The effectiveness of a browser fingerprint as a tool for tracking

Mick Vaites

February 2013

This document is the dissertation by Mick Vaites part submission for the Postgraduate Open University Course T802 Research.

Abstract

Browser fingerprints are derived using information collected about the configuration of a users computer. The permutations are sufficiently distinct that they can be used as a tool for tracking. Fingerprints, unlike cookies, are more difficult for a user to influence because they cannot be deleted.

The aim of this research was to ascertain whether techniques could be identified to reduce the effectiveness of a fingerprint for tracking. A literary review identified attribute collection techniques and a way to rank them, based on their contribution towards the effectiveness of a fingerprint. This revealed that, as updates were applied to computers, fingerprint attributes changed naturally over time. Researchers pointed out that this had required them to develop algorithms to detect a returning browser fingerprint. This suggests that, in introducing legitimate but unexpected changes, the fingerprint could be more difficult to recognise; thus making it less effective for tracking.

Experiments were designed, using the algorithms as inspiration, to understand if changes could be applied to the fingerprint attribute, without effecting the operation of the browser, and reducing the likelihood of it being recognised as a returning browser. When testing to understand the impact on the operation of the browser, four Internet sites were selected and the highest failure rate observed was 18.7%. To understand if the techniques could be used to alter the fingerprint, tests were conducted. Three out of the four Internet fingerprint sites observed a change for 50% of the tests. One of the sites had the capability to detect a returning browser and in 16.1% of the tests the certainty of a match was reduced to less than 90%. Experiments were then conducted to test the hypothesis, that the match certainty could be reduced by changing multiple attribute elements at the same time. Findings revealed that changing two elements consistently achieved a match certainty of 95%, reducing to less than 85% for six attribute elements.

The research conducted demonstrates that there is potential for techniques to manipulate the fingerprint by altering the attributes and reducing its effectiveness for tracking. However, these findings must be taken in context. The scope of the research was limited by the time available, therefore only the four most popular browsers were used and Internet hosted sites were used as laboratory instruments. This implies that there is an opportunity for further research to expand the scope and build on the experiments that were conducted, as I have outlined in the conclusion.

Contents

Figures

Tables

Glossary

Browse	A term used to describe a users movement across the web, navigating from page to page, using a browser
Browser	A software program used to access webpages
Cookie	Small data files written to computer and used by websites to 'remember' information that can be used by a returning visitor
HTTP	The protocol used by webservers to format pages that are displayed by web browsers
JavaScript	A scripting language originally developed by Netscape, that runs from within the browser on the users computer
Web Browser	See Browser
Webpage	A document that is part of a website and designed to be distributed by a webserver
Webserver	A server computer that delivers web pages to browsers
Website	A collection of related web pages

Acknowledgements

I would like to thank my supervisor, Dr Ian Newman for his valuable support and advice towards this dissertation and my mother, Joan, for proofreading the text.

Special thanks go to my wife, Sarah and children, Joshua, Imogen and Patrick for their patience and support in allowing me to complete this research.

Chapter 1 Introduction

1.1 Background to the problem or issue

Tracking people as they browse websites on the Internet allows a behavioural profile to be created which, Bohi (2010) reports, marketers use to understand the interests and habits of potential customers. This information is invaluable and has created a multi-billion dollar industry (Isaac, 2012) as it provides very targeted and therefore affordable advertising to many companies (Tsukayama, 2012).

Using tracking information in this way may seem to cause no harm if only used for advertising. The problem Madrigal (2012) reports, is that the analytical systems operated by tracking companies do not understand the sensitivities of presenting advertising information. For example, something a person may wish to keep private, or not wished to be reminded of, being advertised to them just because they researched it on the web. Furthermore building behavioural profiles based on the websites that a person visits is not a complete picture and Angwin (2010) warns, could be liable to misinterpretation causing inappropriate advertising to be shown.

The most common tool used for tracking is the cookie and a recent survey conducted by Purcell et al. (2012) showed that people are concerned about their privacy and, not wishing to be tracked, routinely delete them (Sipior et al., 2011). Furthermore, modern web browsers have private browsing modes which, if enabled, delete locally cached information, including cookies, when the browsing session is ended (Said et al., 2011). Another problem for trackers, is that in May 2012, the EU passed legislation mandating that unless a cookie was needed as part of the operation of a website, which excludes tracking cookies, consent must be obtained from the 'visiting' user (Information Commissioner's Office, 2012). So with the accuracy of their tracking databases under threat, Angwin and Valentino-Devries (2010) reports that tracking companies have started making use of an alternative, and potentially more persistent identifier, known as the browser fingerprint.

The browser fingerprint is computed from information about the users computer and browser, and therefore cannot be just deleted. It is generated by combining information collected:

1. **From the log files of visited websites**, typically containing: the browsers Internet address; the underlying operating system; the browser types; version number; and any installed browser plugins and enhancements (Baviskar and Thilagam, 2011, Schmücker, 2011).

2. **Using JavaScript code**, originally designed to provide interactive capabilities to web pages (Vander Veer, 2004, Wikipedia, 2012), JavaScript code can retrieve information from the browser environment and transmit it back to the webserver (Keith and Sambells, 2010, pp. 1-6). This includes identifying which fonts are installed, the screen resolution and, since JavaScript code executes on the users computer, it can be used to generate performance information about the computer that can be compared against known benchmarks (Mowery et al., 2011).

If this information is collected, combined and analysed, a distinct fingerprint can be computed and because it was derived from browser environmental information, unlike cookies, it cannot simply be deleted (Eckersley, 2010), making it a reliable resource for tracking companies.

Companies like **BlueCava Inc.**, and **41st Parameter Inc.**, originally introduced fingerprint-tracking technologies, not dependent on cookies, as part of fraud prevention services. The motivation being that fingerprints are not susceptible to interference by users deleting cookies, and are therefore more reliable for tracking. However, lured by the revenues available from online advertising, they have refocused their fingerprinting services to include the advertising market. Using fingerprint technology

BlueCava Inc. is building a "credit bureaux for devices" in which every computer or smart phone will have a "reputation" based on it's users online behaviour, shopping habits and demographics. With the aim of selling the information to those advertisers prepared to pay for granular data about peoples interests and activities (Angwin and Valentino-Devries, 2010), they claim to have one billion fingerprints catalogued in 2011 and aimed to double this number during 2012 (Giles, 2011). In trials **SteelHouse Inc.**, a behavioural marketing company, managed to accurately track 89% of visitors using fingerprints supplied by **AdTruth Inc.** (a division of **41st Parameter Inc.**) compared to only 78% using cookies. It is believed that the difference was due to users being able to delete cookies (Angwin and Valentino-Devries, 2010), and the result of this success is that they intend to utilise the services of **AdTruth Inc.** and totally replace cookies as a means of tracking (AdTruth, 2011).

1.1.1 Conclusion

Tracking browser activities to enable targeted advertising by generating behavioural profiles has built a multi-billion dollar industry and now companies looking to improve their competitive edge are using fingerprints as they are seen to be more reliable than cookies. However, tracking web activity to produce behavioural profiles does not create a complete picture and mistakes can be made, which because mitigation is not accessible is made worse if the advertisers are using fingerprints. The challenge is to give users with privacy concerns control back so that they can, if they wish, protect themselves from being tracked using a fingerprint.

1.2 Justification for the research

The principle justification for this research is that the use of fingerprint technology for tracking cannot be detected or, currently, be easily avoided (Eckersley, 2010), rendering current privacy options ineffective.

People are concerned that targeted advertising generated from tracking profiles, is an invasion of privacy and removes their right to choice (Purcell et al., 2012). Furthermore Angwin (2010) points out that a profile is an incomplete picture, as it is generated by analysing only the websites that a browser has visited. Madrigal (2012) tells us that the analytical systems that process the information do not understand the sensitivities associated with the tracking information that has been collected. This implies that, without asking a users permission, inappropriate personalised advertisements can be displayed on their computer screen.

When cookies are used for tracking, privacy options exist as a user can, if they choose to, delete them (Sipior et al., 2011). This is not the case when fingerprint technology is used, as it is computed from information about the browser environment, something which is not so easy for a user to manipulate (Broenink, 2012, Eckersley, 2010). A fact exploited by **Steelhouse Inc.**, a behavioural marketing company, who have actively stated that they are moving away from the use of cookies for tracking as fingerprints are more reliable (AdTruth, 2011). **BlueCava Inc.** and **41st Parameter Inc.** are actively building and then selling access to their browser fingerprint databases, for the express purpose of giving analytics companies, like **SteelHouse Inc.**, a system for tracking that is less susceptible to interference by users than cookies (Angwin and Valentino-Devries, 2010). It is therefore desirable to investigate how the browser privacy capabilities may be enhanced to mitigate tracking through fingerprinting.

1.3 Aim and Objectives

The aim of this research is to investigate the extent to which it may be possible to reduce the effectiveness of tracking using a browser fingerprint.

The objectives to achieve this aim are as follows:

1. Identify aspects of the browser information that could be suitable for creating a trackable fingerprint.

2. Evaluate critically the effectiveness of the identified information and how aspects contribute towards the fingerprint.

3. Explore how these aspects can be changed without effecting the operation of the browser.

4. Formulate recommendations as to how aspects of browser information can be changed to make the fingerprint less effective.

