Inhibiting Browser Fingerprinting and Tracking

Sakchan Luangmaneerote
Electronic and Computer Science
University of Southampton
Southampton, SO17 1BJ, UK
sl8e14@soton.ac.uk

Ed Zaluska and Leslie Carr
Electronic and Computer Science
University of Southampton
Southampton, SO17 1BJ, UK
ejz@ecs.soton.ac.uk

***Abstract*— This paper discusses possible approaches to address the loss of user privacy when browsing the web and being tracked by websites which compute a browser fingerprint identifying the user computer. The key problem is that the current fingerprinting countermeasures are insufficient to prevent fingerprinting tracking and also frequently produce side-effects on the web browser. The advantages and disadvantages of possible countermeasures are discussed in the context of improving resistance against browser fingerprinting. Finally, using a new browser extension is proposed as the best way to inhibit fingerprinting as it could probably inhibit some of the fingerprinting techniques used and also diminish the side-effects on the user browser experience, compared with existing techniques.**

Keywords; Browser Fingerprinting, User Privacy, Web Tracking, Fingerprinting Countermeasure, side-effects

I. Introduction

A typical user computer usually has a large number of different attributes which facilitates identification of the user. The rapid advances in web technologies (e.g. CSS, JavaScript, HTML, etc.) help the user to browse the web conveniently, but at the expense of user privacy. These technologies provide communication between the web browser and the web server but are unable to restrict access to the user data. A consequence of this inability to control access to user data is that users are at a high risk of a loss of privacy. A third party could identify the user from the user data, a technique known as “fingerprinting”. On a typical user computer, there are many attributes used (e.g. the list of plugins, the list of fonts, user agents, operating system version, etc.) that can be easily accessed using the web browser. This information will be combined into one string, and a fingerprinting ID then created (typically by using a hashing function). These user attributes can be accessed through a variety of different channels (e.g. JavaScript, network traffic, plugins, etc.). The stability of the fingerprinting ID created (sometimes called a “hash ID”) will depend on the alteration of the attributes associated with the user computer (e.g., updating of the operating system, fonts or version of the web browser, etc.). Unlike stateful tracking (e.g. using web cookies or Flash cookies), this fingerprinting technique does not need to store any files on user computers. This creates a substantial problem for Internet users who wish to retain their privacy as they cannot detect any trace on their computers. Today, many companies (e.g. Bluecava [1] and Iovation [2]) have adapted these fingerprinting techniques to fingerprint millions of Internet users and then track them, even after all cookies are disabled. Knowledge of the user identity by a third party has both advantages and disadvantages. Considering the advantages, fingerprinting can be used with web analytics [3] to tackle online fraud such as identity theft or credit card fraud [4, 5]. For example, some new anti-fraud payment systems will not only check the user login with the normal authentication techniques but also authenticate the user using a fingerprinting technique. The information obtained will assist the anti-fraud payment system to check basic information (such as IP address, user agents, screen resolution, timezone, system language, processor characteristic, etc.) and if this is significantly different from the last login, then additional security requirements will typically be invoked. The disadvantages of browser fingerprinting are that it can be used as a web cookie regenerator [6] (in the case of disabled cookies), spread malware [7] and to track users without their consent [6] (which creates a potentially-significant loss of privacy).

A number of countermeasures have recently been proposed to tackle fingerprinting tracking. There are a number of different ideas and concepts - some countermeasures seem to be more effective to prevent fingerprinting tracking, while others appear to be somewhat far-fetched and likely to create more problems than they will solve. This paper will discuss which countermeasures appear to be more suitable for preventing fingerprinting tracking and then evaluate the best approaches to tackle this problem. Finally, it proposes a new countermeasure to address the problem of fingerprinting tracking.

The key contributions of this paper are the following:

1) *Classification of fingerprinting countermeasures*
2) *Discussing possible approaches to prevent fingerprinting tracking*
3) *Proposing a new countermeasure to address the weaknesses of the current countermeasures.*

II. Discussion of possible countermeasures

Browser fingerprinting usually consists of two key stages. The first stage is obtaining user data through various channels (e.g. JavaScript and plugins). The second stage is to combine all attribute values into one string and then calculate the fingerprinting ID. The countermeasure must select which part should be addressed to defeat the fingerprinting algorithm. Despite fingerprinting technique being a relatively new technique, a number of countermeasures have been recently proposed. This paper classifies these into four main methods and discusses the advantages and disadvantages of each as follows.

