information without asking.

C. Building a Fake Profile

If user privacy is a concern, countermeasures against fingerprinting should be taken and their efficacy assessed. A quick fix that one may think of is to disable the operation of leakage vehicles altogether, effectively stopping the transmission of data to the tracking server. However, taking such draconian measures may not be the most effective solution in all cases. Two considerations are in order.

Firstly, context is important. Borrowing from studies on human communication, where it is well known that unwillingness to communicate is still communication (*"if it is accepted that all behavior in an interactional situation has message value, i.e., is communication, it follows that no matter how one may try, one cannot* not *communicate"*) [18], it should be noted that switching a plugin off still sends a signal that can be caught. In particular, consider the case of a browser plugin that is not responding but other information available to the connected server suggests that it should be. The connected party can therefore suspect that the unresponsive behavior is likely to result from deliberate and selective disabling of plugins, rather than from a minimalist browser configuration. Such deduction, for example, can be trivial when Flash is listed as an installed plugin but fails to respond.

On the other hand, the operation of plugins, or other information leakage vehicles, can be essential to the user experience. For example, many web sites simply deny access if the browser does not support JavaScript. Thus, stopping JavaScript may be unacceptable for a large portion of users. Data on the percentage of users who have JavaScript disabled suggest, anyway, that as many as around 1% of users have JavaScript disabled on their browsers [19].

In a broader perspective, when trying to reduce the effectiveness of fingerprinting efforts, special care should be taken to ensure that the very same mechanisms meant to protect privacy would not in fact *increase* the distinguishability of fingerprints instead of diminishing it, what Eckersley calls "The Paradox of Fingerprintable Privacy Enhancing Technologies" [8]. Simply redacting information, in fact, or altering it in a simplistic way (e.g., via User Agent Switcher) can result in a quite uncommon fingerprint and thus be counterproductive.

Instead of thwarting fingerprinting attempts by attempting at blocking information, the objective in this work is to offer fingerprinters a *fake profile*, i.e., a coherent and valid fingerprint that is nevertheless different from the genuine one. At the same time, the profiling information should be constructed in such a way as to mimic real profiles that can be frequently found. The server should be tricked into classifying the browser in the wrong category.

It is important to underline that the contrived profiling information can not be freely chosen: coherence constraints must be satisfied. For example, not every browser supports Flash or Java. In addition, given a browser, some features are not enabled by the majority of users. Even the presence of such features may thus be leaking important pieces of information. Also, fingerprinters methods are known to tailor their approach to the specific parameters of the targeted browser, once they recognize its type by means of a range of techniques that may also include analysis of browser-specific features (e.g., screen.mozBrightness) [9]. Therefore, when constructing a plausible fake profile, limitations and restrictions must be reckoned upon, ensuring that the presented information is consistent. Typical profiles, described as completely as possible, should thus be collected from real-world computers and browsers, forming a collection of profiles from which to choose the one to expose to the connected party.

IV. Case Study: Google Chrome

In the Chrome browser, the browser kernel is the only component that can interact with the local filesystem, while the rendering engine is run in a sandbox with lower privileges. The rendering engine is responsible for all tasks needed for rendering Web pages, including for example parsing of HTML text and interpreting Javascript. However, the rendering engine typically loads fonts files from the filesystem.

A. An Extension to Enhance Privacy

Protection against fingerprinting through the construction of a fake profile could, in principle, be done by modifying the browser source code, via a special-purpose HTTP proxy, or with a browser extension.

The choice of implementing the fake profile as an extension is motivated by the need to realize a module with a structure and user interface as simple as possible, while at the same time retaining the ability to interact intimately with the browser, having access to its internals. While modifying the source code would have been the most powerful tactic, it would have been overly complicated and would have involved a significant maintenance effort. In theory, a specialized HTTP proxy able to interpret, filter, and modify data useful for fingerprinting could be a valuable tool for isolating the key ideas about efforts against fingerprinting from the details about the browser.

