Inhibiting Fingerprinting Technique with FP-prevention

Sakchan Luangmaneerote, Ed Zaluska, Leslie Carr
University of Southampton, UK

Abstract

A number of fingerprinting countermeasures has been proposed to the public. Today, none of the concrete results indicate which countermeasures might be applicable. The research attempts to prove their claims, requires studying the effectiveness of their fingerprint prevention and observe the results of side effects after defence. The result will reflect the limitation of current countermeasure and will assist the attentive users to determine which countermeasure should be suited for them. Under investigation, all countermeasures will be installed on the web browser and visit the developed hybrid fingerprint website in order to gauge the efficiency of the fingerprint resistance of all types. The research indicates that all fingerprint countermeasures nowadays are unable to obstruct all kinds of the fingerprint tracking and countermeasures that use the blocking technique appear more side effects to the web browser than other techniques. Finally, information paradox should be considered concurrent with the part of fingerprinting prevention. Thus, the research proposes a new model to tackle the fingerprinting tracking with the minimum impact to the web browser.

1. Introduction

This paper conducts the empirical study of the efficiency of fingerprinting prevention of existing countermeasures and their side-effects. The initial investigation finds available countermeasures which generally are declared the potential of fingerprinting inhibition. All countermeasures will be installed on the web browser and then visits the fingerprinting website in evaluating the efficiency of the fingerprinting prevention. The second investigation will be conducted the in-depth study of altering attributes after the fingerprint prevention. The result will assist to realise how many fingerprinting attributes to be handled and what is the mechanism behind the operation of tracking inhibition. The third investigation is to evaluate side-effects of each countermeasure. Each metric will be set in order to measure the user satisfaction. The result of this section will demonstrate another part of the fingerprinting prevention which can result in the user browsing experience. The fourth investigation will conduct the information paradox of the web browser. Even though some countermeasure might prevent the fingerprinting, it might make the web browser stand out that can result in easier tracking. Finally, the FP-prevention will be proposed in order to address the weakness of existing fingerprinting technique.

The key contributions of this paper are the following:

1) *Evaluating the fingerprintning prevention of each countermeasure*
2) *Fingding which approachs can prevent fingerprinting effective with the minimum impact to the web browser.*
3) *Proposing a novel countermeasure to tackle the weaknesses of the current countermeasures.*

2. Related Work

Identifying the web browser similar to the fingerprinting the human finger is a real possibility [1]. The overwhelming technologies on the web browser has caused to unable to control the access of the user data. This weakness opens a chance of a web server to learn more information of user data through use of the web browser. The number of attributes learned will be concatenated into a string and then calculated with the hash functions. The output of Hash function that is hash ID will be performed as an individual tracker. Unlike the web cookies, this tracker is unnecessary to store any files on the user computer. Thus, user cannot be aware that they are being tracked by someone. The fingerprinting technique is claimed by Eckersley [2], a man who built the Panopticlick project, can identify the user computer with accuracy of 96%, even frequently updating attributes. He established that list of fonts accessed by the plugin is distinct differences from other attributes. He calculated the entropy of fonts 13.9 bits of entropy, which mean that if he only uses one attribute to create the fingerprint tracker, it has only one in 15,286 similar browsers will be found on the web Internet. It can be clearly seen that only one list of fonts is distinctive enough to create a unique tracker. The Panopticlick project has initially inspired several subsequent papers. Later on, Boda [4] demonstrated that a list of fonts could be accessed by JavaScript. They combined a list of fonts, the IP address, time zone, and screen resolution which the result is a high level of success similar to the Panopticlick project. His discovery had increased a way of fingerprinting a user computer with difficulty to avoid tracking. Next, the canvas element in HTML5 usually uses to draw a graphic on the web browser it turns out to be one attribute to be used fingerprinting [5] known as the “canvas fingerprinting”. Canvas uses its write and read method to fingerprint the user computer. The retrieved images are sufficiently enough as a unique identity. In the same year, the browsing history on the web browser was proposed that it can identity the web browser

with the accuracy of 69% [6]. However, this technique is not currently applicable to latest version of the modern web browser as they initiate to block capability of accessing browsing history. So far, many fingerprinting methods have been launched such as packet ordering information [7], combining IP address and UserAgent [8] and mobile fingerprinting [9] audio fingerprinting [10] which mainly derive from the fundamental concept of the Panopticlick project.

