Browser Fingerprinting as User Tracking Technology

Navpreet Kaur, Sami Azam
School of Engineering and Information Technology, Charles Darwin University
Ellengowan Drive, Casuarina 0810
NT, Australia
navbajwa86@gmail.com, sami.azam@cdu.edu.au

Krishnan Kannoorpatti, Kheng Cher Yeo, Bharanidharan Shanmugam,
School of Engineering and Information Technology, Charles Darwin University
Ellengowan Drive, Casuarina NT 0810
krishnan.kannoorpatti,charles.yeo, bharanidharan.shanmugam@cdu.edu.au

***Abstract*—The Web has become an indispensable part of our society and is currently most commonly used mode of information delivery. Millions of users access the free services provided by the websites on daily basis and while providing these free services websites track and profile their web users. In this environment, the ability to track users and their online habits can be very lucrative for advertising companies, yet very intrusive for the privacy of users. The objective of this paper is to study about the increasingly common yet hardly discussed technique of identifying individual Web users and tracking them across multiple websites known as "Browser Fingerprinting".**

A unique browser fingerprint is derived by the unique pattern of information visible whenever a computer visits a website. The permutations thus collected are sufficiently distinct that they can be used as a tool for tracking. Unlike cookies, Fingerprints are generated on server side and are difficult for a user to influence. The main objective of this research is study about how the fingerprinting was evolved, its positives and negatives, what threat it poses to users' online privacy and what countermeasures could be used to prevent it. This paper will also analyse which different properties the browsers send to the server, allowing a unique fingerprint of those browsers to be created.

***Keywords*—broweser, fingerprinting, web, tracking**

I. INTRODUCTION

The World Wide Web had totally revolutionized the ways we communicate in just few decades. The number of people and the time they spend browsing web has grown exponentially. Consequently, the technologies to enhance the web user's online experience are evolving at great pace. Every evolution has its own purpose like JavaScript libraries and HTML 5 provides more dynamic and interactive web, mobile devices provides more available web, SSL and TSL provides more secure web, browser add-ons such as Ghostery and Adblock provide more private web. In fact, the web browsers themselves are rapidly changing and have become more challenging testing grounds for various new technologies [1].

But like every coin has its flip side, the modern browser technologies that provide beauty, power and richness to Web services, also provide an evil side too. The sophisticated looking websites asks the browser for hardware and software configurations and the browser freely provides all the information thus allowing the websites to better exploit the user's resources. The evil side of websites analyse the small differences between user's systems and create a unique code out of it known as Fingerprint. Now every time the user visiting the same website will be identified by that assigned unique fingerprint. This practice of tracking users using their system configurations jeopardize their privacy and comfort [1].

II. BACKGROUND

While working with Netscape communications in 1994, Lou Montulli introduced the concept of cookies for maintaining the state of session in stateless HTTP protocol. Due to their simple nature and implementation, cookies were embraced both by browser vendors and developers. But, shortly after their introduction, the abuse of their stateful nature was observed. Web servers started using them for third party advertisement and tracking users online which caused public's discomfort [2].

The users who were concerned about their privacy responded to this privacy threat in many ways. Various add-ons such as Ghostery and Adblocker got immense popularity. A study by MediaPost showed that 44 percent of users Opt-Out of targeted ads, 65 percent of users delete both third party and first party cookies and 39 percent of users change their browser settings to request the websites not to track them [3]. Many users switched to "Private Mode" of browsing in which no traces of user history or cookies are left behind once the browser is shut down. The Cookie Law according to which consent of user is required in order to place a cookie on the local memory of machine also prevented users from being tracked by websites to some extent [2].

Thus unavailability of cookies motivated advertisers and trackers to come up with new ways to link the users with their online browsing habits. So, Mika D. Ayenson introduced the idea of Zombie Cookie in which websites not only place the cookies on the normal location of cookies (in the browser memory) but also place them in many different locations (in form of Flash cookies and HTML5 Storage) on user's machine [4]. Even though the user deletes normal cookies from the browser, Zombie cookies still can be used to regenerate the deleted cookies once the user visits the same site to which the cookies belongs to and thus continues the never ending process of tracking [4]. The process of regeneration of normal HTTP cookies from the Adobe flash cookies is shown in Fig. 1.