A. Blocking access to the user data

Blocking is a normal technique to prevent the fingerprinting algorithm to obtain the user data from a user computer. When the fingerprinting algorithm is incapable of obtaining any value from a user computer, it cannot initiate the creation of fingerprinting ID. Once the fingerprinting ID did not send to the fingerprinting server, the fingerprinting server cannot actually identify the user computer as its purpose. Several research studies had recently inferred the blocking technique as follows.

1) Disabled JavaScript

Eckersley [6] and Boda [8] conducted research in this field and strongly advised that an effective way to inhibit the fingerprinting tracking is to disable JavaScript. JavaScript is the primary mechanism used by a fingerprinting server to access a substantial amount of attributes within the user computer. If the fingerprinting technique is unable to access any attributes, the fingerprint ID cannot be created.

Unfortunately, almost all websites (e.g. Facebook, Google, etc.) now make extensive use of JavaScript. Disabling JavaScript will directly impact the user browsing experience very significantly [3, 10].

2) Do not track

“Do Not Track” is a W3C standard also supported by the EU Cookie Directive [25]. The concept of Do Not Track utilises an HTTP header field with three values: ‘1’ a user does not wish to be tracked, ‘0’ a user allows tracking or ‘null’ a user did not set a preference. This value will be sent with a request message by the web browser to notify the web server to inhibit tracking. If the web server follows this standard, the web browser will not be tracked by the web server. This feature is implemented in all modern web browsers (e.g., Chrome, Internet Explorer or Firefox). However, Acar [26] established that this method is unable to prevent fingerprinting tracking because many fingerprinting servers simply ignore the standard and any requests to inhibit. There is currently no enforcement [27], and this solution can only be effective if policymakers take a decision to require all websites to comply with the standard and an effective enforcement regime is imposed [28].

3) Allowing content before execution

This method allows a user to decide which scripts should be executed on their web browsers. The user can block active content such as Java, JavaScript, Flash and other plugins.

When a user browses any web page, the active content is typically blocked by default. The user then decides which scripts should be executed on their computer. For example, NoScript is a browser extension designed to block all scripted content automatically. The user must then take a decision as to which websites should be trusted.

However, the functionalities of the web page are unlikely to operate properly while rendering a web page. Since scripts are blocked by default, a web page might not operate as smoothly as it should. Another problem is that script running is blocked at the domain level, and many websites load scripts from a large number of different sources. Some websites load only from their own domain, but others load from third-party servers for various reasons (e.g. to display advertisements or to track the user). This complicated operation confuses Internet users without the necessary experience of website technologies. In addition, to recognise that a web page does not contain any threat is significantly more complicated than in the past. Mowery [29] established that some fingerprinting web servers could potentially exploit the utilisation of a whitelist by using the websites allowed in the whitelist as one attribute to create a fingerprinting ID.

B. Creating an identical identity

All computers usually have a different identity. However, if all computers had the same identity, the fingerprinting algorithm could not distinguish which computers belonged to any user. This technique will modify all attributes on the user computer to an identical identity. It does not resist any attempts to access the user data by the fingerprinting server. The attribute values obtained will be faked, and thus the fingerprint ID created by the fingerprinting algorithm will be a duplicated fingerprint ID that will no longer identify the user.

1) Agreement to use common APIs.

One mechanism that helps the fingerprinting web server identify the user computer is the use of a number of different engines for each web browser. There are many vendors involved in the improvement of JavaScript engines on a web browser, and all modern web browsers are equipped with different engines which result in different responses to the web server. The web server can detect the rate of response to the web server and then use the different requests and replies as an additional attribute to fingerprint the user computer [11]. One way to tackle this problem is to force every web browser vendor to use the same set of APIs [12]. If all web browsers use the same standard APIs, the fingerprinting technique cannot distinguish any difference between the request and response messages which eliminates this source of potential fingerprinting. However, this approach appears to be too difficult to implement. Many vendors are unwilling to follow this suggestion because they are concerned about such potential difficulties as the possible lower performance of their web browser [5].