On the other hand, details about how some features are implemented in a specific browser are exactly one of the tools used for fingerprinting [12]. Thus, with the proxy-based approach it would not be easy to reproduce the desired behavior for many browsers. In addition to the above considerations, it should be remarked that an extension allows the design to be easily split into two major sections: the first contains general concepts and techniques, whereas the second deals with features specific to the browser under scrutiny. This modular approach will enable the consideration of other browsers, by keeping the basic ideas and adapting only the relevant portions, when needed. For the Chrome browser, moreover, the opportunity of using Javascript as the extension language also paves the way to fast and agile prototyping and development.

The extension prototype, working within the Linux, Windows and MacOS environment, allows the modification of the browser profile by choosing between different predefined profiles or by preparing new ad-hoc ones. In doing this it reconfigures most of the HTTP header fields together with the navigator and screen objects, by simultaneously inhibiting the acquisition of system fonts information.

In the following subsections, each of the above items will be discussed with some details.

1) HTTP Headers: while, in principle, all HTTP headers can be modified, those that convey most information are listed below:

- Accept
- Accept-Charset
- Accept-Encoding
- Accept-Language
- Content-Type
- Referer
- User-Agent

The extension will modify them, inserting coherent data taken from the extension catalogue of configured fake profiles. In order to improve the versatility of the extension, the configuration also includes control over the sites to which the modified profiling data are to be provided. Alterations may thus be applied selectively, for a single site of a group of sites, matching an expression. In this way, users may pick up a different appearance for their browser for different categories of sites.

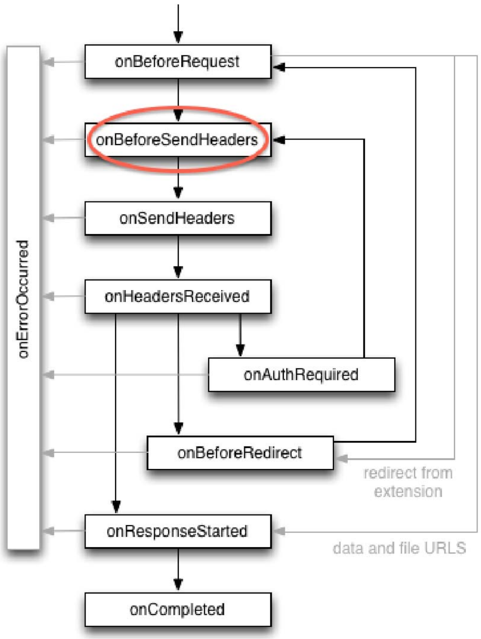
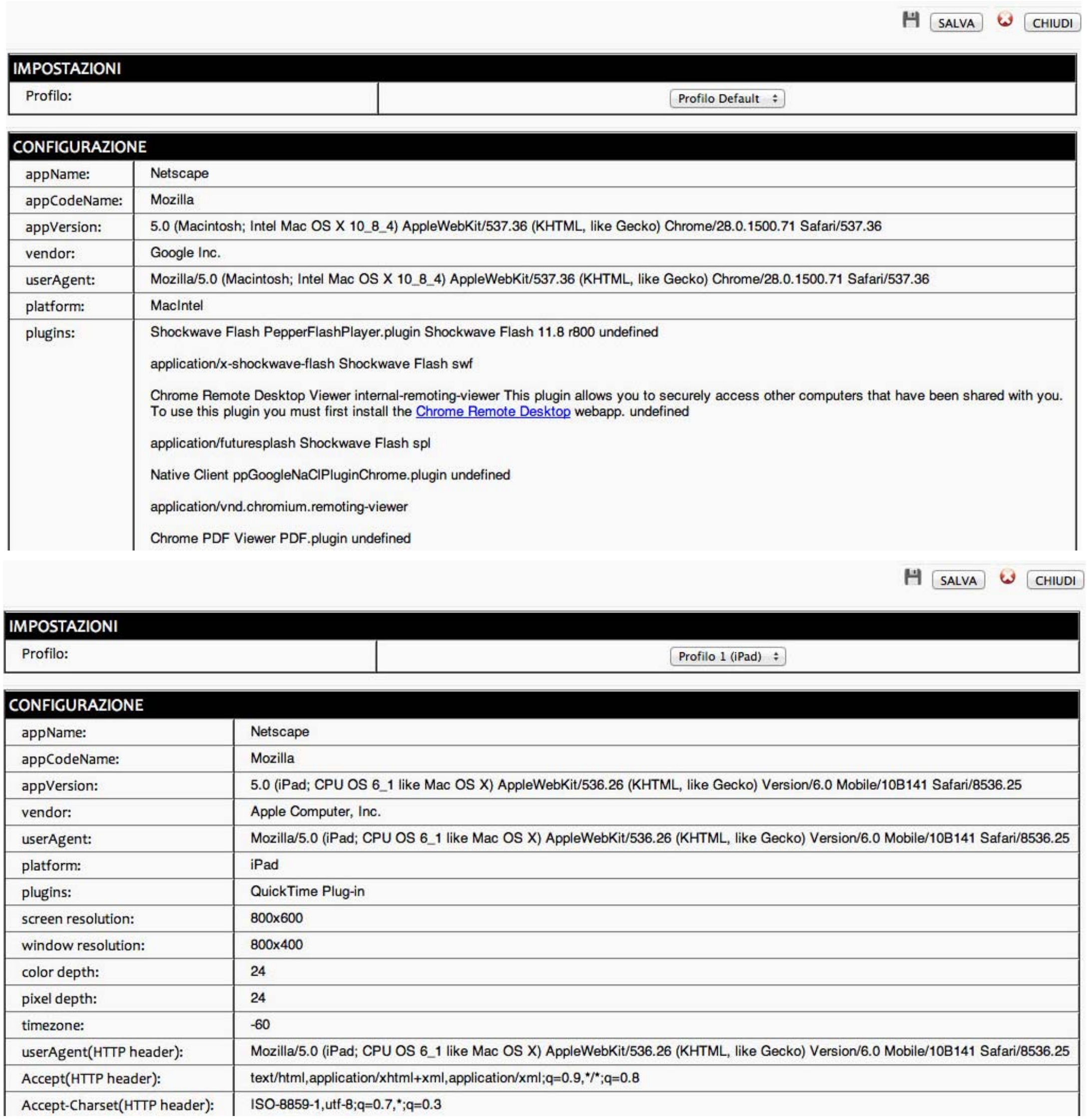