978-1-5090-2717-0/17/$31.00 ©2017 IEEE

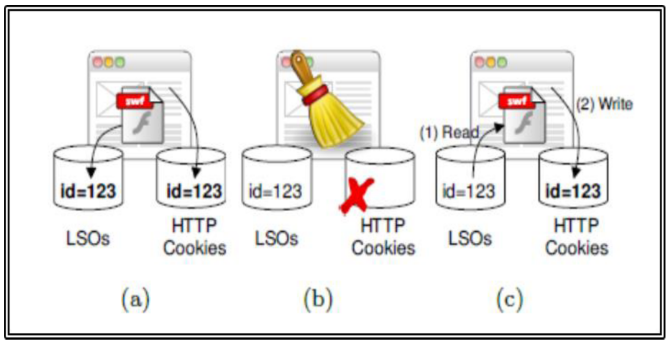

Figure 1: Regeneration of Cookies [5]

A. Webserver stores HTTP Cookies and Flash Cookies (LSO)

B. User deletes HTTP cookies

C. The webserver recreates HTTP cookies by copying value from Flash cookies

But if user deletes the entire zombie as well as normal cookies after certain period of time, the process of tracking stops which is again a matter of concern to advertisers and trackers. So this shortcoming of cookies fueled the research of passive methods of identifying users.

Another method of identifying and tracking users that was proposed is Browser Fingerprinting and the people who have major contribution in the evolution of this technology are Jonathan Mayer, Peter Eckersley and Károly Boda [6][9][7]. We will now discuss the contribution of each of them and what was their shortcoming which ignited the further research.

Evolution of Browser Fingerprinting

In 2009, Mayer used JavaScript's navigator object and screen object to generate unique identifier by using Hash function. He does experiment on 1300 browsers and as a result he found that he could uniquely identify 96 percent them. Fig. 2 can explain the JavaScript based fingerprinting is more detail [6].

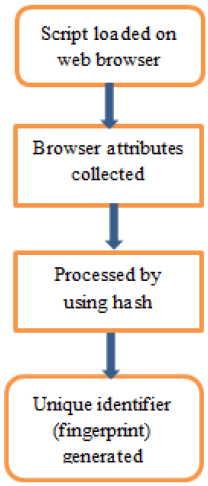

Figure 2: JavaScript-based Device Fingerprinting Activity

As per Fig. 2, JavaScript is loaded on the client side; JavaScript specific attributes of browser are collected and a unique identifier is created by using hash function.

Later in 2010, Peter Eckersley, of Electronic Frontier Foundation does the same but extended research; he identified users on the basis of User Agent String, parameters from the HTTP request, plug-in list, the time zone, the screen resolution, the installed fonts and some persistent cookies available [7]. He named the project as Panopticlick and the purpose of this project was to show that features of a browser and its plugins have unique set of attributes which when combined together can be used to track users without the need of cookies [2] [8].

In the Panopticlick Project, they implemented one fingerprinting algorithm and scanned 470,161 web browsers and they observed that every browser contained at least 18.1 bits of entropy which means if one browser is picked randomly then there will be high probability that only one browser in 286,777 will share its fingerprints. They also observed that browsers who support java and flash contained 18.8 bits of entropy which means 94.2 percent of browsers with flash and java had unique fingerprints [9].

They also scanned the returning browsers to estimate how frequently browsers changed their fingerprints due to installing some updates and they found that browsers changed their fingerprints very rapidly. Even though the browsers changed their fingerprints frequently still some very simple experimentation was enough to guess when they were upgraded and to match them with the previous recorded fingerprints, 99.1 percent of their matching guesses were correct and the failure rate was only 0.86 percent [9].

The HTTP referrer Headers, Java and Flash attributes that Eckersley used to generate unique Fingerprint are shown in Table 1.

Table 1: Attributes used by Eckersley [9]

Variable	Source	Remarks
User Agent	Transmitted by HTTP, logged by server	Contains browser micro-version, OS version, language, toolbars and sometimes other info.
ACCEPT Headers	Transmitted by HTTP, logged by server	
Cookies Enabled?	Inferred in HTTP, logged by server	
Screen Resolution	JavaScript AJAX post	
Time zone	JavaScript AJAX post	

Browser Plugins, Plugin version and MIME type	JavaScript AJAX post	Sorted before collection. Microsoft Internet Explorer offers no way to enumerate plugins; They used the PluginDetect JavaScript library to check for 8 common plugins on that platform, plus extra code to estimate the Adobe Acrobat Reader version
System Font	Flash applet or Java applet, collected by JavaScript/AJAX	Not sorted
Partial super cookie (Zombie or persistence cookie) test	JavaScript AJAX post	They did not implement tests for Flash LSO cookies, Silverlight cookies, HTML5 databases, or DOM global Storage.

In Table 1, the variable column mentions the attributes and source column describes the JavaScript or HTTP header attributes used to fetch the value. "Transmitted by HTTP" means that the value was sent by the browser in the normal HTTP conversation, "Logged by server" means the value was recorded by the server, "JavaScript Ajax Post" means JavaScript code was needed to collect the information and post it back to server for processing, "Flash Applet or Java Applet" means either Java or Flash was required to collect that information [21].