1.4 Definitions

For the purposes of this report:

- The term “attribute” is used to identify an aspect of the browser that contributes towards a trackable fingerprint.

1.5 Scope of the research

The scope of this research was to investigate whether a browser fingerprint could be manipulated in such a way to reduce its effectiveness as a tool for tracking. The primary focus of the research was on the browser attributes that make up the fingerprint and whether they could be manipulated.

1.6 Outline of the dissertation

This dissertation is structured as follows:

Chapter 2: provides an analysis of the practical problem and a review of academic and grey literature on the issue, culminating in a conclusion and leading to primary research questions.

Chapter 3: describes the methodology employed in the primary research with an appraisal of its validity, and details of the procedures used.

Chapter 4: provides a summary and then an analysis of the data, followed by discussions into the relevance of the findings in regards to the research aim and then the questions.

Chapter 5: provides a conclusion to the results of the research, suggests further research and then reflections on the research process.

Chapter 2 Research Definition

2.1 The Practical Problem

"Real browser fingerprints are the result of decentralised decisions by software developers, software users, and occasionally, technical accident. It is not obvious what the set of possible values is, or even how large that set is. Although it is finite, the set is large and sparse, with all of the attendant problems for privacy that that poses." (Eckersley, 2010)

The point Eckersley is making is that because of the chaotic way the web has developed there is a substantial amount of information accessible to a webserver which, under normal circumstances, is sufficient to produce a trackable fingerprint.

Cookies are objects left behind on a users computer, and current privacy measures are effective as they can be deleted (Schmücker, 2011). The problem with a fingerprint, as Mayer (2009) points out, is that it is not something left behind, but is created by correlating information from the browser environment. The result is an identifier that is sufficiently distinct to track, and unaffected by current privacy measures. Eckersley (2010) claims, that this is made worse, as attempts to reduce the information quite often creates the paradox, that the resultant change is more distinct than the aspect being manipulated.

2.2 Existing Relevant Knowledge

The research aims to investigate whether it is possible to reduce the effectiveness of a browser fingerprint when used for tracking. The following review of literature will investigate browser aspects (or attributes) that can be used to build an effective trackable fingerprint; demonstrate how they are collected; how much they contribute to making the fingerprint distinctive; the techniques that could be used for tracking and mechanisms to avoid being tracked using a fingerprint.

Two techniques have been identified for collecting fingerprint attributes, real-time analysis and data mining. This review will address them separately before establishing attribute significance, techniques for tracking and potential mitigation. Finally, conclusions will be drawn that aim to satisfy the research objectives; identifying gaps in the knowledge leading onto research questions.

2.2.1 Fingerprint Identification using Real-Time Analysis

This sub-section focuses on the collection of fingerprint attributes by analysing the conversation between the browser and the webserver, and by using crafted JavaScript code to collect environment or performance information.

Mayer (2009) was one of the first to discuss creating a trackable fingerprint using browser environmental information. His view was that as people were individuals, they would customise their computers by installing software, adding fonts and installing browser plugins. This had the potential to make their computer unique, and using JavaScript to access the information, a trackable fingerprint could be created.

Eckersley (2010) of **The Electronic Frontier Foundation** took this further by setting up the 'Panopticlick' project. People were encouraged to visit the project website, allowing their browser to be scanned, and during the one month period of testing in excess of one million visits were observed. To build a trackable fingerprint Eckersley (2010) selected eight variables (or attributes) from the browsers environment that would be accessible to the webserver (Table 2-1).

Table 2-1 Panopticlick browser characteristics, variable and source (Eckersley, 2010, Table 1)

Variable	Source	Remarks
User Agent	Transmitted by HTTP, logged by server	Contains Browser micro-version, OS version, language, toolbars and sometimes other information, e.g. details of installed plugins.
ACCEPT headers	Transmitted by HTTP, logged by server	
Cookies enabled?	Inferred in HTTP, logged by server	
Screen resolution	JavaScript AJAX post	
Timezone	JavaScript AJAX post	
Browser plugins, plugin versions and MIME types	JavaScript AJAX post	Sorted before collection. Microsoft Internet Explorer offers no way to enumerate plugins; the PluginDetect JavaScript library was used to check for 8 common plugins on that platform, plus extra code to estimate the Adobe Acrobat Reader version
System Fonts	Flash applet or Java applet, collected by JavaScript/AJAX	Not sorted. Note in 200 cases Mac OSX periodically changed the sort order of the "Lucidia" family
Partial supercookie test	JavaScript AJAX post	Tests for Flash LSO cookies, Silverlight cookies, HTML 5 databases and DOM globalStorage were not implemented.

Note: The variable column identifies the attribute, 'logged by server' in the source column means that the information was sent by the browser to the server in a standard HTTP header, which is part of a normal dialogue (Fielding et al., 1999), the server just had to record it; whereas 'Java AJAX post' means that JavaScript code needed to collect the information and post it back to the server for processing.

Boda et al. (2011) identified a number of shortcomings of 'Panopticlick', namely that the solution was browser specific and that a combination of Java and Flash plugins were required to retrieve the fonts list across a range of browsers. Their solution was to omit the list of browser plugins, which are browser specific, and use JavaScript and a table of basic system fonts to identify fonts that are browser independent and installed, without the need for Java or Flash. Table 2-2 lists the fields tracked by Boda et al. (2011), where the 'short user ID' is a hash constructed from the first two bytes of the Internet address, the operating system, the screen resolution, the timezone and the basic fonts. To test their solution Boda et al. (2011), developed a fingerprinting website and from a dataset of 989 visitors, managed to recognise 28% as visiting from multiple browsers on the same computer.

Table 2-2 Database structure: field names and their content (Boda et al., 2011, Table 1)

Field Name	Content
Locality	Hungarian or international
short user ID	user ID in a shorter, hashed format
created	time of fingerprint creation
ip	visitor IP address in hashed format
UAS	the User Agent string of the browser
os	operating system
screen	screen resolution
timezone	time zone
basic fonts	standard font list for user ID generation
all fonts	all detected installed font list stored for analysis

A completely different approach was taken by Mowery et al. (2011), with the aim of identifying the browser type and version, using custom JavaScript code to profile browsers. Using the SunSpider[1] and V8[2] JavaScript benchmarks, a thirty nine dimension fingerprint array was created from the timing

[1] SunSpider is the JavaScript benchmarking kit by the WebKit team and more information can be found at http://www.webkit.org/blog/152/announcing-sunspider-09/

[2] Google V8 is a set of JavaScript benchmarking scripts found at https://developers.google.com/v8/benchmarks

results. In lab conditions a total 1015 sets of fingerprints were taken across multiple configurations of hardware, operating system and web browser. The results showed that browsers could be identified with an accuracy of 98.2% in these lab conditions. A problem identified in the solution was the influence of jitter, the variation in delay in the time the browser took to complete, before the next test could be started. To overcome this problem an 800ms wait was introduced between tests, which Mowery et al. (2011) identified as a major weakness of the solution, since overall three minutes would need to be added for each complete test cycle.

2.2.2 Fingerprint Identification using Data-Mining

This sub-section focuses on the analysis of network and web server logs to identify attributes that may be used to form a distinct fingerprint. Two papers have been identified. The first by Yen et al. (2009) uses course network logs to identify the browser and, the second by Yen et al. (2012) investigates whether hosts (web browsers) can be tracked using the logs from **Microsoft's Hotmail** and **Bing** services.

Yen et al. (2009) identified nine attributes that could be collected from the flow logs (Table 2-3).

Table 2-3 Main features extracted for retrieval when profiling browser flow logs

Flow Statistics	1. Byte count (in each direction) 2. Packet count (in each direction) 3. Flow duration 4. Number of flows active simultaneously to this one 5. Start time minus the most closely preceding flow start time
Retrieval Statistics	6. Total number of flows 7. Cumulative byte count from flow destination 8. Cumulative flow duration 9. Retrieval duration

The experiment used four browsers (Internet Explorer, Firefox, Chrome and Safari), which were configured to collect the home page of http://www.cnn.com along with all associated graphics files. Attributes were collected from the log files and plots for each browser against flows were created, with the following findings:

1. When plotting the cumulative number of packets sent by each of the four browsers against the flows, the following traits were observed: Firefox initiates more flows than the other browsers, Opera sends more packets in earlier flows and Safari sends fewer packets overall.

2. Plotting the cumulative time between consecutive flows showed that Firefox multiplexed the retrieval of content across more flows than the other browsers.

However these results were taken in lab conditions that only provided a way to differentiate between the browsers operating from identical environments. To overcome this Yen et al. (2009) used Support Vector Machines (SVM), which are learning models able to analyse and classify data (Hall et al., 2009), providing a mechanism to compare pairs of browser data.

Figure 2-1 shows an SVM plot of two sets of profile data with a hyperplane that separates them, the further the attribute is from the hyperplane, the more certain it is to belong to that browser. So Yen et al. (2009) introduced the notion of a confidence, which is a minimum threshold distance from the hyperplane. Any attribute on or inside the threshold is considered invalid and not used in the classification. To avoid errors introduced when a browser with a small number of transactions was observed, Yen et al. (2009) introduced a further check requiring 30 hits by a specific browser before it was included. Their solution was then able to correctly classify a browser with 75% certainty, increasing to 100% as the confidence was decreased.