2) Share fingerprinting with others

The unique identity is one of the most significant attributes for the fingerprinting technique. In contrast, assuming if web browser A’s identity is indifferent from web browser B’s identity, the fingerprinting server is incapable of distinguishing web browser A and web browser B. This concept attempts to degrade the unique identity of the web browser by modifying and restricting attributes on the web browser until the web browser’s identity is homogenous. This concept empowers the fingerprinting technique to access selective attributes − it did not restrict access to attributes so that fingerprinting technique can bring accessed information to create the fingerprinting ID. If all web browsers have the same identity, the fingerprinting technique cannot distinguish which users are browsing website due to the same identity. Tor is a great example. The concept of Tor is to modify and restrict many attributes (e.g., a list of fonts, a list of plugins, User-Agent, etc.) on the web browser. The results of attribute modification and restriction cause all Tor users having the same identity [14]. The fingerprinting techniques cannot identify any web browsers because they will all appear to be the same browser. Unfortunately, the speed of Tor is significantly slower than the speed of regular web browser due to nature of its connections. In addition, Tor hides user's location and randomly pick server's location which can cause problems of login system [15], and Tor users are treated as with the second class on the Web, frequently being denied services by many websites [16].

Tor is an open network which was designed to secure the user privacy by concealing the user's location [36]. Tor not only can hide the user's location but also can protect the entire information being transferred using encryption [17, 18]. Tor sends user information through a number of Tor servers located around the world, provided by a voluntary basis (to provide a public service). Each server is designed to transfer information to another server – this is called a relay.

The Tor browser has been designed to be straightforward for the general user to use [19, 20] and modified many attributes on the Firefox browser in order to preserve the user privacy, below [21].

- Restricted JavaScript code and HTTP header
- Size alteration of possible window dimensions
- Reading browser history not supported
- Clearing cookies and DOM storage
- Changing user-agent and Plugins disabled

Tor has been praised by many researchers because of the high levels of privacy provided. For example, Eckersley [6] (who introduced the concept of fingerprint tracking on the Internet) noted that the Tor browser provides an effective defence against fingerprinting.

The low effective bandwidth of Tor (noted by Lenhard [15]) results in a slow speed which is the most significant problem when using Tor (Buder [22]).

C. Degrading uniqueness of fingerprinting ID

The uniqueness of the fingerprinting ID normally depends on attributes detected on the user computer and the fingerprinting web server will attempt to derive high-entropy values for the user computer. If the uniqueness can be decreased, then the ability to track is also decreased.

1) Decreasing use of plugins

The list of plugins in a web browser provides a relatively-high-entropy attribute, around 15.4 bits of entropy as measured by Eckersley [6]. This list is significant not only because it makes an important contribution to the attributes available to a fingerprinting web server, but it also reveals underlying user information through plugin APIs, (e.g. a list of fonts, operating system, screen resolutions, etc.). The Tor browser decreased the use of plugins by allowing only selected plugins, hence reducing leakage of user information. Another example is a smartphone browser which typically allows only a few plugins. This implicitly reduces the entropy value, and it is harder for a fingerprinting website to create a unique tracking ID [6]. Boda [13] noted that the increasing number of plugins had improved the robustness of fingerprinting algorithms, hence decreasing the number of plugins should help to prevent fingerprint tracking. In addition, Natalia [34] explained that the inability to install plugins on a mobile phone makes fingerprinting harder as well as Nikiforakis [5] who concluded that regardless of the complexity of plugins there was an inability to control user data.

Although degrading the stability of fingerprinting by decreasing the number of plugins appears to be an attractive method, the consequence of disabled plugins can produce adverse effects on the browsing experience. Any plugin deficiency can result in the problems affecting the smooth and proper running of the web page. For example, installing a PDF plugin will assist the user to view PDF documents conveniently. If the browser did not provide a built-in PDF facility, users would be unable to view PDF files easily.

2) Using multiple browsers

This approach makes use of different browsers for different activities when surfing the Web. User profiles will be more difficult to create if the user makes use of more than two browsers. For example, suppose one browser is used only for Facebook, Google+ and e-commerce, and another one is used for general browsing. This concept assumes that the user profile is more difficult to build up if users can change their web browser. Despite the seemingly-straightforward concept, it provides only a partial solution because a fingerprinting web server can select other attributes unrelated to a web browser (e.g. operating system, the underlying hardware, screen resolution or the IP Address) to fingerprint the user computer rather than relying on browser attributes [8].