Further in 2011, Boda et al. identified that the major drawback in Panopticlick project was its reliance on Browser instances and either Java or Adobe Flash (the attributes with highest entropy) must be enabled to get the list of fonts. To avoid this weakness Boda et al. proposed a new solution in which they omitted the browser specific details and used JavaScript, some basic system fonts to identify fonts that are browser independent and installed without the need of Java or Flash, system features (Operating system, screen resolution) and the first two octets of the IP address. The main motive of this proposal was to create a unique fingerprint that is browser independent, plugins independent and provide cross browser identification of user. They tested the proposed solution on 989 visitors and managed to identify 28 percent of them who were using multiple browsers on same computer [7] [10]. The User Agent when combined with IP address increase the entropy to 20.29 bits from 18.1 bits which was entropy of whole fingerprint as identified by Eckersley in 2010.

Later on more granular approach for detecting a returning browser was done by Stocks and Broenink in 2012 and they accurately identified 86% of the returning browsers [10]. They created the rule set for recognizing the same browser over time which is shown in Table 2.

Table 2: Attributes used by Eckersley [13]

Property	Assumpted Rules
Accept	does not change
Accept-Language	does not change (*)
Accept-Encoding	does not change
Accept-Charset	does not change
Connection	does not change
User-Agent	browser name does not change, browser version does not decrease
DNT	does not change (*)
JavaScript enabled	does not change (*)
JavaScript version	does not decrease
Platform	does not change
Charset	does not change
Language	does not change
Cookie support	does not change (*)
Java support	does not change
Screen resolution	does not change (**)
Time zone	does not change (Daylight Saving Time correction)
Plugin versions	does not decrease
Font list order	does not change, fonts are not removed, fonts may be added in between
* attributes the can be easily amended by the user, ** screen resolution effected by external monitors being plugged in	

III. FINGERPRINTING AS A SESSION PROTECTOR

Constructively, JavaScript engine fingerprinting can be used by webservers to protect against session hijacking. For session attack, session hijackers usually clone all the possible identifying information such as Http Headers or plain text session cookies (using FireSheep) and by using JavaScript Engine Fingerprinting such attacks can be detected at the server side, as the modified user agent can be detected (Web-Server just not rely on HTTP headers for all identifying information but it uses JavaScript scripting language too). For instance, by cloning all the HTTP header and cookies information of legitimate user by using tools like ModHeader and FireSheep if the attacker pretends to be the genuine user then he can be easily identified by the server as the information related to

browser in the HTTP header will not match with the values that the server got querying the browser using JavaScript [11] [2].

Constantly challenging the browser and adding JavaScript engine Fingerprint as an additional security layer would be another way to secure the HTTP sessions. In the beginning of every session, the browser is identified with the minimal fingerprinting and for all the following requests the webserver selects some random test cases and include them in JavaScript code, thus challenge the client with every request. The overhead caused by this challenging would be very minimal and non-noticeable by the client. If the response sent by client does not relate with the desired outcome, the session is terminated. Even if the attacker is able to see the challenges, he might not know the correct responses. So in order to keep the hijacked session alive the attacker will be forced either to use the same or nearly same version browser as that of genuine user or collect the fingerprints of all the replies of the victim beforehand but both the options are not feasible [11]. A machine correctly identified during session can prevent online frauds, for instance by finding that the user trying to login with accurate credentials is actually and attacker with stolen credentials or cookies [18].

Thus JavaScript fingerprinting raises the bar of protection against session hijacking. This method could also be used along with HTTPS sessions to avoid Man in the middle attack as HTTPS alone is not enough to provide secure online communication [11].

IV. Fingerprinting as Threat

Fingerprints as Global Identifier

Privacy advocates show that JavaScript, Adobe Flash and various other add-ons of browser can be used to track users across session without the need of cookies. Attackers also use Browser fingerprints to deliver exploits designed for a particular set of browser configuration [12]. There is no easy way to opt-out of this privacy threat and it is even harder to say if the fingerprinting servers respect the "Do Not Track" or DNT identifier of the browsers [2].

Fingerprint + IP address as Cookie Regenerators

Many websites use Adobe Flash cookies as a way to regenerate normal cookies when the user deletes them. Fingerprinting also pose the similar threat of cookie regeneration even if those fingerprints are not globally identifying. Fingerprints usually carry 18-20 bits of identifying information which is enough to uniquely identify a browser. Some websites also use the IP address or its subnet to create a unique fingerprint and when the user deletes all the cookies but continues to use the same IP address, there is high probability to get the cookies regenerated or linked to previous ones.