3. Testing Countermeasures

mA number of approaches are proposed to solve problems of fingerprinting tracking seemingly more intense. The research will download countermeasures and install this approach on the web browser except for Tor Browser [14] and then force the web browser to visit the fingerprinting websites many times in observing the tracking of fingerprinting technique. For "Do not Track" (DNT), it will not be evaluated as it is proven that many fingerprinting companies are not compliant with the agreement of DNT. The number of countermeasures is shown in Table 1 below.

Table 1. List of countermeasures

Countermeasure	Platform
Tor [13]	Running on Tor browser
Rubberglove [15]	Running on Chrome
Chameleon [16]	Running on Chrome
CanvasFingerprintBlock [18]	Running on Chrome
ChromeDust [21]	Running on Chrome
StopFingerprinting [20]	Running on Chrome
UserAgent Switcher for Chrome [21]	Running on Chrome
Canvas Fingerprinting Blocker [22]	Running on Firefox
FireGloves [23]	Running on FireFox
FP-Block [17]	Running on FireFox
Stop fingerprinting [24]	Running on FireFox

The research will give more attention to countermeasures which are widely available on the Internet. Countermeasures are unavailable for download, such as FPGuard [11] and Privaricator [12]. They will not be evaluated for this research.

4. Evaluating the Efficiency of Fingerprinting Prevention

In this section, the downloaded fingerprint countermeasures had been evaluated whether they could inhibit the fingerprinting tracking. In order to estimate the potential of each countermeasure, the hybrid fingerprinting site had been developed specifically in order to observe the operation of prevention and alteration of user attributes [3]. This site will calculate the fingerprinting ID obtaining access of user data. The Output of site will consist of five fingerprinting ID, namely: object fingerprinting ID, canvas fingerprinting ID, plugin fingerprinting ID, and font fingerprinting ID and hybrid fingerprinting ID, created by integrating four types of fingerprinting technique. The countermeasure installed will be forced to visit the site and observe the alteration of fingerprinting ID. If fingerprinting ID change every time, it indicates that the countermeasure test is capable of preventing the fingerprinting tracking. The research has two steps to evaluate the countermeasure which does not change the fingerprinting ID. Firstly, it will embed fingerprinting code different websites and then force the web browser to visit that websites. If the fingerprinting ID is altered, it means that the countermeasure can prevent the fingerprinting tracking. However, if the fingerprinting ID is not changed, the research will use the second step to test the fingerprinting technique. The second step will install the countermeasure to 30 browsers in different user computers and then force the web browser to visit the site. If the fingerprinting site cannot distinguish 30 user computers, it means that the countermeasure can prevent the fingerprinting tracking. In contrast, if the site can distinguish which computers belong to any user, it means that it cannot prevent the fingerprinting tracking.

4.1. Result of testing prevention

Table 2 shows Tor and FireGlove perform as the best prevention which three types of fingerprinting can be inhibited. The second prevention is RubberGlove, Stop fingerprinting and FP-block that can prevent two types of fingerprinting. CanvasFingerprintBlock and canvas fingerprinting can only prevent one type of fingerprinting as expected.

Table 2. Efficiency of fingering prevention

Countermeasure	Object JavaScript (navigator, screen)	List of fonts	List of plugins	Canvas
Tor	√	√	√	-
RubberGlove	√	-	√	-
Chameleon	≈	-	-	≈
CanvasFingerprintBlock	-	-	-	≈
ChromeDust	-	-	-	-
StopFingerprinting	-	-	-	-
Canvas Fingerprinting blocker	-	-	-	√
FireGloves	√	√	√	-
FP-Block	√	-	√	-
Stop Fingerprinting	-	√	√	-
UserAgent Switcher to Chrome	-	-	-	-

(√) The countermeasure can prevent fingerprint by blocking, spoofing or randomization
(-) The countermeasure allows fingerprinting
≈ The countermeasure can detect fingerprint but cannot

prevent fingerprint tracking.

The remaining of countermeasures consisting of Chameleon, ChromeDust and StopFingerprinting is not appearing to protect the fingerprinting tracking as they claim. Thus, these countermeasures will be cut out from the next experiments due to ineffective use. Notably, almost countermeasures are incapable of preventing the fingerprinting tracking as some countermeasures are useless to inhibit the fingerprinting tracking.