D. Using extension

Modern web browsers (e.g. Firefox, Internet Explorer and Chrome) have allowed developers to extend browser functionality for various purposes. They provide online stores which permit users to find available extensions. Typically browser extensions are used for enhancing security, inhibiting advertisements, and adding new features. Browser extensions can be used to enhance user privacy by monitoring and intercepting activities (i.e., it can inject the Logger component into the Document Object Model (DOM)) on the web browser while the web page has begun loading. A number of fingerprinting countermeasures with different techniques have been implemented using browser extensions as it is straightforward for the user to download and use them. Because each user computer has various attributes and each attribute has a different entropy value, this helps to classify the current fingerprint countermeasures used or proposed to the public.

1) Selecting high entropy attribute

High-entropy attributes are a significant factor when creating a unique tracker ID for fingerprinting. Several research studies have considered the characteristics of user-computer attributes. The results all show [6, 8, 30] that the list of fonts, a list of plugins and user agent provide high-entropy attributes. Using only a list of plugins, for example, has a sufficiently–high entropy (15.4 bits) to generate a 1-in-43,237 fingerprint. The fingerprinting webserver will typically select additional attributes which might be low-entropy attributes (e.g. operating system, screen resolution, etc.) to create a unique tracker ID with sufficient stability. This is a key reason why many countermeasures focus on reducing the value of high-entropy attributes.

It is highly likely that most fingerprintering websites will make use of these high-entropy attributes in their algorithms [5].

a) User-agent spoofing

All Internet users who are browsing the web have their own user agent. A user agent contains many attributes within one string of the user agent (e.g. browser name, version of browser, operating system, CPU, etc.) Spoofing the user agent not only assists the user to view a website properly as originally designed, but it also helps a webmaster to test the web resources with other user agents. Yen [31] suggests that changing the user agent is an approach to avoid fingerprint tracking as it makes fingerprinting less unique. A number of extensions have been recently made available by developers (e.g., UserAgent Switcher, UserAgent RG, UAControl, UserAgentUpdater, etc.) for various purposes. One approach is to alter user agent information to inhibit fingerprint tracking. Nikiforakis [5] noted that using user agent spoofing often results in unwanted side-effects because it increases the risk of being fingerprinted by somebody. If user agent spoofing reports inconsistent data continually, this can lead to an impossible configuration (e.g. mismatching of the user agent information with the JavaScript layer) which ironically could result in an easier identification of the web browser [6, 32].

b) Privaricator

Privaricator [14] employs the principle of random policies. It assumes that the user will visit websites for the first time. Visiting a website with the same browser the information returned will be altered in a random fashion every time so that each visit appears to be from a new user. This technique focuses on the list of plugins and list of fonts by using the method of randomness to hide plugins and fonts and also adds some random noise to the offset properties. However, because it adds noise to the offset properties this may influence the rendering of the web page, and random plugins may involve the execution of code in the Internet page. For example, multimedia web pages often require the Flash plugin. If the web browser does not support Flash, it cannot render the web page correctly. Moreover, Privaricator does not prevent the use of the Flash plugin which is capable of exposing the list of fonts [33]. In addition, this countermeasure pays attention to high-entropy attributes and the remaining attributes might provide a soft target for fingerprinting websites.

2) Ignoring actual entropy values

One of the most significant current discussions is how to prevent different types of fingerprinting technique as much as possible. As a consequence of the previous method, it appears that changing only the high-entropy attributes might not be sufficient to prevent fingerprint tracking. On the application layer, JavaScript might allow the user to be fingerprinted effectively even without the use of high-entropy attributes such as canvas fingerprinting, JavaScript objects, etc. This countermeasure seeks to address this problem by altering all attributes irrespective of the entropy value.

a) RubberGlove

RubberGlove is a Chrome extension which can be installed from the Chrome web store. It finds JavaScript objects within a web page (navigator and screen) and then replaces them with null values. This method blocks the access of JavaScript objects which cannot then be used to create a fingerprinting ID. However, the initial study had noted that the Rubberglove has a significant impact to the user browsing experience [37].