V. Classification of Browser Fingerprinting Techniques

A unique Web Browser fingerprint can be created by different methods such as by using JavaScript, Plugins, Browser Extensions and Headers. We will now discuss about all the techniques one by one.

A. Header Based Fingerprinting

HTTP Headers: Whenever a Web browser sends an HTTP request to Webserver, it has to at least send its protocol version, requested path and requested host with every request [13]. But it also sends some more information too which can be used for technical purpose. For instance, User-Agent can be used to identify the capabilities of the browser and also resolve the browser specific debugging issues. Yet, it can also be used to identify the browser by its name, version and platform [13].

Table 3 shows the attributes of HTTP Headers that are sent by almost every Web-Browser and provides enough information to websites thus allowing them to be used for tracking purpose.

Table 3: Http Header attributes [20]

Http Header Attribute	Information Provided
ACCEPT	It is used to define the data types supported by the browser. **Accept: text/plain, text/html**
ACCEPT-LANGUAGE	This header is used to announce the preferred human language for the response. **Accept-Language: en-us, en**
ACCEPT-ENCODING	This header is used to announce the encoding capabilities of the Browser. **Accept-Encoding: gzip, deflate**
ACCEPT-CHARSET	It is used to announce the character sets that are acceptable by the browser. **Accept-Charset: utf-8**
CONNECTION	It is used to specify what will happen to the connection when the request has been completed.
USER_AGENT	Specifies the browser and platform information. **User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.1.7) Gecko/20100106 Ubuntu/9.10 (karmic) Firefox/3.5.7**

Some users even set the Do Not Track flag in their browsers and this user preference is sent to Webserver through DNT header. This particular header is used to announce the user's preference is not to track the browser. Even though, the websites may respect the preference but it can be used to identify the user more uniquely as very small subset of users has this header set [13].

As shown in Table 4, combination of HTTP headers contains 7.37 bits of entropy and the User-Agent attribute has highest entropy among all other headers. The attributes with

higher entropy are the ones revealing important information of browser to the websites. Researchers have found that User-Agent when combined with IP address of machine may help in tracking user with high precision [14].

Table 4: Entropy related with HTTP Header Attributes [13]

Header	Entropy (bits, all visits)	Entropy (bits, unique visitors)
Accept Accept-Language Accept-Encoding Accept-Charset Connection User-Agent DNT	1.73 4.16 1.67 1.86 0.28 6.01 0.51	2.14 4.25 1.84 1.83 0.37 6.31 0.57
HTTP Headers combined Entropy	**6.99**	**7.37**

The User-Agent string reveals a lot of information about the browser and system architecture (software and hardware) and the installation of certain plugins further increase the uniqueness of User Agent String. Even if multiple people share same IP address (for example, family members, office workers), they can be differentiated just by the User agent string if they are not using same browser with same version. In this way User-Agent String can be used as a local ID and can help in differentiating users with same IP address [7].

B. JavaScript Based Fingerprinting

JavaScript has been known as a de-facto client side scripting language since its invention in 1995. Its expressiveness, forgivable nature (negligence towards syntax errors) and availability in most modern browsers have made it a crucial tool for modern web sites [14]. A study by GOV.UK in 2013 showed that only 1% of actual human visitors disable the JavaScript in their browsers [15].

The main purpose of using JavaScript as a client side language is to dynamically manipulate the Document Object Model (DOM) of page, enrich the user experience through asynchronous requests and responses and to balance the load on server by transferring non-critical server functionality to the client. Its presence in most of the web browsers also makes it a very effective fingerprinting tool [14].

Navigator and screen objects are the most probed JavaScript resources.Navigator object contains information regarding browser vendor, browser version, installed plugins, time zone and MIME (Multipurpose Internet Mail Extensions) type (such as language, charset) of the browser. It can also provide fine grained information about the operating system and architecture (using Platform attribute) on which the browser is executing [14].

Table 5: The attributes of JavaScript that can be used in the process of Fingerprinting

Attribute	Information Provided
JavaScript Version	It is the consistent value for a specific browser vendor. Most of the times it remains the same but they may increase with any upgraded version installation. Different browser vendors have different versions of JavaScript.
Platform	It can be used to distinguish between the Operating System. Platform attribute can be used to identify for which the browser was built. Win32 and Win64 are the common examples.
Charset	Depending on the browser settings, this attribute specifies the default decoding setting. This value could vary from the one that was got using the headers as this determines the rendering of page whereas header just declares the capabilities of the browser. Example: <script type="text/javascript" src="path/something.js" charset="utf-8"></script>
Language	This attribute specifies the interface language of the browser which could be is different from HTTP Header Value (determines user preferred language) as this is the interface language.
Cookie Support	It is used to determine if the browser supports the HTTP cookies.
Java Support	It is used to determine if Java is supported by the client.
Time Zone	It is used to specify the current time zone of the computer's clock.
Screen Resolution	It is used to determine the screen height and width, color and pixel depth of user's monitor.
Plugin versions	It is used to retrieve the array of installed plugins. All browsers, except Internet Explorer, announce the plugins that are installed.