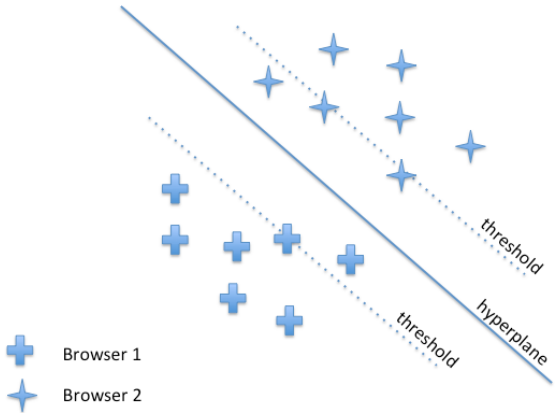

Figure 2-1 SVM Plot of Browser 1 and Browser 2

Yen et al. (2012) completed a large scale study to see what information was revealed by hosts (web browsers) using logs, from **Microsoft Bing, Hotmail** and the 'Windows Update Service'. From each service a dataset of attributes was derived, which is detailed in Table 2-4.

Table 2-4 Data Set Attribute Details (Yen et al., 2012)

Dataset	User-Agent information	IP Address	Timestamp	ID	Unique IP Addresses
Hotmail	OS and browser type	Yes	Yes	User-ID	308 million
Bing	User-agent string (UA)	Yes	Yes	Cooke-ID	131 million
Windows Update	N/A	Yes	Yes	Hardware-ID	74 million

The dataset from the 'Windows Update Service' contains a globally unique Hardware-ID, allowing it to be used for the validation of the fingerprints. So Yen et al. (2012) made the assumption that a perfect fingerprint, one that is unique, will only cross reference a single Hardware-ID. This allows a calculation of how accurately the fingerprint can be used, to single out an individual browser (its precision) or recognise it when returning (its recall). In this case Yen et al. (2012) calculates the precision as the percentage of fingerprints that correspond to a Hardware-ID and the recall as the percentage of Hardware-ID's that correspond to one fingerprint. Table 2-5 presents the results of the precision and recall calculations.

Table 2-5 Common identifiers, Precision and Recall (Yen et al., 2012)

Fingerprint Attribute(s)	Precision (%)	Recall (%)	Fingerprint Count	Hardware ID Count
UA	62.01%	72.11%	245,762	3,073,690
UA, Internet Address	80.62%	68.84%	1,685,416	1,771,907
UA, /24 IP Prefix	79.33%	69.43%	1,652,546	1,772,104
Cookie-ID	82.35%	68.64%	1,340,635	1,375,074
Cookie-ID (with HostTracker)	79.74%	99.13%	713,110	1,001,450
User-ID (with HostTracker)	92.82%	93.51%	4,608,980	4,820,116

Notes:

1. The UA is an abbreviation for the User-Agent
2. The '/24 IP Prefix' refers to a network of 254 hosts.
3. HostTracker, identifies hosts sharing the same Internet connections by analysing the application login information. Their assumption is that if the same application login identifiers keep occurring on the same Internet addresses then they are most likely to be from multiple hosts sharing the same Internet connection (Xie et al., 2009).

2.2.3 Contribution to effectiveness of fingerprint (Entropy)

For a fingerprint to be effective it should ideally be unique, Eckersley (2010), Broenink (2012), Boda et al. (2011) and Yen et al. (2012) took the probability of each collected attribute value and, using Shannon's formula (Shannon, 1948, pp. 10-12) for entropy, which provides a mechanism to calculate how unique a specific value is, based on the amount of information it contains, calculated the number of values that needed to be observed on average, before duplication occurred. In this way they were able to give attributes a ranking. The higher the entropy, the greater its contribution towards the effectiveness of a trackable fingerprint. Table 2-6 presents a comparison of the attributes entropy identified in the papers, and the average probability of a duplicate value being observed.

Table 2-6 Comparison of attribute entropy and probability of duplicates occurring

	(Eckersley, 2010)		(Broenink, 2012)		(Boda et al., 2012)		(Yen et al., 2012)	
Attribute/Variable	**Entropy (bits)**	**Duplicate probability**	**Entropy (bits)**	**Duplicate probability**	**Entropy (bits)**	**Duplicate probability**	**Entropy (bits)**	**Duplicate probability**
User-Agent	10.00	0.098%	6.31	1.260%	8.10	0.366%	11.59	0.032%
Accept (All Fields)	6.09	1.468%						
Accept			2.14	22.688%				
Accept-language			4.25	5.256%				
Accept-encoding			1.84	27.932%				
Accept-charset			1.83	28.126%				
Connection			0.37	77.378%				
DNT (Do Not Track)			0.57	67.362%				
Plugins and Versions	15.40	0.002%	6.25	1.314%				
Fonts List (All)	13.90	0.007%	6.23	1.332%	8.57	0.263%		
Universal Fonts					6.83	0.879%		
Detected Fonts					7.63	0.505%		
Screen Resolution	4.83	3.516%	4.58	4.181%				
Supercookies Partial	2.12	23.005%						
JavaScript Enabled			0.79	57.834%				
JavaScript Version			2.09	23.488%				
Platform (OS)			2.13	22.846%				
Charset			1.63	32.309%				
Language			2.24	21.169%				
Java Support			0.95	51.763%				
Timezone	3.04	12.158%	1.82	28.322%	2.22	21.464%		
Cookies Enabled	0.35	78.295%						
Panoptilick*	18.10	0.00036%						
User ID*					9.03	0.191%		
User-Agent + Internet Address*							20.29	0.000078%
Sample Size	470161		1124		989		1771907	

*Notes: All entropy calculations are based on unique visitors, * Attribute combinations identified by Eckersley (2010), Boda et al. (2011) and Yen et al. (2012) are included for comparison.*

Yen et al. (2012) calculated the entropy for the combination of User-Agent and Internet Address to be 20.29 bits, compared to 18.1bits for the whole fingerprint identified by Eckersley (2010), suggesting the significance of the Internet Address. However they also pointed out that despite its high value for

entropy, only 62% of their dataset had a unique User-Agent suggesting that large numbers must be identical. This they deduced meant that a fingerprint could be made less distinct by changing the User-Agent to one that was more common.

2.2.4 Detecting a returning browser

Another consideration covered by Eckersley (2010) was how to detect a returning fingerprint, given that the attribute values may change over time, making a returning browser more difficult to detect. To address this a simple algorithm was introduced which compared all eight attributes of a visitor (as listed in Table 2-1) to sets previously recorded as unique visitors. If all attributes were identical, except for the 'User-Agent', the plugins, or the fonts and they were less than 15% different[3] a match was recorded.

In total the Panopticlick website received over one million hits, which was reduced to 470,000 individual visitors identified using a three month cookie. From them 83% were seen as unique, when the browser plugin information was excluded, increasing to 94.2% when it was included and returning browsers were identified by the algorithm with 99.1% accuracy. A primary reason given by Eckersley (2010) for the high success rate was attributed to version and sub-version details included in the browser identification string (User-Agent) and the browser plugins list. An aspect Eckersley (2010) identifies, where this tracking is proven to be less effective, is with smart phones which do not currently have support for so many plugins.

Further research was completed by Broenink (2012) who created a website (letmetrackyou.org) and received 1124 visitors. What made Broenink's work different was his approach to detecting returning browsers. An alternative algorithm was introduced which takes into account things that don't change and things that do and how. For example, the browser name and operating system cannot change, but the number of fonts could increase and versions associated with plugins can go up but not down. The browser properties along with their associated rules are listed and detailed in Table 2-7.

Table 2-7 Ruleset for recognising fingerprints over time (Broenink, 2012, Table 3)

Property	Assumpted Rule
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 increase
DNT (Do Not Track)	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
Cookies Enabled	does not change (*)
Java support	does not change
Screen resolution	does not change (**)
Timezone	does not change (Daylight Saving Time corrected)
Plugin versions	do not decrease
Font List (All)	order does not change, fonts are not removed, fonts may be added between

** attributes the can be easily amended by the user, ** screen resolution effected by external monitors being plugged in.*

[3] The python built in diff.SequenceMatcher library is used (PSP, 2008).

When tested the algorithm accurately identified 86% of returning visitors over time; Broenink (2012) suggests that the screen size could be a wildcard attribute due to the increase in numbers of people using laptops with external screens or overhead projectors. To identify how this impacted on the algorithm, Broenink (2012) recorded the results with and without the screensize attribute. From the results, shown in Table 2-8, it was identified that overall the inclusion of this attribute improved the results. It was also observed that whilst more false positives were observed without the screensize attribute, the number of false negatives slightly reduced.

Table 2-8 Results of testing variation over time algorithm (Broenink, 2012)

	Without Screensize	With Screen Size
Correct identification	83.3%	86.6%
False Positive	13.2%	8.6%
False Negative	3.5%	4.8%

Stocks (2012) used a more granular approach for detecting a returning browser. The fingerprint attributes were each assigned a weight representing its contribution towards the effectiveness of the fingerprint. To find the most likely match, attributes were collected from the visiting browser and for each of the fingerprints on file a comparison score is computed to identify how likely a match is to exist. Attributes were compared using different tests, which are listed and detailed in Table 2-9. Once the tests have been completed the fingerprint on file with the highest score will be selected. To calculate the maximum score possible the visitors attributes are analysed against themselves. Then, if the scores associated with the selected fingerprint and the visitor are within 15%, they are deemed a match.