4.2. Study of fingerprinting attributes

This section will conduct the in-depth study of fingerprinting attributes in order to observe how many fingerprinting attributes are modified over the fingerprinting prevention. Each countermeasure will visit the fingerprinting site which all attributes will be studied in order to realize the mechanism behind each countermeasure.

As for the result from Table 3, The data compares the five countermeasures with 26 fingerprinting attributes used. As is seen from the given illustration, each countermeasure handled the number of fingerprinting attributes and techniques differently. To regards the number of attributes modified, some countermeasures deal with almost attributes while some deal only with two attributes. The number of selected attributes in each countermeasure to resist the fingerprinting tracking are so interesting why they select differently. Notably, the approach against fingerprinting tracking consist of three techniques, namely blocking, randomising, and spoofing technique. Blocking technique is to disallow the fingerprinting algorithm to access the user data. The randomization technique is a technique to alter browser's attributes every time to visit the website. The spoofing technique is a technique to alter attributes when the user changes the URL of a website. However, the number of attributes to be handled and technique to prevent the fingerprinting tracking are so noticeable. The result of this experiment will be kept considering with the next experiment in order to study the side-effects simultaneously.

4.3. Side effects of prevention

The remaining countermeasures that can prevent the fingerprinting tracking in Table 2 will be taken to conduct the additional study of adverse impacts to the web browser. The four matrixes are determined in measuring the side-effects. Some sample of popular websites will be evaluated by installing each countermeasure on the web browser and then visit websites in order to test side-effects.

Considering the results of side-effects in Table 4, the number one of problem is Tor, particularly the login problem and difficulty to use such as the low speed of Internet. The most second problem are RubberGlove and Fireglove except for the login problem. CanvasFingerprintBlock and Canvas Fingerprinting appeared to have insignificant adverse effects.

Table 3. The number of attributes to be handled

	Fingerprinters				
Properties	**Tor**	**Stop fingerprinting**	**Fireglove**	**Rubberglove**	**FP-Block**
List of plugins	Blocking	Blocking	Blocking	Blocking	Blocking
List of fonts	Spoofing	randomizing	Blocking	-	-
User-Agent	Spoofing		Spoofing	Blocking	Spoofing
HTTP header Accept-Language	Spoofing	-	Spoofing	-	Spoofing
appCodeName	Spoofing	-	-	Blocking	Spoofing
Product	Spoofing	-	-	Blocking	Spoofing
Product-Sub	Spoofing	-	-	Blocking	-
Vender	Spoofing	-	-	Blocking	Spoofing
Vendersub	Spoofing	-	-	Blocking	-
Online	-	-	-	Blocking	Spoofing
appVersion	Spoofing	-	-	Blocking	Spoofing
cookiesEnabled()	-	-	-	Blocking	Spoofing
javaEnable()	-	-	-	Blocking	spoofing
Navigator.mimeType ()	blocking	-	Blocking	Blocking	Blocking
Screen color and pixel depth	spoofing	-	-	-	Spoofing
Screen width and height	Spoofing	-	-	-	Spoofing
Screen availLeft, availTop, availHeight and availWidth	Spoofing	-	-	-	-
Screen hosizontalDPI, verticalDPI	Spoofing	-	-	-	-
Canvas fingerprinting	-	-	-	-	-
Do not track	-	-	-	Blocking	Spoofing
Timezone	Spoofing	-	-	-	Spoofing
OS & Kernel Version	Spoofing	-	-	Blocking	Spoofing
JS: Flash Enabled	Blocking	-	Blocking	Blocking	Blocking
CPU	Spoofing	-	-	Blocking	Spoofing
Language	Spoofing	-	Spoofing	Blocking	Spoofing
Languages	Spoofing	-	Spoofing	Blocking	Spoofing

Table 4. Negative effects to the web browser

Countermeasure	Problem of display	Problem of functionality	Difficult to use	Login problem
Tor	√	√	√	√
RubberGlove	√	√	√	-
CanvasFingerprintBlock	-	-	-	-
Canvas Fingerprinting	-	-	-	-
FireGloves	√	√	√	-
FP-Block	-	√	-	-
Stop Fingerprinting	-	√	-	-

Problems of display: Content, fonts or screen size are changed.
Problem of functionality: The video and music do not play, or some functionalities are unavailable.
Difficult to use: Rendering a webpage is a slow process, makes user annoying, and is unsmooth.
Login Problem: The user faces the login problems.