Screen object provides the information about the resolution, the color and pixel depth of user's monitor [14].

Mayer in 2009, just used navigator, screen, navigator.plugins, navigator.mimeTypes to uniquely identify more than 96% of the browsers [14]. The research by Ralph Broenink has shown that the JavaScript attributes carries Entropy of 7.47 bits (Refer Table 6). Higher the Entropy more revealing the web browser is.

Table 6: Entropy recorded in JavaScript Fields [13]

Field	Entropy (bits, all values)	Entropy (bits, unique visitors)
JavaScript enabled	0.87	0.79
JavaScript version	2.03	2.09
Platform	2.19	2.13
Charset	1.60	1.63
Language	2.23	2.24
Cookie support	0.87	0.80
Java support	0.98	0.95
Time zone	1.57	1.82
Plugin versions	5.54	6.25
Screen resolution	4.30	4.58
Font list	5.55	6.23
JavaScript Fields combined	**6.80**	**7.43**
JavaScript Fields combined (with screen resolution)	**6.89**	**7.47**

C. *Plugin based Fingerprinting*

HTML5 when combined with the advanced capabilities of JavaScript and Cascading Style Sheets (CSS), provide the ability to develop feature rich applications. But this was not the case with older versions of HTML because their ability to provide interactive applications such as games, videos and music were limited. So third party plugins such as Adobe Flash and Java were used to create, deliver and render interactive multimedia content [14]. Browser Plugin is a piece of software component which provide additional features to the browser that are actually missing in the browser [16].

Plugins offer Application Programming Interface (APIs) for accessing the underlying system's hardware and software properties and thus enable the developers to gain information about the running system. Flash is widely used for displaying videos and games in .swf format on the browser. It is also capable of getting information about the underlying system (e.g. operating system version) and the running application (e.g. flash plugin version) using the Flash's Capabilities Class [22]. In addition to system knowledge, 'EnumerateFont' method of Font class is used by developers for font enumeration. APIs are in favor of Fingerprinters who want to collect as much information as possible about the targeted user's system [16].

Java is another plugin that can be used by web browsers to display interactive web content such as online games and online chat programs. While it can be used for collecting the system information, it is not a desirable method for Fingerprinters [16]. User's explicit permission is required for Java Applet's execution in some situations as shown in Fig. 3 whereas Flash APIs can operate without targeted user's explicit consent or awareness [19].

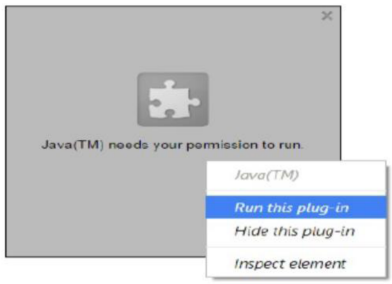

Figure 3: Java Plugin security-prompt for Java Applet

The entropy given by fonts and plugins is 13.4 bits and 15.4 which are highest of all other fingerprinting methods.

D. *Extension based Fingerprinting*

Browser extensions are mainly added to provide an additional functionality to browser. Some of the very common examples of browser extensions are Adblocker, Ghostery and Password Manager. Browser Plugins are software components which enable the browser to show content that is otherwise not supported by the browser whereas Browser extensions are the programs written in JavaScript to add new functionality to the browser. These extensions can also be used by Fingerprinters to collect information about user's browser. There is trade-off between privacy enhancing tools and fingerprinting as more the user install extensions to protect privacy, the more he will become unique for fingerprinting. For example, NoScript which is a popular browser extension for blocking the JavaScript and enhancing the privacy and security of the user can be exploited for fingerprinting purpose as only 1 out of 93 people disable or block JavaScript (as shown by GOV.UK) [16]. According to 'take−a−bite.org', dominant features that can be used in Fingerprinting are as shown in Fig. 4 [17].