Table 2-9 Returning browser comparison tests

Test	Description
UA Calc	The browser name and version are taken from the visiting User-Agent and fingerprints are selected with the same values. Each type of browser has a weighting depending on its popularity. This value is used as the initial score for the compared fingerprint.
Plugin Calc	A version comparison is performed between the visitors plugins and the ones found in the fingerprints. In the event that the 'visitors' plugin is found and its version is higher then the weight associated with the plugin is multiplied by the difference in version numbers. The result is added to the fingerprints score.
Font Calc	If the visitor has fonts that are not in the fingerprint, the weighting associated with fonts is multiplied by the difference in number and this result is added to the fingerprints score.
Other Calc	A simple comparison of the remaining attributes is completed where the value of its weight is added to the fingerprints score if a match occurs.

2.2.5 Browser Fingerprint Mitigation

Over half the high entropy attributes identified by Eckersley (2010) are accessible using JavaScript code, but a number of browser enhancements (plugins) are available that can selectively disable it running from specific websites (Schmücker, 2011). Unfortunately only disabling JavaScript from certain websites creates a paradox, as this creates a potentially more distinct behaviour than the one being avoided (Broenink, 2012). In addition to this plugins tend to be browser specific, a fact that may too form part of a fingerprint (Eckersley, 2010).

An alternative, proposed by Baviskar and Thilagam (2011), is a browser plugin, which dynamically disables JavaScript code before it is interpreted by the browser. The plugin intercepts the web page and changes its content, rewriting the JavaScript code to disable attributes, before releasing it to be processed by the web browser. Figure 2-2 illustrates the process the plugin follows to achieve this.

However, Broenink (2012) argues that the result of its operation can also present an unnatural behaviour, so it too could create the paradox, that the fingerprint is more distinct than it would have been without the mitigation being attempted.

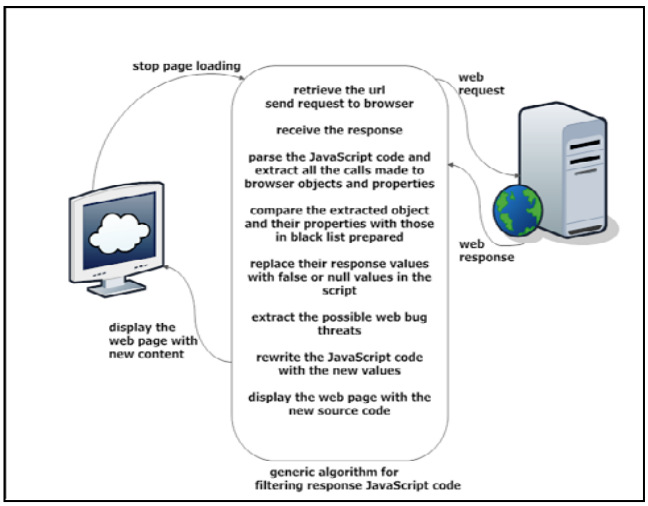

Figure 2-2 Browser plugin message flows (Baviskar and Thilagam, 2011)

Yen et al. (2012) suggests that the Internet address of the browser, is a valuable tracking attribute with high entropy. So with the aim of obscuring this attribute, the TOR Network (Dingledine et al., 2004) operates as an onion router environment. To avoid being tracked packets are first wrapped in an encryption layer (hence the onion metaphor) and then, to hide the original Internet address packets are bounced randomly through a global network of relays (or TOR nodes). Only being decrypted as they leave heading for the server the browser is communicating with. Figure 2-3 illustrates an example where the user Alice is communicating over the TOR network with a server called Bob.

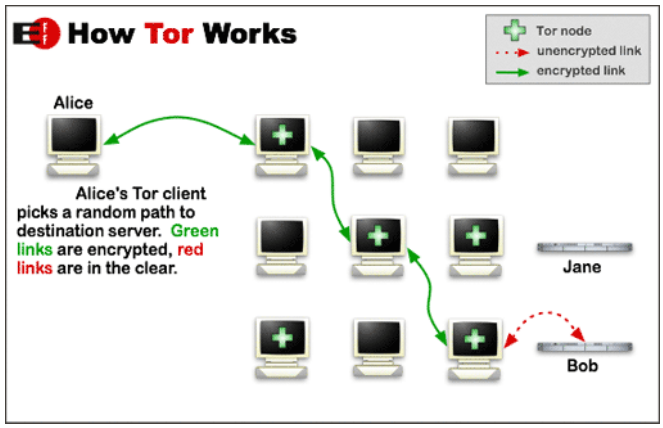

Figure 2-3 Communication example over TOR (The Tor Project, 2012)

Unfortunately, to maintain low latency[4] communications, no attempt is made by the network to hide other attributes that may be used to fingerprint the web browser (Eckersley, 2010). In part this is addressed using Privoxy, a personal proxy which can be installed on the users computer, by Keil et al. (2012), and provides privacy enhancements, including the ability to manipulate cookies and the HTTP headers, before they are transmitted to the webserver (Schmücker, 2011).

A further enhancement is the TOR browser, as documented by Perry et al. (2011), which uses a common User-Agent, disables plugins and uses default values for some other attributes. The problem as

[4] Latency is a measure of time delay experienced in the system

Broenink (2012) points out is that only a small subset of people are likely to use this solution making its use distinct and therefore trackable.

2.2.6 Summary and Conclusions

The first research objective was to identify the browser attributes that could be used to form a trackable fingerprint. This was achieved by the review and the attributes are listed in Table 2-10. Eckersley (2010), Broenink (2012) and Stocks (2012) collected attributes by analysing the browser environment and used them to create browser fingerprinting websites, and Boda et al. (2011) built a site that could identify a computer by excluding browser specific attributes.

Other attributes, browser profiling statistics, described by Mowery et al. (2011), and flow logs, described by Yen et al. (2009), are excluded as they aim to identify the browser type and not a specific instance. The Internet address has also been omitted, as the TOR network, described by Dingledine et al. (2004) combined with Privoxy provides a robust mitigation (Keil et al., 2012).

Table 2-10 Fingerprinting sites and attributes identified by literature review

ID	Paper	Abbrieviation	Which attributes are tested	Website Address
1	(Eckersley, 2010)	Panopticlick	User-Agent Accept (All Fields) Plugins and Versions Timezone Screen Resolution Fonts List (All) Cookies Enabled Partial supercookie	**http://panopticlick.eff.org**
2	(Broenink, 2012)	LetMeTrackYou	User-Agent Accept Accept-Language Accept-Encoding Accept-Charset Connection DNT (Do Not Track) JavaScript Enabled JavaScript Version Platform Charset Language Cookies Enabled Java Support Timezone Plugins and Versions Screen Resolution Fonts List (All)	**http://letmetrackyou.org/identify.php**
3	(Stocks, 2012)	HowUniqueAreYou	User-Agent Accept Accept-Language Accept-Encoding Accept-Charset Plugins and Versions Fonts List (All) Screen Resolution Timezone	**http://howuniqueareyou.net**
4	(Boda et al., 2011)	PetPortal	User-Agent Platform (OS) Screen resolution Timezone Detected Fonts	**http://pet-portal.eu/fingerprint/**

Research objective two required that the effectiveness of an attribute and its contribution to the fingerprint is established. The papers discussed in this review use entropy as a mechanism to address this, suggesting that the quantity of information contained in an attribute is directly related to its usefulness for tracking. Yen et al. (2012) point out that caution must be used, and illustrate this by advising that, the 'User-Agent' has a high entropy but only 62% of their dataset were unique. To accommodate these concerns, the attributes are divided into three groups (high, medium and low) based on the relative importance of the attributes as indicated by the entropy value. The results are provided in Table 2-11, which addresses the requirements of this research objective.

Table 2-11 Attributes and collection techniques identified from literature review

Attribute	Entropy	Collection technique
User-Agent	High	A
HTTP Accept (All fields)	Medium	A
Cookies Enabled	Low	B
Screen Resolution	Medium	B
Timezone	Low	B
Plugins and versions	High	B
Font List (All)	High	B*
Partial supercookie	Low	B
HTTP Accept	Low	A
HTTP Accept-Language	Medium	A
HTTP Accept-Encoding	Low	A
HTTP Accept-Charset	Low	A
Connection	Low	A
DNT (Do Not Track)	Low	A
Universal Fonts	High	B
Detected Fonts	High	B
JavaScript Enabled	Low	B
JavaScript Version	Low	B
Platform (OS)	Low	B
Charset	Low	B
Language	Low	B
Java Support	low	B

Key to collection techniques:	
A	Transmitted in the HTTP header
B	JavaScript code collects attribute and results are transmitted back using a HTTP post
B*	In addition to JavaScript code, Java and Flash was also used to collect attribute data

2.3 Research Questions

The literature review was effective at addressing the first two objectives leaving primary research to complete objectives three and four (Table 2-12).