Fig. 1. The event life cycle of requests in Google Chrome [20].

It should be noted that HTTP request headers are dynamically built on the basis of data relative to the browser, the configuration, and the request. In order to be able to modify fingerprinting-relevant information such as the headers, the extension must act when the involved data are ready but before they are sent over to the connected server. To this end, a listener has been added to the onBeforeSendHeaders event, which occurs when an HTTP request is about to occur, that is, before any HTTP data is sent. A short example illustrating how to modify the "User-Agent" field by



IMPOSTAZIONI	
Profilo:	Profilo Default

CONFIGURAZIONE	
appName:	Netscape
appCodeName:	Mozilla
appVersion:	5.0 (Macintosh; Intel Mac OS X 10_8_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/28.0.1500.71 Safari/537.36
vendor:	Google Inc.
userAgent:	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/28.0.1500.71 Safari/537.36
platform:	MacIntel
plugins:	Shockwave Flash PepperFlashPlayer.plugin Shockwave Flash 11.8 r800 undefined application/x-shockwave-flash Shockwave Flash swf Chrome Remote Desktop Viewer internal-remoting-viewer This plugin allows you to securely access other computers that have been shared with you. To use this plugin you must first install the Chrome Remote Desktop webapp. undefined application/futuresplash Shockwave Flash spl Native Client ppGoogleNaClPluginChrome.plugin undefined application/vnd.chromium.remoting-viewer Chrome PDF Viewer PDF.plugin undefined

IMPOSTAZIONI	
Profilo:	Profilo 1 (iPad)