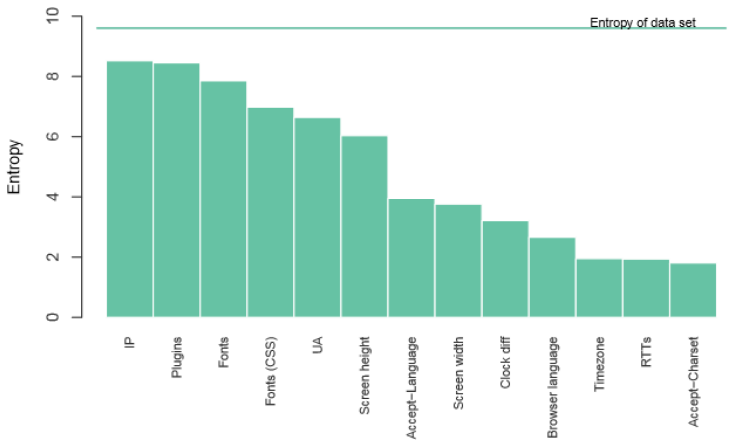

Figure 4: Top 13 Browser attributes with the highest entropy

IP address and Plugins installed plays a significant role in making a fingerprint more unique as they both have highest entropy followed by Fonts, User-Agent and Screen height.

VI. ANTI-FINGERPRINTING SOLUTIONS

According to Acar [5], the users whose data are collected through Fingerprinting are subjected to two types of concerns. First, they have no or very little knowledge about handling of their data as Fingerprinting leaves no traces behind that the user's data is being collected by the websites. Its stateless nature makes it very hard to detect even for the technical savvy users. The study conducted by Acar, showed that none of the companies that use fingerprinting offer an option to its users to get access of data stored about them. Second imbalance eased by Fingerprinting is lack of control over the process of fingerprinting [5]. A user can disable cookies but there is no facility provided by browser to disable or block fingerprinting. Some users switch to Private browsing mode but the private browsing mode is not developed in a way that it can protect the user against fingerprinting. Disabling cookies, Java, JavaScript and Flash may prevent the collection of data but provides a very rare configuration that could be used to enhance user identification [5]. When browser fingerprints are matched with individual's real identity (through social media plug-ins in the website) and are collected without explicit consent, lead to violation of privacy [16]. Even the "Do Not Track header" cannot affect fingerprinting in any way as the study by Acar has shown that not even single website stopped fingerprinting when they visited them with DNT header on [5].

Here, we will highlight some of the solution to lessen the intrusion caused by Browser Fingerprinting and we will also highlight some of their shortcomings.

A. Blocking Extensions

JavaScript plays the evilest role in browser fingerprinting and there are various browser extensions available (such as No-Script, Ghostery and Privacy Badger) that can be used to block the JavaScript based fingerprinting scripts. No-Script only runs those scripts that have been whitelisted by the user. Ghostery and Privacy Badger avoid downloading the tracking script from known trackers. However, these extensions do not offer any guarantee to protect against fingerprinting as it is not easy to maintain up-to-date whitelists. Moreover, these privacy enhancing extensions can be detected by the fingerprinting websites and thus make the browser more unique as less number of people install such extensions [18].

B. Spoofing extensions

In their study on browser fingerprinting, Nikiforakis et al. found 12 extensions that are specially designed to alter the user agent string and give Fingerprinters false information about the browser. They also shown that major problem with these extensions is that do not completely hide the true identity of user and since these extensions are installed by few people, they also lead to increased distinguishability of user [9].

For purpose of illustration, we installed a browser spoofing extension called ModHeader on Google Chrome and altered all the prominent attributes of HTTP Headers (charset, encoding, DateTime and User-Agent); with altered HTTP Header values when we visited LetMeTrackYou.org we found inconsistency in the result produced (Refer Table 7).

Table 7: Inconsistency in the result retrieved through HTTP Headers and JavaScript

The values retrieved by LetMeTrackYou.org through HTTP header were: **User-Agent:** Mozilla/5.0 (X11; Linux x86_64; rv: 12.0) Gecko/20100101 Firefox/21.0, **Accept-Charset:** utf **Accept-Language**: en (These are same values that were modified using extension)
The values retrieved by LetMeTrackYou.org through JavaScript were: **navigator.appVersion**: 5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.94 Safari/537.36 **navigator.languages:** en-US, en, hi **document.charset:** UTF-8

As shown in Table 7, all the values modified by ModHeader extension were correctly fetched by website using JavaScript as there are multiple ways to fetch value of a single browser property, such as the value of User-Agent can not only be retrieved by User-Agent attribute of Http Header but also with navigator.appVersion attribute of JavaScript. Fingerprinting websites through the mismatching of User-Agent can clearly identify that the user is trying to reduce the trackability of his online activities.