The third objective required techniques be explored that could alter the attributes without affecting the operation of the browser. Two types of attribute were identified in literature: HTTP header and those derived using JavaScript code (listed in Table 2-11), suggesting that more than one technique will be required to manipulate them. Two research questions were required to bridge the gap, firstly to identify attribute manipulations that could be used and secondly to ask which ones could be used without impacting the operation of the browser.

The final research objective is to make recommendations about how attributes could be changed to make the fingerprint less effective. Eckersley (2010), Broenink (2012) and Stocks (2012) suggest that some attribute changes can be anticipated, allowing them to formulate detection algorithms. Therefore if an attribute changes differently to what is expected, then this could alter the fingerprint so it is no longer recognised as belonging to the same browser.

To provide the data needed to complete the final research objective, two research questions were required; one to identify attribute changes that could affect the fingerprint and a second to ask whether implementing them could reduce the fingerprints effectiveness for tracking.

Table 2-12 Research questions

Objective 3:	What techniques are available for manipulating browser attributes? Can these techniques be used to manipulate the attributes without affecting the operation of the browser?
Objective 4:	Which attribute variable manipulation techniques can be used to alter a browser fingerprint? Can the manipulation techniques make the browser fingerprint less effective?

Chapter 3 Methodology

3.1 Methods and Techniques Selected

To answer the research questions it was necessary to determine which of the attributes identified in Table 2-11 could be amended, without effecting the operation of the browser. What techniques could be used to change the fingerprint and whether these changes reduced its effectiveness for tracking.

Laboratory experiments were considered to collect the data to develop the manipulation techniques, using a dedicated test website. However having confirmed that the fingerprinting sites, identified in Table 2-10, were still operational, 'in-situ' experiments were designed with the data being collected as the browsers 'visited' them.

Testing the effect of the manipulation techniques on the browser, and the ability to alter the fingerprint, required a more structured environment, so laboratory experiments were designed. Instead of developing test websites, ones hosted on the Internet were used as laboratory instruments. The motivation being that it would not be practical to design test websites with the required variety and capabilities needed in the time available.

3.2 Justification

The influence behind the selection of methodologies was primarily a result of the secondary research as it gave an insight into how the researchers had collected their data, and whilst the prevailing view was that experimentation would be the method of choice, others needed to be ruled out.

A case study requires multiple sources of evidence (Open University T802, 2010), so was ruled out because available research had been focused on using fingerprints for detection, and not the avoidance of tracking. This lack of evidence also meant that a survey would have needed to focus on expert opinions, most likely from the authors of the papers discussed in the literature review. This could open the potential for a much greater insight into possible attribute manipulation techniques. However the effort involved would have been disproportionate considering the time available and would only provide the data to answer the first question.

Eckersley (2010), Stocks (2012) and Broenink (2012) performed experiments to collect data with the aim of detecting a returning browser, so a further consideration was whether their data could be used to derive the manipulation techniques, instead of performing new experiments. The problem was that their research had not required the browser or the data that was transmitted to be controlled, meaning that any findings would produce inconclusive results. Mowery et al. (2011) and Yen et al. (2009) used experiments to collect data with the aim of differentiating browser types, so controlled the ones that were used, but their data did not relate to the attributes being considered in this research (See Table 2-11). The approach taken, since the fingerprint sites identified in Table 2-10 were still operational, was 'in-situ' experiments where the attribute data would be collected as the browsers 'visited' them and analysed to derive the techniques.

Having derived the manipulation techniques, the next task was to understand if their use impacted the operation of the browser. Baviskar and Thilagam (2011) performed laboratory experiments to test a browser plugin they had designed to manipulate JavaScript attributes, and a laboratory website was used as a measuring instrument. What was not discussed was whether any experiments had been conducted to understand the impact on a browsers operation, something needed in this research. Taking inspiration from their approach, laboratory experiments were designed but instead of designing a test site Internet sites were used.

The primary reason for using Internet hosted sites, as test instruments in laboratory experiments, was that it would not be practical to design test websites, in the time available and with the diversity needed to collect the data to answer the questions. The second reason was that the laboratory environment, that was used to collect the data to answer the first research question, could be used to answer the remaining questions, by selecting appropriate websites. Furthermore four websites had already been identified, listed in Table 2-10, which could be used to test for browser fingerprint changes.

3.3 Research Procedures

This section discusses the tasks and decisions taken to design the experiments that generated the data to answer the research questions. The first of which was to collect the data so that the manipulation techniques could be designed, followed by experiments to understand how they impact the operation of the browser and whether they could reduce the effectiveness of the fingerprint for tracking.

3.3.1 Raw attribute collection

The first task was to collect the raw attribute data so the manipulation techniques could be derived, and two types were identified by Eckersley (2010) and Broenink (2012).

1. HTTP header attributes are standards based and exist within the packets that are transmitted by the browser (Fielding et al., 1999).

2. JavaScript attributes are more complex, code is transmitted by the webserver to the browser, the browser executes the code causing data to be collected and transmitted back to the webserver for processing (Broenink, 2012, Eckersley, 2010).

Laboratory experiments were considered, with test websites designed to collect the raw attribute data, as they were 'visited' by browsers. However the approach was deemed impractical due to the time available to complete the research.

Another consideration was whether the data collected by the researchers discussed in the literature review could be used. The problem was that the research used to identify the attributes (See Table 2-10), had not needed to control the data transmitted by the browsers, so assuming that the required raw attribute data could be identified, the analysis to produce manipulation techniques would be inconclusive. Other research by Mowery et al. (2011) and Yen et al. (2009) had used specific browsers, but had been researching other attributes ruling out its use.

After confirming that the fingerprint sites, listed in Table 2-10, were still operational, the solution to collect the raw attribute data was, 'in-situ' experiments. A controlled set of browsers, installed on a test computer, 'visited' the fingerprint sites causing the attribute data to be generated. To collect the data, a web proxy was connected between the test computer and the Internet connection, configured to capture the content of the browser/website conversations.

Browsers were selected to enforce the validity of the experiments. The work undertaken by Perry et al. (2011) to deliver a browser that was less susceptible to fingerprinting was considered, but discounted being unlikely to be systematic of what a 'normal' user would be using. Instead the four most popular browsers, as reported by StatCounter (2012a) in October 2012, were selected.

Due to the time constraints, it was decided that only one version of each of the browser would be used and based on the recommendations by some of the vendors to 'update to avoid malware' (Mozilla, 2012, Apple, 2012), the current versions were used (see Table 3-1).

Table 3-1 Tested browsers

Browser	Abbreviation	Sequence to clear browsing data
Microsoft Internet Explorer 9.0	MSIE	Internet Options -> Delete (Browser History)
Mozilla Firefox 16.0	Firefox	Tools -> Clear Recent History -> Clear Now
Google Chrome 23.0	Chrome	History -> Clear all browsing data
Apple Safari 5.1.7	Safari	Preferences -> Privacy -> Remove All Website Data

To avoid the potential for cross contamination of data skewing the results, built in privacy mechanisms were considered but discounted because the implementations were not consistent (Aggarwal et al., 2010). Instead all browsing data was manually cleared before an experiment was performed.

The time constraints associated with this research also meant that it would only be practical to experiment with a single operating system and Windows 7 was selected. The primary reason being, the inclusion of Microsoft Internet Explorer necessitated a version of Windows, but secondly it was also reported to be the most popular operating system in October 2012 by StatCounter (2012b).

3.3.2 Designing the attribute manipulation techniques

Having collected the raw attribute data, to answer the remaining research questions manipulation techniques needed to be derived.

Table 3-2 Attributes, detection and detection avoidance rules

Attribute/tests	Attribute Detection Rule	Detection avoidance rules	Entropy
Accept (1)	Does not change	Make changes	Low
Accept-Language (2)	Does not change, but can be influenced by user	Make changes	Medium
Accept-Encoding (3)	Does not change	Make changes	Low
Accept-Charset (4)	Does not change	Make changes	Low
Connection (5)	Does not change	Make changes	Low
User-Agent (6)	Operating system and browser name does not change. Versions only increase in value.	Change operating system	High
User-Agent (7)		Change browser name	
User-Agent (8)		Reduce operating system version	
User-Agent (9)		Reduce browser version	
DNT (Do Not Track)(10)	User can enable/disable	Make changes	Low
JavaScript Enabled (11)	Does not change	Make changes	Low
JavaScript Version (12)	Does not decrease	Decrease version number	Low
Platform (OS)(13)	Does not change	Make changes	Low
Charset (14)	Does not change	Make changes	Low
Language (15)	Does not change	Make changes	Low
Cookies Enabled (16)	User can enable/disable	Make changes	Low
Java Support (17)	Does not change	Make changes	Low
Screen Resolution (18)	Does not change, but can be influenced by user	Make changes	Medium
Timezone(19)	Does not change	Make changes	Low
Plugins and versions (20)	Plugin versions do not decrease	Decrease plugin versions	High
Font List (All)(21)	Order does not change, fonts may be added and not removed	Change font order	High
Font List (All)(22)		Remove fonts	
Partial supercookie (23)	Does not change	Make changes	Low
Universal Fonts (24)	Does not change	Make changes	High
Detected Fonts (25)	Does not change	Make changes	High

Baviskar and Thilagam (2011) designed a browser plugin that manipulated attributes by replacing JavaScript code before it was interpreted by the browser. To achieve this their plugin replaced the code that retrieved the attributes with other JavaScript code and doing so manipulated the responses received by the webserver. Using this as inspiration, the raw HTTP headers and JavaScript code, captured from the conversations, was analysed and attribute manipulation techniques were defined.