FP-block and Stop Fingerprinting seemed the slightest problem.

To consider results of Table 2, Table 3 and Table 4, all given results reflects different aspects of fingerprinting countermeasures. The countermeasure to inhibit the great fingerprinting tracking is unnecessary to be a great countermeasure. Tor and Fireglove are excellent in part of the fingerprinting prevention, but they turn out to produce more side-effects to the browsing experience than other. The second worst countermeasure is Rubberglove using the blocking technique. This technique causes direct impact on the browsing experiences which should avoid this technique. The result is clear that a decreasing number of fingerprinting attributes prompts an insignificant impact comparable to increasing number of fingerprinting attributes. In addition, the result in Table 4 is relatively distinct that blocking technique is more substantial effects than other technique.

4.4. Studying information paradox

Both Eckersley [2] and Nikiforakis [13] found that some existing fingerprint countermeasures created an unintentional problem rather than inhibiting the fingerprinting prevention. From their study indicate that some countermeasures make the web browser stand out which this is easier to be fingerprinted. These countermeasures having a problem show an abnormal combination of their fingerprinting over sending back their information to the server which is lead to problems pf the incomplete coverage of JavaScript object, inconsistencies between related attributes and the mismatch between values and HTTP header. Therefore, this section will investigate whether the five countermeasures are having problems or not.

Table 5. Result of the information Paradox

Countermeasure	Paradox
Tor	-
UserAgent Spoofing	Inconsistency between operating system and navigator.plugins
Fireglove	Inconsistency between CPU and userAgent
FP-Block	Inconsistency of DNT, CookieEnabled, JavaEnabled()
CanvasFingerprint Block	-
Stop fingerprinting	-
RubberGlove	Inconsistency between navigator object and HTTP header request

From the result in Table 5, it can be seen that most countermeasures handle fingerprinting attributes without attention to conflicts of browser's attributes. This result is not only shown the inconsistency of browser's attributes, but also discovers some attributes (Do Not Track, and cookieEnabled) that display in the HTTP header request should not be handled because these attributes might be set by the user at any time which randomizing these attributes will be contradicted with the user preference. Also, JavaEnabled attribute usually associates with the plugin if this attribute is changed without attention to effect. It might produce a problem of inconsistency of information. For example, let say that JavaEnabled() was set to the false value, while the JavaEnabled() appear in plugin attribute which is a contradiction.

5. Proposing FP-prevention

FP-prevention is designed to address the weaknesses of existing countermeasures. FP-prevention circumvents the blocking technique by selecting the randomization technique because of insignificant impact to the web browser according to Table 3, 4 and 5. The main reason why FP-prevention does not select the spoofing technique as it does not keep a list of URL which might cause to easily fingerprinting rather than preventing tracking. It intends to inhibit fonts, canvas, plugin, and object fingerprinting (more than current existing countermeasures). It selects both high and low entropy attributes (plugin, mimeTypes, fonts, and UserAgent) before randomization. The FP-prevention is expected to deter the fingerprinting tracking more efficient than current existing countermeasures and reduce the side-effects to the Internet browser more than existing countermeasures.