3) Selecting appropriate attributes

During browsing web pages, the browsing experience is a crucial factor for the user. The two previous approaches (selecting high-entropy attributes and ignoring entropy values) did not consider the consistent combination after changing attributes. There is some evidence from Eckersley [6] that inconsistent combinations of web browser attributes may result in a web browser being easily fingerprinted.

a) FP-Block

FP-Block uses randomization, blocking, and spoofing techniques to change the identity of the user computer. FP-block allows separation of web identities by changing the identity of the user computer if the user browses to a different website. In addition, it offers a model to change the identity of the user computer by requiring a consistent combination after the user computer identity change and also examines the relationship between the HTTP header and JavaScript APIs. However, FP-Block is not designed to protect at runtime - users have to make the decision to alter the identity of their computers before they first visit any websites.

b) Fireglove

Fireglove is a Firefox extension specifically designed to prevent browser fingerprinting. Fireglove spoofs the user agent and platform, returns a random value for the offsetWidth and offsetHeight parameters and disables both plugins and mimeType. Fireglove restricts loading of fonts and offset value by intention to prevent font fingerprinting. All of this is done in order to obfuscate fingerprinting algorithm at runtime. However, Acar [26] discovered that they could use another instruction (getBoudingclientRect) to disclose the value of offsetWidth and offsetHeight and the fix of a number of loaded fonts is not workable as suggested.

c) CanvasFingerprintBlock

CanvasFingerprintBlock is a Chrome extension which is specifically designed to block canvas fingerprinting by detecting canvas script with finding keywords. It detects .toDataURL() which is an instruction to write a canvas element. If it finds a suspicious keyword, it will send a message to notify the background page and display a deleted symbol on the Chrome browser.

d) FPGuard

FPGuard [30] implements a number of algorithms to analyse fingerprinting at runtime. Different techniques are then used to prevent each type of fingerprinting by randomization and blocking techniques. This system will examine each type of fingerprinting and then attempt to change attributes using various techniques. For example, if it discovers fingerprinting with object fingerprinting, it will randomise the attributes. Another example is if it detects fingerprinting with a list of plugins, it will generate virtual plugins to confuse the fingerprint detection. It is possible that some of these techniques might be counterproductive if they permit easier fingerprinting by ignoring the consistency of attributes.

III. CONCLUDE POSSIBLE WAYS

Users are not certain which countermeasures are appropriate to prevent the loss of privacy from fingerprint tracking with a minimum impact on their browsing experience. This section will propose an optimal approach to inhibit fingerprinting tracking.

Many proposed tracking countermeasures appear to be difficult to implement in practice and also lack efficient prevention. Firstly, blocking access to the user data has a very significant adverse impact on the browsing experiences, relying only on the web server and is effectively impossible to use in the real world. For example, disabling JavaScript is very hard to achieve in practice [3, 10] because almost all websites require JavaScript to run. Even though users can prevent fingerprinting tracking, if JavaScript is disabled users will face the inevitable consequences of serious loss of functionality. Acer [26] conducted an experiment (Do not Track, DNT) which confirmed that DNT is ignored by fingerprinting websites. Any agreement to use common APIs will take a long time to achieve because each modern web browser wishes to develop their own browser in their own way [5]. Decreasing use of plugins did not solve the entire problem of fingerprint tracking, it just decreased the uniqueness of fingerprinting while the fingerprinting web servers still fingerprint the user computer. Using multiple browsers is still vulnerable to attack by fingerprinting techniques because fingerprinting websites can use other attributes unconnected with browser attributes to create a unique tracker ID [13]. Sharing a fingerprinting ID with other users seems to be a useful idea but none of the research currently shows a definitive result that this is a useful method to prevent fingerprinting. In addition, the efficiency of fingerprinting prevention depends on the number of users sharing the same fingerprint. If the number of users using this method decreased, the users have a renewed risk of being fingerprinted. All of these problems with the current method can be addressed using a browser extension and there are several reasons to support this proposal.

1) It is unnecessary to disable JavaScript

Loading of web pages can be intercepted and detected by using the browser extension. The web page modified can run JavaScript as same as an unmodified web page. This is a excellent solution to enhance the browsing experience and still prevent the fingerprint tracking as user are using.

2) Solving the fingerprinting problem with a custom method

As pointed out earlier, it is unrealistic to expect modern web browser implementers to negotiate and agree to use a common API [5]. The proposal is to implement a custom method rather than expecting someone else to tackle the problem of fingerprinting.

3) Possibility of addressing all fingerprinting problems

Many countermeasures have been made based on using a browser extension [14, 30] to deal with the problems of the fingerprinting technique. This solution seems to tackle entire problems of the fingerprinting technique rather than reducing plugins on the web browser.

4) Improving the browsing experience

Installing a browser extension does not affect the services of typical websites. A user will be treated exactly as a normal user. It contrasts with the Tor browser which provides a “second class” service to users when browsing the Internet.