The next aspect was to identify particular manipulations that could be used to change the browser fingerprint. Eckersley (2010), Broenink (2012) and Stocks (2012) reported that fingerprint attributes change naturally over time and developed rules that allowed them to detect a returning browser. Using the strategy of exploiting changes that are technically possible but unlikely, it was possible to derive detection avoidance rules (Table 3-2). Then by incorporating these avoidance rules into the attribute manipulation techniques, a series of attribute manipulation tests were defined that could be used to change the fingerprint, and potentially reduce its effectiveness for tracking.

3.3.3 Testing manipulations to understand browser impact

Once the attribute manipulation tests were designed it was necessary to understand whether their use would impact the operation of the browser. A laboratory website was considered, but ruled out in favour of experiments using Internet hosted sites. The motivation being that in the time available for the research, it would not be possible to develop a website with the diversity needed, whereas a selection of popular Internet hosted sites could. Four Internet sites were selected: 'YouTube' and 'BBC' to represent popular Internet sites, 'zeFrank' a popular interactive game site and 'BrowserHawk' which has features used by commercial sites to test browser compatibility, to improve the validity of the experiments. The sites are listed and detailed in Table 3-3.

Table 3-3 Internet Test Sites

Ref.	Site	Site Web Address
1	YouTube, the most popular video site in 2012 (eBizMBA, 2012)	http://www.youtube.com
2	BBC, the fifth most popular site in the UK (Alexa, 2012)	http://www.bbc.co.uk
3	BrowserHawk – Browser Capabilities test page, can detect browser capabilities and is recommended by Smith (2006)	http://www.cyscape.com/showbrow.aspx?bhcp=1
4	zeFrank – A kaleidoscope application, a popular interactive website identified by Touchton (2012).	http://www.zefrank.com/byokal/kal2.html

Each of the browsers 'visited' the Internet sites and the manipulations were applied on the web proxy. The data collected was qualitative in nature, so to simplify the recording process, scores of '0' to '3' were used ('0' the page failed to display, '1' an element on the page was missing, '2' functionality was impaired and '3' the page displayed correctly) which included two levels of what could be considered 'acceptable failures'.

3.3.4 Understanding how the manipulations affect the browser fingerprint

Having gathered data to understand how the manipulations impact the operation of a browser, their effectiveness against fingerprint tracking needed testing. To collect this information the existing environment was re-used as, the fingerprint sites could act as laboratory instruments and the web proxy had already been setup to implement the manipulations. Again the data collected from these experiments was qualitative in nature so a scoring of '0' to '2' ('0' the page failed to display, '1' the manipulation had no effect on the fingerprint and '2' the fingerprint was changed) was considered. However, the fingerprint site 'HowUniqueAreYou' also reported when it did not recognise a browser as returning and to capture this a score of '3' was used.

When 'HowUniqueAreYou' recognised a returning browser it also reported: the percentage fingerprint match, the attribute values that had changed and a timestamp of when the browser was last seen. This information was also collected, as it offered a mechanism to understand the effectiveness of manipulation techniques against fingerprint tracking. A concern over the validity of this data was that, because the site was accessible to the Internet and not exclusive to this research, there was a potential

for false positive matches. The solution to reduce this risk was to record and compare the timestamps when a match occurred. In this way spurious results could be identified.

3.4 Ethical considerations

The primary research carried out was a combination of 'in-situ' and laboratory based experiments producing qualitative data, and the results have been reported in a balanced and unbiased way. The experiments did not involve a third parties data, however Internet hosted sites were used so efforts were made to ensure that they were in no way compromised by the experiments.

In all cases where usage policies were identified, they were observed and in the case of the fingerprinting sites, whilst not required to do so, courtesy emails were sent to the operators advising them of the research being carried out.

Chapter 4 Analysis and interpretation

4.1 Summary of Data Collected

In the period 24th October to 17th November 2012 a combination of in-situ and laboratory experiments were conducted and the following is a summary of the data collected.

Using in-situ experiments, raw attribute data was collected for the twenty-five attributes listed in Table 2-11 and following the analysis in sub-sections 4.2.1 and 4.2.2, techniques were derived to manipulate eighteen of them. To provide a mechanism to test the techniques ability to impact the effectiveness of the fingerprint, thirty-seven attribute tests, which are listed and detailed in Table 4-1, were designed.

Table 4-1 Derived attribute manipulation tests

Attribute Tests	Attribute manipulations
Accept (1)	Add 'text/xml' or 'text/html' file types to the permitted list
Accept-Language (2)	Change language from US to GB
Accept-Encoding (3)	Disable 'gzip' encoding type by removing it from the list
Accept-Charset (4)	Replace the character set with 'ISO-8859-2'
Connection (5)	Swap the text 'keep-alive' and 'Keep-Alive'
User-Agent (6)	Change operating system from Windows to MacOS
User-Agent (7a)	Change browser from MSIE to Firefox
User-Agent (7b)	Change browser from MSIE to Chrome
User-Agent (7c)	Change browser from MSIE to Safari
User-Agent (7d)	Change browser from Firefox to MSIE
User-Agent (7e)	Change browser from Firefox to Chrome
User-Agent (7f)	Change browser from Firefox to Safari
User-Agent (7g)	Change browser from Chrome to MSIE
User-Agent (7h)	Change browser from Chrome to Firefox
User-Agent (7i)	Change browser from Chrome to Safari
User-Agent (7j)	Change browser from Safari to MSIE
User-Agent (7k)	Change browser from Safari to Firefox
User-Agent (7l)	Change browser from Safari to Chrome
User-Agent (8)	Reduce version of windows
User-Agent (9a)	Reduce version of MSIE
User-Agent (9b)	Reduce version of Firefox
User-Agent (9c)	Reduce version of Chrome
User-Agent (9d)	Reduce version of Safari
User-Agent (9e)	Reduce sub-version of Chrome
User-Agent (9f)	Reduce sub-version of Safari
Platform (OS)(13a)	Replace JavaScript variable 'navigator.platform' with 'MacIntel'
Platform (OS)(13b)	Replace JavaScript variable 'navigator.appVersion' with '5.0 (Macintosh)'
Charset (14)	Replace JavaScript variable 'document.defaultCharset' with 'ISO-8859-1'
Language (15)	Replace JavaScript variable 'navigator.language' with 'en-US'
Cookies Enabled (16)	Replace JavaScript variable 'navigator.cookieEnabled' with 'null'
Java Support (17)	Replace JavaScript variable 'navigator.javaEnabled()' with 'null'
Screen Resolution (18a)	Replace JavaScript 'screen height' variables with '780'
Screen Resolution (18b)	Replace JavaScript 'screen width' variables with '1024'
Screen Resolution (18c)	Replace JavaScript 'screen colorDepth' with '32'
Timezone (19)	Randomise the timezone, by replacing the JavaScript getTimeZoneOffiset method
Partial supercookie (23a)	Replace JavaScript variable localStorage. with notlocalStor.
Partial supercookie (23b)	Replace JavaScript variable sessionStorage. with notSessionStor.

Note: Raw attribute manipulation tests listed and detailed in Appendix Tables A-3 and A-4

A summary of the attribute data that was collected is presented in Appendix Tables A-1 and A-2, the full raw data is available electronically.

Having identified the manipulation techniques, 364 tests were conducted in laboratory experiments to understand the impact of the manipulations on the operation of the browser. For each of the attribute manipulation tests, the browsers 'visited' the test websites (identified as '1' to '4') and the results are presented in Table 4-2.

Table 4-2 Encoded summary of browser impact tests

Browser	MSIE				Firefox				Chrome				Safari			
Attribute Manipulation Tests (See Table 4-1) / **Test Website Ref. (See Table 3-3)**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Accept (1)	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Accept-Language (2)	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Accept-Encoding (3)	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Accept-Charset (4)									3	3	3	3				
Connection (5)	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
User-Agent (6)	3	3	**0**	3	3	3	3	3	3	3	3	3	3	3	3	3
User-Agent (7a)	3	3	3	3												
User-Agent (7b)	3	3	3	3												
User-Agent (7c)	3	3	**0**	3												
User-Agent (7d)					3	3	3	3								
User-Agent (7e)					3	3	3	3								
User-Agent (7f)					3	3	3	3								
User-Agent (7g)									3	3	3	3				
User-Agent (7h)									3	3	3	3				
User-Agent (7i)									3	3	3	3				
User-Agent (7j)													3	3	**0**	3
User-Agent (7k)													3	3	**0**	3
User-Agent (7l)													3	3	3	3
User-Agent (8)	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
User-Agent (9a)	3	3	3	3												
User-Agent (9b)					3	3	3	3								
User-Agent (9c)									3	3	3	3				
User-Agent (9d)													3	3	3	3
User-Agent (9e)									3	3	3	3				
User-Agent (9f)													3	3	3	3
Platform (OS) (13a)	3	**0**	3	3	3	**0**	3	3	3	**0**	3	3	3	**0**	3	3
Platform (OS) (13b)	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Charset (14)	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Language (15)	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Cookies Enabled (16)	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Java Support (17)	3	3	3	3	3	3	**0**	3	3	3	**0**	3	3	3	**0**	3
Screen Resolution (18a)	3	**0**	**0**	3	3	**0**	**0**	3	3	**0**	**0**	3	3	**0**	**0**	3
Screen Resolution (18b)	3	**0**	**0**	3	3	**0**	**0**	3	3	**0**	**0**	3	3	**0**	**0**	3
Screen Resolution (18c)	3	3	**0**	3	3	3	**0**	3	3	3	**0**	3	3	3	**0**	3
Timezone (19)	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Partial Supercookie (23a)	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Partial Supercookie (23b)	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

*Note: **0** – The page did not display, 3 – The page displayed correctly. Shaded cells indicate a permutation that was not tested.*

The permutations were only tested if relevant, 'Accept-Charset (4)' is only transmitted by Chrome so was not tested on other browsers and, 'User-Agent (7)' and 'User-Agent (9)' had browser specific elements so browser specific sub-manipulations were derived and tested. An unexpected outcome was that, even though two levels of 'acceptable failure' was allowed for, in all cases either the page displayed as expected (3) or it did not display at all (0). An analysis of Table 4-2 is provided in sub-section 4.2.4.