Thus the spoofing browser extensions inadequately hide the true identity of browser which can still be exposed using JavaScript because of the following reasons [18]:

- Impossible configuration: None of the browser spoofing extensions can alter the value of screen object of the JavaScript language. So, the user using normal laptop or workstation and pretending to be using a mobile phone is showing an impossible width and height configurations, for instance resolution of 1,920 and 1,080 for a mobile phone is impossible.
- Mismatch of user agents: The extensions can only change browser's HTTP headers and leave the corresponding JavaScript attributes intact thus giving inconsistencies between the extension's and JavaScript's reported values. This inconsistency of the values can be used to uncover the presence of specific extensions which can further make the web browser more unique.

Using blocking extension with spoofing extensions may prevent the collection of data but will again make the fingerprinting easier.

C. Use of multiple browsers

Another solution to avoid fingerprinting is to use multiple browsers, each one revels a different fingerprint. But Boda et al. in their research on Cross browser fingerprinting has shown that just changing the browser is insufficient as Fingerprinters also relies on the Operating System specific data and system fonts to generate a unique fingerprint and fingerprints generated in this way will remain same from browser to browser on the same computer [18]. As shown in Fig. 5, software components such as fonts, plugins, operating system and browser also contribute in generation of unique fingerprint. By disabling the installed plug-ins can limit the cross browser fingerprinting but again will limit the functionality of the websites too as some websites rely on plug-ins for full functionality.

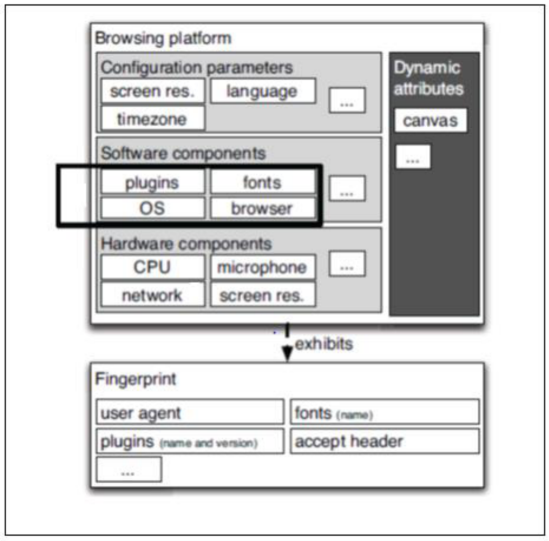

Figure 5: User platform elements also contribute in generation of Fingerprints

D. Tor Browser

The Tor browser which is a modified Firefox browser and designed specifically for Tor network is the only software that is capable of spoofing the browser attributes to prevent fingerprinting. The main goal of using Tor is to achieve Cross-Origin-Fingerprinting-Unlinkability which means to prevent the linking of user's activities across multiple websites by collecting and comparing his fingerprints [5]. Tor browser create single configuration for all the Tor users so that they can never be recognized from their configuration [18]. Tor users generate a unique fingerprint and to achieve this Tor browser spoofs various properties such as system language, browser versions, time zone and operating system. It also fixed values for User-Agent, Accept-Language and Accept-Charset HTTP headers [5].

Battery API, Network API and plugins such as Flash plugin are also disabled by default in the Tor browser. Tor browser returns a blank image when it is asked to extract image from the canvas thus save the user from canvas fingerprinting as well. The numbers of fonts that a web page can use are also very limited in this browser. It also disables all the attempts made by website to list the underlying system fonts for fingerprinting purpose. It also provides No-Script by default thus protects the user from JavaScript fingerprinting too [5].

All those browser properties that have to be disclosed for the communication to happen and can be used for fingerprinting are replaced by uniform values for all Tor users. The functionalities that may help an attacker to uncover the actual devise are simply disabled. Additionally, the properties which can be retrieved by multiple sources (for example, screen size can be retrieved by both by JavaScript and CSS), the Tor reports the same value for them [5].

But using Tor functionality for avoiding fingerprinting is also very brittle as minor changes in screen size or enabling JavaScript can disclose the identity of browser and users. Tor browser's mono-configuration is its valuable weapon that can be used effectively against fingerprinting.

When we visited Panopticlick website through Tor browser, we found that it provides strong protection against web tracking and one in every 58th browsers share the same fingerprints as produced by Tor Browser which shows the fingerprints generated by Tor Browser are very common. Also, the fingerprints of Tor browser convey only 5.85 bits of identifying information. This entropy can further be reduced if the number of people using Tor browser increases as they all will generate same fingerprint.