IV. DISCUSSION ABOUT FUTURE COUNTERMEASURES

Surprisingly, almost all of the countermeasures that have been proposed based on browser extensions do not provide detailed results and discuss how to measure the efficiency of fingerprinting prevention as well as how to measure any web browser side-effects. Almost all methods pay attention only to fingerprinting prevention and overlook any impact on the user experience. In practice, both the measurement of fingerprinting prevention and the impact on the user experience are very important, especially when a minimum impact to the user is required. The FP-block approach is the only method to consider both prevention and user experience. Therefore, the new countermeasures should enhance the principle of randomising attributes on the web browser before the fingerprinting algorithm gain the user data in order to avoid the web browser stand out rather than arbitrary randomness. In addition, attributes on the HTTP header field should be reviewed according to attributes on the JavaScript in assisting to circumvent the information paradox.

It should be noted that it is extremely difficult to predict all possible events which could lead to the generation of the same fingerprint ID. Different countermeasure approaches can bring their countermeasures to test with the real fingerprinting websites. If the fingerprint ID changes, the assumption will be that they can prevent fingerprint tracking. However, it is not clear how many fingerprinting websites need to be tested to determine that technique can completely prevent fingerprinting tracking. This is a more controversial issue which requires further research.

V. CONCLUSION

A number of possible methods to prevent browser fingerprinting have been proposed in this paper. Implementing the recommended options depends on the individual preference of the Internet user. This paper proposes the use of a browser extension that could potentially achieve more than the other countermeasures currently available. In addition, the paper proposes that the new countermeasures based on the browser extension should address the important issues of preventing all types of fingerprinting and reducing any adverse side-effects on the user browsing experiences.

REFERENCES

[1] BlueCava. "BlueCava Opt-out Preferences," 10/05/2015; http://bluecava.com/opt-out/.

[2] Iovation. "Multifactor Authentication and Online Fraud Prevention Solutions | iovation," https://www.iovation.com/.

[3] F. Roesner, T. Kohno, and D. Wetherall, "Detecting and defending against third-party tracking on the web," in Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, 2012, pp. 12-12.

[4] J. Caldera, J. M. Hain, and K. Sherlock, "Enhanced Automated Anti-Fraud and Anti-Money-Laundering Payment System," ed: US Patent 20,160,071,108, 2016.

[5] 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, pp. 541-555.

[6] P. Eckersley, "How unique is your web browser?," in PETS'10 Proceedings of the 10th international conference on Privacy enhancing technologies Springer-Verlag Berlin, Heidelberg, 2010, pp. 1-18.

[7] M. Egele, P. Wurzinger, C. Kruegel, and E. Kirda, "Defending browsers against drive-by downloads: Mitigating heap-spraying code injection attacks," in Detection of Intrusions and Malware, and Vulnerability Assessment, 2009, pp. 88-106.

[8] K. Boda, Á. M. Földes, G. G. Gulyás, and S. Imre, "User tracking on the web via cross-browser fingerprinting," in Information Security Technology for Applications, ed: Springer, 2012, pp. 31-46.

[9] D. Fifield and S. Egelman, "Fingerprinting web users through font metrics," in In: Proceedings of the 19th international conference on Financial Cryptography and Data Security., 2015, 10 pages.

[10] P. Chairunnanda, N. Pham, and U. Hengartner, "Privacy: Gone with the typing! identifying web users by their typing patterns," in Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on, 2011, pp. 974-980.

[11] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham, "Fingerprinting information in JavaScript implementations," Proceedings of W2SP, vol. 2, 11 pages, 2011.

[12] R. Upathilake, Y. Li, and A. Matrawy, "A classification of web browser fingerprinting techniques," in New Technologies, Mobility and Security (NTMS), 2015 7th International Conference on, 2015, pp. 1-5.

[13] K. Boda, Á. M. Földes, G. G. Gulyás, and S. Imre, "User tracking on the web via cross-browser fingerprinting," Information Security Technology for Applications, vol. 7161, pp. 31-46, 2012.

[14] N. Nikiforakis, W. Joosen, and B. Livshits. "Privaricator: Deceiving fingerprinters with little white lies," 25/06/2015; http://research.microsoft.com/pubs/209989/tr1.pdf.

[15] J. Lenhard, K. Loesing, and G. Wirtz, "Performance measurements of Tor hidden services in low-bandwidth access networks," in Applied Cryptography and Network Security, 2009, pp. 324-341.