Laboratory experiments were then conducted to understand if the attribute manipulations tests could alter the fingerprint. The four fingerprint sites listed in Table 2-10, and numbered '1' to '4', were 'visited' by the test browsers and the encoded results are presented in Table 4-3.

Table 4-3 Encoded summary of fingerprint manipulation tests

Browser	MSIE				Firefox				Chrome				Safari			
Attribute Manipulation Tests (See Table 4-1) / **Fingerprinting Site (See Table 2-10)**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Accept (1)	2	2	2	1	2	2	2	1	2	2	2	1	2	2	2	1
Accept-Language (2)	1	2	2	1	2	2	2	1	2	2	2	1	2	2	2	1
Accept-Encoding (3)	2	2	2	1	2	2	2	1	2	2	2	1	2	2	2	1
Accept-Charset (4)									2	2	2	1				
Connection (5)		2				2				2				2		
User-Agent (6)	2	2	1	1	**0**	2	**0**	**0**	2	2	1	1	2	2	1	1
User-Agent (7a)	2	2	**3**	2												
User-Agent (7b)	2	2	2	2												
User-Agent (7c)	2	2	2	2												
User-Agent (7d)					2	2	2	2								
User-Agent (7e)					2	2	2	2								
User-Agent (7f)					2	2	2	2								
User-Agent (7g)									2	2	2	2				
User-Agent (7h)									2	2	2	2				
User-Agent (7i)									2	2	2	2				
User-Agent (7j)													2	2	2	2
User-Agent (7k)													2	2	2	2
User-Agent (7l)													2	2	2	2
User-Agent (8)	2	2	2	2	2	2	2	2	2	2	2	2	2	2	**3**	2
User-Agent (9a)	2	2	2	1												
User-Agent (9b)					2	2	2	1								
User-Agent (9c)									2	2	2	1				
User-Agent (9d)													2	2	2	1
User-Agent (9e)									2	2	2	1				
User-Agent (9f)													2	2	2	1
Platform (OS) (13a)	1	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1
Platform (OS) (13b)	1	1	1	2	1	1	1	2	1	1	1	2	1	1	1	1
Charset (14)	1	**0**	1	1	**0**	**0**	1	1	1	**0**	1	1	1	**0**	1	1
Language (15)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Cookies Enabled (16)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Java Support (17)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Screen Resolution (18a)	**0**	2	2	**0**	2	1	2	2	**0**	1	2	2	2	1	2	2
Screen Resolution (18b)	**0**	2	2	**0**	2	1	2	2	**0**	1	2	2	2	1	2	2
Screen Resolution (18c)	**0**	1	2	1	2	1	2	1	2	2	2	1	2	2	1	1
Timezone (19)	1	**0**	**0**	**0**	1	2	2	**0**	**0**	2	**0**	**0**	2	2	**0**	**0**
Partial Supercookie (23a)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Partial Supercookie (23b)	1	1	1	1	**0**	1	1	1	1	1	1	1	1	1	1	1

*Notes: **0** – The page did not display, 1 – The page displayed but the fingerprint did not change, 2 – The page displayed and the fingerprint changed, **3** – The page displayed correctly, the fingerprint changed and was not recognised as returning. Shaded cells indicate a permutation that was not tested.*

Note that in addition to the untested permutations already identified, 'Connection (5)' was only tested against 'LetMeTrackYou' because it was the only site to include it in fingerprint calculations.

In addition to reporting the fingerprint of the 'visiting' browser, 'HowUniqueAreYou' also compares it against fingerprints from previous visits. If a match of 85% certainty or greater is calculated then this is reported along with the percentage. This additional data, which is presented in full in Appendix Table A-5, was used to understand whether the manipulations could reduce the effectiveness of a fingerprint for tracking.

4.2 Data Analysis

In the following sub-sections the data collected from the experiments will be analysed: to allow the manipulations to be derived and then to understand whether they impact the operation of the browser or can alter the browser fingerprint reducing its effectiveness for tracking.

4.2.1 HTTP Attribute Analysis

An analysis of the HTTP header data in Appendix Table A-1 reveals the following:

- All browsers transmitted the HTTP attributes: 'Accept', 'Accept-Language', 'User-Agent', 'Accept-Encoding' and 'Connection', and there was little variation between occurrences of the same attributes when transmitted by the same browser.

- Chrome was the only browser to transmit the attribute 'Accept-Charset'.

- The amount of information contained in the headers transmitted by MSIE appeared less than the other browsers. The greatest difference was 'User-Agent', where for MSIE only four attribute elements were identified, compared to twelve elements for Chrome and Safari.

- The 'Connection' attribute was capitalised for MSIE, and lower case for the other three browsers.

- The attribute 'DNT (Do Not Track)' was not transmitted by any of the browsers.

The predictability of the attribute formatting meant that the most appropriate manipulation technique would be the substitution of the HTTP header data, before it was received by the webserver.

To accommodate the variation in the amount of attribute data identified in 'User-Agent', browser specific manipulations were derived for 'User-Agent (7)' and 'User-Agent (9)'. 'DNT (Do Not Track)' was not tested further, as it had not been transmitted by any of the browsers and 'Accept Charset' was only tested with Chrome, as that was the only browser to transmit it.

4.2.2 JavaScript Attribute Analysis

The code used to generate the JavaScript attributes was analysed and a site-by-site synopsis detailed in Appendix Table A-2, reveals that three collection techniques are used:

1. Attributes including: 'Platform (OS)(13)', 'Screen Resolution (18)' were collected from the JavaScript system variables 'navigator' and 'window'.

2. Custom code was used to derive attributes including: 'JavaScript Version (12)', 'Plugins and Versions (20)' and some of the attributes containing fonts.

3. An external Flash library was also used to collect the remaining attributes relating to fonts.

Taking inspiration from Baviskar and Thilagam (2011), the manipulation technique selected to change the attributes collected from JavaScript system variables, was to substitute the variable name in the webpage before the browser received it.

The attributes collected using custom JavaScript code and external Flash libraries used techniques specific to the fingerprinting sites, making it difficult to derive manipulation techniques that could be generalised. So techniques were not identified and the associated attributes were excluded from further testing.

4.2.3 Deriving attribute manipulation tests

Having identified the manipulation techniques, attribute manipulation tests were derived with the aim of altering the browser fingerprint so it would not be recognised.

Unfortunately it was not possible to alter the system variable that controlled the attribute 'Plugins and Versions (20)' in a way to accommodate the detection avoidance rules proposed in Table 3-2, so this attribute was not tested further. The remaining rules were applied to the manipulation techniques and the derived attribute manipulations tests are listed and detailed in Table 4-1. The raw manipulation text is included in Appendix Tables A-3 and A-4.

4.2.4 Analysis of experiments testing impact of manipulations on the operation of the browser

To understand if the manipulation tests impacted the operation of the browser, experiments were designed where the four browsers 'visited' the Internet test sites after the manipulation tests had been applied.

Table 4-4 details an analysis comparing the impact of the manipulations between the browsers and it can be observed that more than 90% of them were successful, and whilst the difference is less than 3%, MSIE and Safari experienced the highest number of failures.

Table 4-4 Analysis of the impact of the manipulation tests on the browsers

Browser	Tests Completed	Page Failed to Display (0)	Page Element Missing (1)	Functionality Impaired (2)	Page Displayed as Expected (3)
MSIE	88	8 (9.1%)			80 (90.9%)
Firefox	88	7 (8.0%)			81 (92.0%)
Chrome	96	7 (7.3%)			89 (92.7%)
Safari	92	9 (9.8%)			83 (90.2%)

Note: All percentages are rounded to one decimal place.

An analysis to understand which sites caused the browser to fail most often, is presented in Table 4-5, showing that the highest failure rate was 18.7% for 'BrowserHawk', followed by 13.2% for 'BBC'. There were no problems performing the manipulations as browsers 'visited' 'YouTube' and 'zeFrank'.