CONFIGURAZIONE	
appName:	Netscape
appCodeName:	Mozilla
appVersion:	5.0 (iPad; CPU OS 6_1 like Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10B141 Safari/8536.25
vendor:	Apple Computer, Inc.
userAgent:	Mozilla/5.0 (iPad; CPU OS 6_1 like Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10B141 Safari/8536.25
platform:	iPad
plugins:	QuickTime Plug-in
screen resolution:	800x600
window resolution:	800x400
color depth:	24
pixel depth:	24
timezone:	-60
userAgent(HTTP header):	Mozilla/5.0 (iPad; CPU OS 6_1 like Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10B141 Safari/8536.25
Accept(HTTP header):	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Charset(HTTP header):	ISO-8859-1,utf-8;q=0.7,*;q=0.3

Fig. 2. Building a fake browsing profile: original profile on top, and fake profile on bottom.

using the onBeforeSendHeaders event is given in Listing 1. An overview of the events that can be used is shown in Fig. 1.

```
chrome.webRequest.onBeforeSendHeaders.addListener(function(details)
{

var headers = details.requestHeaders, blockingResponse = {};

for (var i = 0; i < headers.length; ++i) {

        if (headers[i].name === 'User—Agent') {

              headers[i].value = ``Firefox/25.0 (FakeOS 25.0) Benji/26060803'';
        }
}
},

blockingResponse.requestHeaders = headers;

return blockingResponse;
);
```

Listing 1. Modifying the User-Agent field by means of the onBeforeSendHeaders event.

2) The Navigator Object: this client-side object contains information about the browser. It can be queried for the browser name and version, the supported MIME types and plugins, the platform on which the browser was compiled, the language supported, and the operating system under which the browser is running.

The extension will change the following properties:

- navigator.appCodeName
- navigator.appName
- navigator.appVersion
- navigator.language
- navigator.mimeTypes
- navigator.vendor
- navigator.userAgent

- navigator.platform
- navigator.plugins

Notably, the navigator.plugins will property enumerate all the plugins currently installed on the browser. Thus, special care must be taken with plugins that are unsupported on the platform and that we are attempting to emulate will not be reported as installed.

3) The screen object: this object contains information about the user's screen, including resolution (height and width) as well as color depth. While no public standard applies to the screen object, all major browsers support it.

B. *Proof-of-concept Verification*

The proposed extension has been verified both by direct human-driven examination and by using the Panopticlick web site, by comparing the results for the fake profile and the one associated to an actual device showing the same characteristics, for a variety of devices and browsers. In all the cases, the tests' results where substantially indistinguishable, as expected. An example showing the effects of creating a fake profile can be appreciated from what depicted in Fig. 2.

V. Conclusions

In this work, a strategy to deter improper use of browser fingerprinting is proposed. The key point is to supply the connected party with coherent fingerprinting information that is taken from genuine profiles, derived from real devices or browsers, or specifically crafted profiles designed in such a way as to make profiling, identification, and tracking as hard as possible.

Directions for future work are manifold. Development lines involve study on how to modify/block actions performed via the jQuery library, expanding the list of items to modify, automate the collection of complete, credible profiles, and experimenting on other browsers. In particular, alteration of the reported list of installed fonts should be completed in such a way as to correctly handle the fingerprinting analysis based on font rendering with JavaScript. Analogously, encapsulation of JavaScript is under study with the purpose of attempting at neutralizing the technique by Mulazzani *et al.* [12]. In a complementary way, the analysis of how fingerprints vary over time are another direction towards which further study will we focused.

Acknowledgment

The authors would like to thank Michele D'Amato and Valerio Piccolo for contributing in the development of the Chrome extension.

References

[1] C. Colombini and A. Colella, "Digital Profiling: A Computer Forensics Approach," in *Availability, Reliability and Security for Business, Enterprise and Health Information Systems*, ser. Lecture Notes in Computer Science, A. Tjoa, G. Quirchmayr, I. You, and L. Xu, Eds. Springer Berlin Heidelberg, 2011, vol. 6908, pp. 330–343. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-23300-5_26

[2] C. Colombini, A. Colella, M. Mattiucci, and A. Castiglione, "Network Profiling: Content Analysis of Users Behavior in Digital Communication Channel," in *Multidisciplinary Research and Practice for Information Systems*, ser. Lecture Notes in Computer Science, G. Quirchmayr, J. Basl, I. You, L. Xu, and E. Weippl, Eds. Springer Berlin Heidelberg, 2012, vol. 7465, pp. 416–429. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-32498-7_31

[3] J. Mayer and J. Mitchell, "Third-Party Web Tracking: Policy and Technology," in *Security and Privacy (SP), 2012 IEEE Symposium on*, May 2012, pp. 413–427.