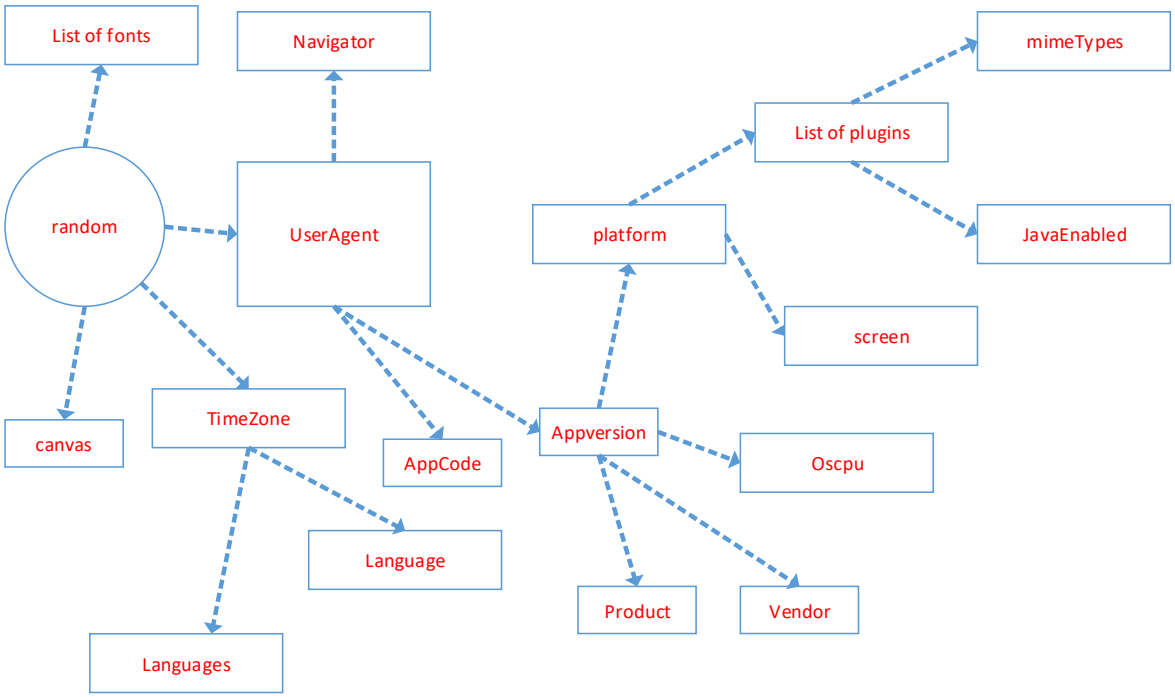

Figure 1. FP-prevention model

The FP-prevention designed to operate on the Chrome extension. The concept of FP-prevention alters the selected attributes automatically in order to change the identity of the user computer away from tracking of the fingerprinting algorithm. The selected attributes in FP-prevention model consist as Table 6 below:

Table 6. List of selected attributes

Attributes	**List of selected attributes**
navigator	UserAgent, appCodeName, appName, appVersion, product, oscpu, vendor, platform, language, languages, mimeTypes, JavaEnabled, geolocation.
screen	Hight, width, ColorDepth, pixelDepth, availLeftt, availHeight, availWidth, offsetwidth, offsetHeight, getBoundingClientRect
Plugins	name, filename, description, length
MimeTypes	enablePlugin, description, suffixes, type
Timezone	Randomizing Timezone
List of fonts	randomizing virtual fonts
List of plugins	randomizing virtual plugins
Canvas	randomizing minor noise into canvas element

The FP-prevention will handle some part of detecting the fingerprinting code as little as possible. This reason that comes from complex of the fingerprinting code is extremely difficult to be detected. In order to reduce mistake of detection, FP-prevention will pay attention to change the identity of the user computer without creating any side-effects to the web browser rather than interesting on the side of fingerprinting detection. In addition, the problems of information paradox should be addressed parallel with the part of fingerprinting prevention. The operating procedure of FP-prevention will originally randomise userAgent and then bring randomised userAgent to distribute to other attributes. The result of randomization will alter the identity of the user computer without activating side-effects to the web browser. This model aims to reduce the information paradox on the user computer which can cause easier fingerprinting.

Given Figure 1, the workflow of FP-prevention would initiate to randomizie the UserAgent string. The reason why chooses the UserAgent because the UserAgent string has various attributes that are stored its inside, more details in Table 7 below. The model regards that it is unnecessary to randomize each attribute as a sequence due to waste of time. The form of userAgert is displayed below.

Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.94 Safari/537.36

Table 7. List of selected attributes

Chrome 50.0.2661.94	
Mozilla/5.0	Mozilla version
Windows NT 6.1	Operating system (Windows 7)
WOW64	A 32-bit application is executing on a 64-bit processor
ApplewebKit	a set of core classes to show web content in Windows
537.39	Web Kit build
KHTML	Open source HTML layout engine developed by KDE project
Gecko	Web browser engine developed by Mozilla Foundation
Chrome	Chrome
50.0.2661.94	Chrome version
Safari	Based on Safari
537.36	Safari build

The web browser using the FP-prevention will obtain the value of randomised UserAgent string first, and then the mechanism of extracting other attributes will be established according to randomised UserAgent string. The FP-prevention model randomly selects userAgent string from Mozilla Firefox, Google Chrome, Internet Explorer and Opera. Randomising UserAget at once can obtain many attributes such as the web browser's information (name, version and engine of the web browser), and operating system (name of the operating system and a number of bits CPU). After obtaining the UserAgent, it will be split into navigator.userAgent, navigator.appCode, and navigator.appVersion respectively. The navigator.appCode will be divided into four attributes and then replace to the original attribute value. This way will reduce consuming time Of all attributes on the web browser while randomising attributes.

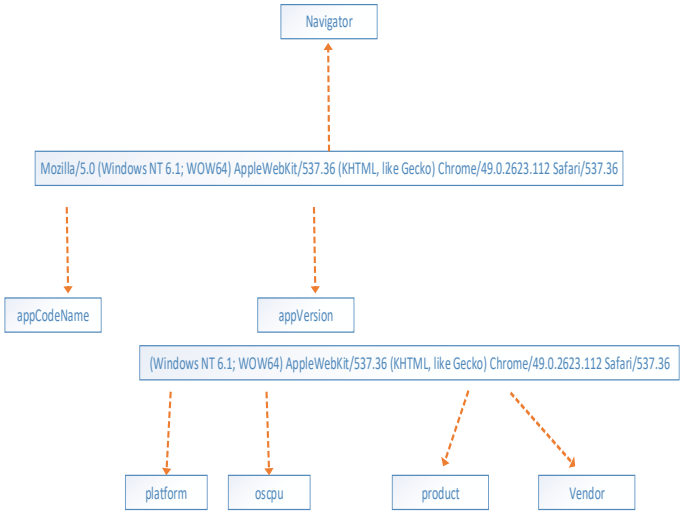

Figure 2. Workflow of splitting UserAgent string

Considering navigator.appVersion in Figure 2, it will be analysed which information are collected inside the UserAgent. For example, the platform of user computer will be analysed from navigator.appVersion and then bring the result of analysis to replace at the navigator.platform. All new attributes in connection with appVersion will be performed as the same procedure to replace a new value according to navigator.appVersion. Another example, assuming that:

AppVersion = (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.94 Safari/537.36

It means that this is a Chrome Browser using Gecko (browser's engine) and running on Windows 7 by using CPU 64 bits. Therefore, the FP-prevention will attempt to randomise obtained information according to appVersion (is navigator.product = Gecko and navigator.vendor = Google Inc).

In part of the platform, the FP-prevention selects Windows, Linux and Mac consistent with a randomness of the UserAgent string. Assuming that UserAgent randomly select Windows. Therefore, navigator.platform = Windows and then FP-prevention will attempt to randomise visual plugins related to use of the Windows. As each plugin possesses own mimeType, randomised plugins are associated with a random mimeType.

In part of the screen, FP-prevention randomly selects most popular desktop screen resolution between 2012 and September 2014 as follows, 1366x768, 1920x1080, 1280x1024, 1024x768, 1280x800, 1440x900, 1680x1050, 1280x720, 1360x768, and 1280x768.

Timezone associated with the web browser language (navigator.language and navigator.languages). Therefore, randomising Timezone must also consider the web browser language. For example, if randomising that the web browser uses in the UK, it must display of en-US because it is impossible to support another language.

In order to reduce the information paradox, the consistency between JavaScript object (navigator) and HTTP header field must be coordinated. Usually, the communication from the Web browser to the website is done through HTTP protocol. The web browser conveys the HTTP request to the website and then the website will acknowledge the web browser with HTTP response as detailed in Table 8 and Table 9 below.