VII. Conclusion

The complete invisibility in tracking and by rendering the existing countermeasures against tracking useless makes the Device Fingerprinting a major threat to user privacy. We lack basic deployable solutions against fingerprinting that may provide effective control and transparency over the fingerprinting process. If deceiving the websites through spoofing method is only solution to avoid device Fingerprinting, then a carefully designed spoofing scheme which is sensitive to dependency between properties needs to be considered. The obfuscation scheme must not only convince the attacker with carefully adjusted parameters but also needs to play well with highly complex web ecosystem so that it would not "break the web"; a huge problem which could discourage browser vendors from thinking about the solution. Unlike Tor, the obfuscation scheme proposed must be universal as the more and more users will have same fingerprints, the more devise fingerprinting process become hard for the Fingerprinters.

References

[1] Lapwedrix, P., Rudametkin, W. & Baudry, B. Beauty and the Beast: Diverting modern web browsers to build unique browser fingerprints. 37th IEEE Symposium on Security and Privacy (S&P 2016), 2016.

[2] Nikiforakis, N., Kapravelos, A., Joosen, W., Kruegel, C., Piessens, F. & Vigna, G. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. Security and privacy (SP), 2013 IEEE symposium on, 2013. IEEE, 541-555.

[3] Mediapost. 2013. Study: 44%Of Adults Opt Out Of Targeted Ads, 66% Delete Cookies [Online]. Available: http://www.mediapost.com/publications/article/191809/study-44of-adults-opt-out-of-targeted-ads-66-d.html [Accessed 08 May 2016].

[4] Rausch, M., Bakke, A., Patt, S., Wegner, B. & Scott, D. Demonstrating a Simple Device Fingerprinting System. Proceedings, Midewest Instruction and Computing Symposium, 2014.

[5] Acar, G., Eubank, C., Englehardt, S., Juarez, M., Narayanan, A. & Diaz, C. The web never forgets: Persistent tracking mechanisms in the wild. Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, 2014. ACM, 674-689.

[6] Laksono, T. D., Rosmansyah, Y., Dabarsyah, B. & Choi, J. U. JavaScript-based device fingerprinting mitigation using personal HTTP proxy. 2015 International Conference on Information Technology Systems and Innovation (ICITSI), 2015. IEEE, 1-6.

[7] Boda, K., Foldes, Á. M., Gulyas, G. G. & Imre, S. 2011. User tracking on the web via cross-browser fingerprinting. Information Security Technology for Applications. Springer.

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

[9] Eckersley, P. How unique is your web browser? Privacy Enhancing Technologies, 2010. Springer, 1-18.

[10] VAITES, M. 2013. The effectiveness of a browser fingerprint as a tool for tracking.

[11] Mulazzani, M., Reschl, P., Huber, M., Leithner, M., Sshrittwieser, S., Weippl, E. & Wien, F. Fast and reliable browser identification with javascript engine fingerprinting. Web 2.0 Workshop on Security and Privacy (W2SP), 2013.

[12] Kolbitsch, C., Livshits, B., Zorn, B. & Seifert, C. Rozzle: De-cloaking internet malware. Security and Privacy (SP), 2012 IEEE Symposium on, 2012. IEEE, 443-457.

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

[14] Acar, G., Juarez, M., Nikiforakis, N., Diaz, C., Gureses, S., Piessens, F. & Preneel, B. Fpdetective: dusting the web for fingerprinters. Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, 2013. ACM, 1129-1140.

[15] Gou.uk. 2013. How many people are missing out on JavaScript enhancement? [Online]. Available: https://gds.blog.gov.uk/2013/10/21/how-many-people-are-missing-out-on-javascript-enhancement/ [Accessed 09 May 2016].

[16] Faiz Khademi, A. 2014. Browser Fingerprinting: Analysis, Detection, and Prevention at Runtime.

[17] Flood, E. and Karlsson, J., 2012. Browser fingerprinting.

[18] Laperdrix, P., Rudametkin, W. & Baudry, B. Mitigating browser fingerprint tracking: multi-level reconfiguration and diversification. Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, 2015. IEEE Press, 98-108.

[19] Nikiforakis, N., Kapravelos, A., Joosen, W., Kruegel, C., Piessens, F. & Vigna, G. 2014. On the workings and current practices of web-based device fingerprinting. IEEE Security & Privacy, 12, 28-36.

[20] Fiore, U., Castiglione, A., De Santis, A. & Palmieri, F. Countering Browser Fingerprinting Techniques: Constructing a Fake Profile with Google Chrome. 2014 17th International Conference on Network-Based Information Systems, 2014. IEEE, 355-360.

[21] Wikepdia. 2016. HTTP [Online]. Available: https://hu.wikipedia.org/wiki/HTTP [Accessed 25 May 2016].

[22] Vaites, M. 2013. The effectiveness of a browser fingerprint as a tool for tracking.

[23] Khademi, A. F., Zulkernine, M. & Weldmariam, K. 2015. An Empirical Evaluation of Web-Based Fingerprinting. IEEE Software, 32, 46-52.