[4] B. Krishnamurthy and C. E. Wills, "Generating a privacy footprint on the Internet," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. ACM, 2006, pp. 65–70.

[5] A. Soltani, S. Canty, Q. Mayo, L. Thomas, and C. J. Hoofnagle, "Flash Cookies and Privacy," in *AAAI Spring Symposium: Intelligent Information Privacy Management*, 2010.

[6] A. Castiglione, A. De Santis, U. Fiore, and F. Palmieri, "Device Tracking in Private Networks via NAPT Log Analysis," in *6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2012*. IEEE, 2012, pp. 603–608.

[7] J. E. Tapiador, J. C. Hernandez-Castro, and P. Peris-Lopez, "Online Randomization Strategies to Obfuscate User Behavioral Patterns," *Journal of Network and Systems Management*, vol. 20, no. 4, pp. 561–578, 2012.

[8] P. Eckersley, "How Unique Is Your Web Browser?" in *Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science, vol. 6205. Springer, 2010, pp. 1–18.

[9] 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*. IEEE, 2013, pp. 541–555.

[10] F. Roesner, T. Kohno, and D. Wetherall, "Detecting and defending against third-party tracking on the web," in *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. San Jose, CA: USENIX, 2012, pp. 155–168. [Online]. Available: https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/roesner

[11] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel, "FPDetective: dusting the web for fingerprinters," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 1129–1140.

[12] M. Mulazzani, P. Reschl, M. Huber, M. Leithner, S. Schrittwieser, and E. Weippl, "Fast and reliable browser identification with Javascript engine fingerprinting," in *Web 2.0 Workshop on Security and Privacy (W2SP)*, 2013.

[13] N. Nikiforakis, W. Joosen, and B. Livshits, "PriVaricator: Deceiving Fingerprinters with Little White Lies," Microsoft Corporation, Tech. Rep. MSR-TR-2014-26, February 2014.

[14] A. De Santis, A. Castiglione, G. Cattaneo, G. De Maio, and M. Ianulardo, "Automated construction of a false digital alibi," in *Availability, Reliability and Security for Business, Enterprise and Health Information Systems*, ser. Lecture Notes in Computer Science, A. Tjoa, G. Quirchmayr, I. You, and L. Xu, Eds. Springer Berlin Heidelberg, 2011, vol. 6908, pp. 359–373. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-23300-5_28

[15] P. Albano, A. Castiglione, G. Cattaneo, G. De Maio, and A. De Santis, "On the Construction of a False Digital Alibi on the Android OS," *Intelligent Networking and Collaborative Systems, International Conference on*, vol. 0, pp. 685–690, 2011.

[16] A. Castiglione, G. Cattaneo, G. De Maio, A. De Santis, G. Costabile, and M. Epifani, "The Forensic Analysis of a False Digital Alibi," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, July 2012, pp. 114–121.

[17] A. Castiglione, G. Cattaneo, R. De Prisco, A. De Santis, and K. Yim, "How to Forge a Digital Alibi on MacOS X," in *Multidisciplinary Research and Practice for Information Systems*, ser. Lecture Notes in Computer Science, G. Quirchmayr, J. Basl, I. You, L. Xu, and E. Weippl, Eds. Springer Berlin Heidelberg, 2012, vol. 7465, pp. 430–444. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-32498-7_32

[18] P. Watzlawick, J. Helmick-Beavin, and D. D. Jackson, *Pragmatics of Human Communication*. New York: Norton, 1967.

[19] Nicholas C. Zakas, "How many users have JavaScript disabled?" Oct 2010. [Online]. Available: https://developer.yahoo.com/blogs/ydn/many-users-javascript-disabled-14121.html

[20] Google Developers, "chrome.webRequest documentation," 2014. [Online]. Available: https://developer.chrome.com/extensions/webRequest