[16] S. Khattak, D. Fifield, S. Afroz, M. Javed, S. Sundaresan, V. Paxson, *et al.*, "Do you see what i see? differential treatment of anonymous users," in *Network and Distributed System Security Symposium*, 2016.

[17] A. Ruiz-Martínez, "A survey on solutions and main free tools for privacy enhancing Web communications," Journal of Network and Computer Applications, vol. 35, pp. 1473-1492, 2012.

[18] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, "Hiding routing information," in Information Hiding, 1996, pp. 137-150.

[19] J. Clark, P. C. Van Oorschot, and C. Adams, "Usability of anonymous web browsing: an examination of tor interfaces and deployability," in *Proceedings of the 3rd symposium on Usable privacy and security*, 2007, pp. 41-51.

[20] D. Abou-Tair, L. Pimenidis, J. Schomburg, and B. Westermann, "Usability inspection of anonymity networks," in Privacy, Security, Trust and the Management of e-Business, 2009. CONGRESS'09. World Congress on, 2009, pp. 100-109.

[21] N. Schmücker, "Web Tracking," in SNET2 Seminar Paper. Berlin University of Technology, 2011, p. 12 pages.

[22] G. Aggarwal, E. Bursztein, C. Jackson, and D. Boneh, "An Analysis of Private Browsing Modes in Modern Browsers," in USENIX Security'10 Proceedings of the 19th USENIX conference on Security 2010, pp. 79-94.

[23] H. Said, N. Al Mutawa, I. Al Awadhi, and M. Guimaraes, "Forensic analysis of private browsing artifacts," in Innovations in information technology (IIT), 2011 International conference on, 2011, pp. 197-202.

[24] 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.

[25] H. Tschofenig and R. van Eijk. (2011, 21/06/2015). DO NOT TRACK. Available: http://www.w3.org/2011/track-privacy/papers/Tschofenig.pdf

[26] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, et al., "FPDetective: dusting the web for fingerprinters," in Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, 2013, pp. 1129-1140.

[27] R. Balebako, P. Leon, R. Shay, B. Ur, Y. Wang, and L. Cranor, "Measuring the effectiveness of privacy tools for limiting behavioral advertising," in WEB 2.0 SECURITY & PRIVACY 2012, 2012, 10 pages.

[28] R. Broenink, "Using Browser Properties for Fingerprinting Purposes," in 16th biennial Twente Student Conference on IT, Enschede, Holanda, 2012.

.[29] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham, "Fingerprinting information in JavaScript implementations," in Proceedings of W2SP 2011, vol. 2, pp. 1-11, May, 2011, 2011.

[30] A. FaizKhademi, M. Zulkernine, and K. Weldemariam, "FPGuard: Detection and Prevention of Browser Fingerprinting," in Data and Applications Security and Privacy XXIX, 2015, pp. 293-308.

[31] T.-F. Yen, Y. Xie, F. Yu, R. P. Yu, and M. Abadi, "Host Fingerprinting and Tracking on the Web: Privacy and Security Implications," in NDSS, 2012, 16 pages.

[32] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "On the Workings and Current Practices of Web-Based Device Fingerprinting," in Security & Privacy, IEEE, 2014, pp. 28-36.

[33] P. Laperdrix, W. Rudametkin, and B. Baudry, "Mitigating browser fingerprint tracking: multi-level reconfiguration and diversification," in Proceedings of the International Symposium on Software Engineering for Adaptive and SelfManaging Systems (SEAMS'15), 2015, pp. 98-108.

[34] N. Karbasova. (2013). Your browser's 'fingerprints' and how to reduce them | Digital Safety. Available: http://akademie.dw.de/digitalsafety/your-browsers-fingerprints-and-how-to-reduce-them/

[35] J. Samuel and B. Zhang, "RequestPolicy: Increasing web browsing privacy through control of cross-site requests," in International Symposium on Privacy Enhancing Technologies Symposium, 2009, pp. 128-142.

[36] TOR. 2015. Tor Project: Anonymity Online [Online]. Available: https://www.torproject.org/ [Accessed 27/04/2015].

[37] E. Z. Sakchan Luangmaneerote, Leslie Carr, "Survey of existing fingerprint countermeasures," presented at the Information Society (i-Society), 2016 International Conference on, Dublin, Ireland 2016.