Table 8. Detail of request header

Request Header	**Data**
Accept	text/html,application/xhtml+xml,application/xml;q=0.9 ,image/webp,*/*;q=0.8
Accept-Encoding	gzip, deflate, sdch
Accept-Language	en-US,en;q=0.8
Cache-Control	max-age=0
Connection	keep-alive
Cookie	ga=GA1.2.700983430.1460133764;_cbclose=1; _cbclose22799=1;_uid22799=9D5FEB32.11; verify=test
DNT	1
Host	dict.longdo.com
Referer	http://dict.longdo.com/search/unseeing
User-Agent	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.94 Safari/537.36

Table 9. Detail of response header

Request Header	**Data**
Connection	keep-alive
Content-Encoding	gzip
Content-Length	24688
Content-Type	text/html
Date	Fri, 13 May 2016 15:29:20 GMT
Server	nginx/1.8.0
Vary	Accept-Encoding
X-Powered-By	PHP/5.3.10-1ubuntu3.19

The FP-prevention model will embed a proxy in the browser extension. The communication between the web browser and the web server will be intercepted every time consistent with randomised UserAgent. Therefore, HTTP requests filed that are conveyed to the web server will not be the authentic HTTP field.

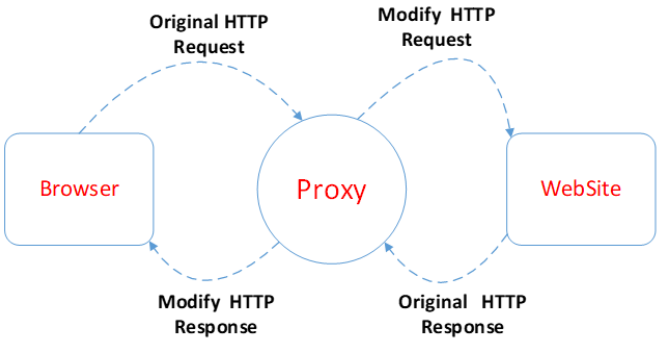

Figure 3. Workflow of embed proxy

To considering the Figure 3, the concept of operation will listen to HTTP-on-modify-request and notify browser extension while HTTP request is ready to send. The browser extension will intercept HTTP request and then replace the original HTTP request by spoofing UserAgent and Accept-Language which obtain from a defined list of UserAgent. This mechanism will assist problem of abnormal combination between HTTP Header and JavaScript object.

5.1. Features of FP-prevention

The purpose of FP-prevention aims to change the identity of the web browser. The result of changing browser's identity can inhibit fingerprinting tracking, and the web page is able to execute web pages smoothly without annoyance to the user.

FP-Prevention model differs from the current existing method as following. Firstly, this model uses only randomization technique (not blocking) by considering the relationship of attributes. Reducing the number of fingerprinting attributes unrelated is to mitigate the browser stand out which could cause easier fingerprinting. Secondly, this model considers sufficient fingerprinting attributes to prevent effectively and could reduction of side-effects to the web browser. Also, this model does not randomise all fingerprinting attributes. Some attributes are ignored (e.g., Do Not Track (DNT) or cookieEnabled) because these attributes are set to the user preference which randomising these attributes probably make browser stand out. Thirdly, this model considers the consistency between HTTP header and JavaScript object. Many existing randomization techniques (UserAgent spoofing, Fireglove, FP-block and RubberGlove) focus only on JavaScript objects which produce the biggest mistake because some JavaScript objects related to HTTP header are detected by the fingerprinting companies. Thus, considering the consistency between HTTP header and JavaScript object would reduce the problem of browser stand out as well. Fourthly, this model has improved the workflow of randomization technique. It is unnecessary to randomise all attributes. This model starts to randomise from UserAgent to other attributes because it contains many attributes inside which UserAgent randomised at once can link automatically

to other attributes. Fifthly, this model requires inhibiting hybrid fingerprinting because fingerprinters currently attempt to fingerprint browser with multiple ways. The information is shown below in Table 10.

Table 10. Feature of FP-prevention

	Prevention Features					
Countermeasure	**Object fingerprinting**	**Font fingerprinting**	**Plugin fingerprinting**	**Canvas fingerprinting**	**Reducing browser stand out**	**Consistency between JavaScript and HTTP header**
Fireglove	√	√	√			
FP-Block	√		√	√	√	√
Stop fingerprinting		√	√			
RubberGlove	√	√	√	√		
FP-prevention	√	√	√	√	√	√

6. Conclusion

This paper desires to realize the capability and limitation of existing countermeasures. The given results indicate that the existing countermeasures still have a shortcoming. The number of selected attributes and method to inhibit the fingerprinting tracking contribute to the browsing experiences. In addition, some existing countermeasures remain creation of information paradox which leads to easier fingerprinting. The countermeasure that prevent well does not mean the great countermeasure. The great fingerprinting countermeasure should regard the part of fingerprinting prevention, side-effects and information paradox simultaneously. The paper proposes a new model, FP-prevention, to address the weakness of existing countermeasures.

7. References

[1] J. R. Mayer, “Any person... a pamphleteer”: Internet Anonymity in the Age of Web 2.0,” Undergraduate Senior Thesis, Princeton University, 2009.

[2] 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.

[3] S. Luangmaneerote. "hybrid fingerprinting website," http://www.pleasefingerprintme.org

[4] 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.

[5] K. Mowery, and H. Shacham, “Pixel perfect: Fingerprinting canvas in HTML5,” Proceedings of W2SP 2012 IEEE Computerc Society, pp. pp. 1-12, May 2012, 2012.

[6] L. Olejnik, C. Castelluccia, and A. Janc, "Why johnny can't browse in peace: On the uniqueness of web browsing history patterns," in *5th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2012)*, 2012, p. 16 pages.

[7] L. Lu, E.-C. Chang, and M. C. Chan, "Website fingerprinting and identification using ordered feature sequences," in In Proceedings of the European Symposium on Research in Computer Security, 2010, pp. 199-214.

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

[9] T. Hupperich, D. Maiorca, M. Kührer, T. Holz, and G. Giacinto, "On the Robustness of Mobile Device Fingerprinting: Can Mobile Users Escape Modern Web-Tracking Mechanisms?," in Proceedings of the 31st Annual Computer Security Applications Conference, 2015, pp.. 191-200.

[10] P. W. T. A. Project. "AudioContext Fingerprint Test Page," 12/06/2016, 2019; https://audiofingerprint.openwpm.com/.

[11] K. Weldemariam, "FPGuard: Detection and Prevention of Browser Fingerprinting," in *Data and Applications Security and Privacy XXIX: 29th Annual IFIP WG 11.3 Working Conference, DBSec 2015, Fairfax, VA, USA, July 13-15, 2015, Proceedings*, 2015, p. 293.

[12] 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.

[13] 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. pp. 541-555.

[14] Tor. "Tor Project: Anonymity Online," 27/04/2015; https://www.torproject.org/.

[15] J. S. Clary, "RubberGlove" 19/06/2016; https://chrome.google.com/webstore/detail/rubberglove/koabfojebhfdjnligkcihoeekimoekpg?hl=en.

[16] Mozilla_Public_License. "Chameleon," 20/06/2016; https://github.com/ghostwords/chameleon.

[17] C. F. Torres, H. Jonker, and S. Mauw, "FP-Block: Usable Web Privacy by Controlling Browser Fingerprinting." pp. 3-19.

[18] Addidrime.net. "CanvasFingerprintBlock," 15/05/2015; https://chrome.google.com/webstore/detail/canvasfingerprintblock/ipmjngkmngdcdpmgmiebdmfbkcecdndc.

[19] R. T. a. T. G. Ram Bhaskar. "Combatting Browser Fingerprinting with ChromeDust," 20/06/2016; https://css.csail.mit.edu/6.858/2013/projects/rambhask-tgalvin-rrt.pdf.

[20] stopfingerprinting. "StopFingerprinting," 20/06/2016, 2016; https://chrome.google.com/webstore/detail/stopfingerprinting/kfhlgmfkolojpnmhgggilmillpcokmnb?hl=en-US.

[21] G. Wilson. "User-Agent Switcher for Chrome," 15/05/2015; https://chrome.google.com/webstore/detail/user-agent-switcher-for-c/djflhoibgkdhkhhcedjiklpkjnoahfmg.

[22] askeing. "Canvas Fingerprint Blocker," 20/06/2016; https://addons.mozilla.org/en-US/firefox/addon/canvas-fingerprint-blocker/?src=api.

[23] K. Boda. "Firegloves," 23/04/2015; http://fingerprint.pet-portal.eu/?menu=6.

[24] NiklasG. "Stop Fingerprinting," 20/06/2016; https://addons.mozilla.org/en-US/firefox/addon/stop-fingerprinting/?src=api.