Table 4-5 Analysis of browser manipulation impact tests

Internet Test Site	Tests Completed	Page Failed to Display (0)	Page Element Missing (1)	Functionality Impaired (2)	Page displayed as expected (3)
YouTube	91	0 (0.0%)			91 (100.0%)
BBC	91	12 (13.2%)			79 (86.8%)
BrowserHawk	91	17 (18.7%)			74 (81.3%)
zeFrank	91	0 (0.0%)			91 (100.0%)

Note: The percentages are rounded to one decimal place.

Table 4-6 provides a break down of the manipulations that caused the browser to fail and as already identified, 'BrowserHawk' experienced the most issues, which considering its function is to test browser capabilities, was not a surprise.

Table 4-6 Manipulations that affected operation of the browser

Attribute Tests	Browser	Site that failed to display
User-Agent (6)	MSIE	BrowserHawk
User-Agent (7c)	MSIE	BrowserHawk
User-Agent (7j)	Safari	BrowserHawk
User-Agent (7k)	Safari	BrowserHawk
Platform (OS)(13a)	All	BBC
JavaSupport (17)	Firefox, Chrome and Safari	BrowserHawk
Screen Resolution (18a & b)	All	BrowserHawk and BBC
Screen Resolution (18c)	All	BrowserHawk

An anticipated issue was 'User-Agent (6)', changing the operating system to MacOS for the browser MSIE (which should only be found running from Windows). The 'User-Agent (7)' manipulations, which changed the browser type, produced fewer problems than expected, and 'BrowserHawk' only had problems when MSIE and Safari were swapped, and also when Safari was changed to Firefox. An outcome that was surprising was that 'Screen Resolution (18a & b)' caused 'BBC' to fail.

4.2.5 Analysis of experiments testing impact of manipulations on the fingerprint

To understand whether the attribute manipulations could change a browser fingerprint, experiments were designed where the four browsers 'visited' the fingerprint websites after the manipulations had been applied.

Table 4-7 details an analysis comparing the impact of the manipulations on the browsers, as they 'visited' the fingerprint sites. From the table it can be observed that whilst there are only small differences between the browsers, MSIE had the greatest number of failures, and the fewest successful fingerprint changes. Furthermore there was only a 0.1% difference between the number of manipulations presenting successful fingerprint changes for Firefox, Chrome and Safari.

Table 4-7 Analysis of browser fingerprint tests

Browser	Tests Completed	Page display Failures (0)	Fingerprint not Changed (1)	Fingerprint Changed (2 & 3)	Fingerprint Changed and not recognised (3)
MSIE	85	9 (10.6%)	40 (47.1%)	36 (42.4%)	1 (1.2%)
Firefox	85	7 (8.2%)	36 (42.4%	42 (49.4%)	0 (0.0%)
Chrome	93	6 (6.5%)	41 (44.1%)	46 (49.5%)	0 (0.0%)
Safari	89	3 (3.4%)	42 (47.2%)	44 (49.4%)	1 (1.1%)

Notes: The difference in the number of tests completed is because some were not relevant to all browsers. All percentages are rounded to one decimal place.

Table 4-8 details an analysis comparing the effect of applying the manipulations between the fingerprint sites. The site detecting the smallest number of fingerprint changes was 'PetPortal', which may be explained due to its dependency on high entropy attributes involving fonts, and the Internet address, for which no manipulations were tested.

Table 4-8 Analysis of manipulations tested against fingerprint sites

Fingerprint Site	Tests completed	Page display failures (0)	Fingerprint not changed (1)	Fingerprint changed (2 & 3)	Fingerprint Changed and not recognised (3)*
Panopticlick	87	9 (10.3%)	33 (37.9%)	45 (51.7%)	
LetMeTrackYou	91**	5 (5.5%)	36 (39.6%)	50 (54.9%)	
HowUniqueAreYou	87	4 (4.6%)	35 (40.2%)	48 (55.2%)	2 (2.3%)
PetPortal	87	7 (8.0%)	55 (63.2%)	25 (28.7%)	

*Notes: * HowUniqueAreYou also detects returning fingerprints, ** LetMeTrackYou recognised the HTTP 'Connection:' attribute accounting for the additional four tests. All percentages are rounded to one decimal place.*

Most failures (10.3%) were recorded when the browsers 'visited' 'Panopticlick', which is 2.3% higher than 'PetPortal', which is 2.5% higher than 'LetMeTrackYou' and 'HowUniqueAreYou'. There are no indications from the data collected to explain these percentages, and as the code is private further analysis was not possible.

A feature of the site 'HowUniqueAreYou' was that it calculated a fingerprint identity for 'visiting' browsers, and then attempted to match it against ones it had previously seen. In the event that a match of 85% or more was achieved, it would report that the fingerprint was recognised, the percentage match, what had changed and when it had been previously seen. This information was recorded and

from a review of Table 4-8 it can be observed that in two of the 87 manipulation tests the returning browser was not recognised.

Table 4-9 details the occurrences when the returning fingerprint was recognised with a certainty of less than 90%. It can be observed that 'User-Agent (6)', which changes the operating system to Mac reduced the match to 86%, which is not surprising because MSIE requires Windows to operate. Then for the three manipulations 'User-Agent (7a-c)', involving the browser type MSIE, a post manipulation match of 87% or lower was achieved, furthermore 'User-Agent (7d-e)', which are manipulations involving the browser Firefox, consistently reduced the percentage match to 87%.

Table 4-9 Returning browser recognition results

Attribute Manipulation Tests	Browser	Percentage Match
Accept (1)	Firefox	86%
Accept-Language (2)	MSIE	86%
User-Agent (6)	MSIE	87%
User-Agent (7a)	MSIE	<85%
User-Agent (7b)	MSIE	86%
User-Agent (7c)	MSIE	87%
User-Agent (7d)	Firefox	87%
User-Agent (7e)	Firefox	87%
User-Agent (7f)	Firefox	87%
User-Agent (8)	Firefox	86%
User-Agent (8)	Safari	<85%
User-Agent (9f)	Safari	88%
Platform (OS)(13b)	Firefox	86%
Timezone (19a)	MSIE	87%
	Firefox	85%

Note: '<85%' signifies that the site did not recognise the browser as returning

In the case of 'User-Agent (8)' in addition to Safari not being recognised as a returning browser, Firefox returned a certainty of only 86%. From the discussion in 4.2.1, comparisons had been drawn between the number of 'User-Agent' attribute elements for Chrome and Safari, so it was surprising that this connection had not manifested in these results.

Another finding that was surprising was that whilst the manipulations 'Accept (1)', 'Accept-Language (2)' and 'Platform (OS)(13b)' are all related to low entropy attributes, they produced a sufficient change so that a match of only 86% was achieved for some browsers. The attribute manipulation tests were designed to focus on changing individual elements of an attribute, which may offer an explanation to understand why the cross section of low, medium and high entropy attributes appeared in Table 4-9.

4.2.6 Experiments testing the impact of multiple manipulations on fingerprint effectiveness

Additional experiments were considered to understand how manipulating combinations of attributes impacted the effectiveness of the fingerprint. However, these were discounted due to the binary nature of the low and medium attribute manipulation tests and that only one high entropy attribute test had been derived. Instead, because the high entropy attribute 'User-Agent' had between four (MSIE) and fourteen (Safari and Chrome) variable elements that could be changed, experiments were designed to test the hypothesis that increasing the number of element manipulations could reduce the effectiveness of the fingerprint for tracking.

To add a further level of certainty and ensure the findings would be reproducible, repeat 'visits' were attempted for each set of manipulations that were tested. The full findings are presented in Appendix Table A-6, and a summary is presented in Table 4-10. A problem was that 'User-Agent (6)' and all permutations of 'User-Agent (7a-l)' had already been tested and would have been recognised by the

site 'HowUniqueAreYou', meaning that they could not be used. The consequence of this was that the number of attribute elements available for use was reduced (MSIE only had two elements). To overcome this limitation different browsers were used and to provide validity both Firefox and Safari were tested with four attribute element changes, where it can be observed that the percentage certainty in both cases was 88%.

Table 4-10 Repeated browser returning recognition results

Test	Attributes Manipulations Tested	Browser	Elements changed	Number of repeat visits	Percentage Match Certainty
1	User-Agent (8) + User-Agent (9a)	MSIE	2	5	95%
2	User-Agent (8) + User-Agent (9b)	Firefox	3	5	90%
3	User-Agent (8) + User-Agent (9b)	Firefox	4	5	88%
4	User-Agent (8) + User-Agent (9d) + User-Agent (9f)	Safari	4	5	88%
5	User-Agent (8) + User-Agent (9c) + User-Agent (9e)	Chrome	5	5	86%
6	User-Agent (8) + User-Agent (9c) + User-Agent (9e)	Chrome	6	5	<85%

The findings show, that the more attribute elements that are altered the less likely the fingerprint is to be recognised, and that the percentage match drops below 85% once six elements have been changed. However whilst the values are consistent for a specific number of elements altered, Appendix Table A-6 shows that on two occasions (Tests 1 and 4) the very first iteration produced a different percentage. Test 1 first reported a percentage match certainty of 86%, before reverting to 95%, and Test 4 reported 87% before reverting to 86%. These findings are not repeated in any of the other tests so no explanation can be deduced.

Figure 4-1 shows a plot of the data in Appendix Table A-6, for Tests 1 to 5, with a trend line indicating that at six elements a certainty of less that 85% should occur, supporting the findings in Table 4-10 for Test 6.

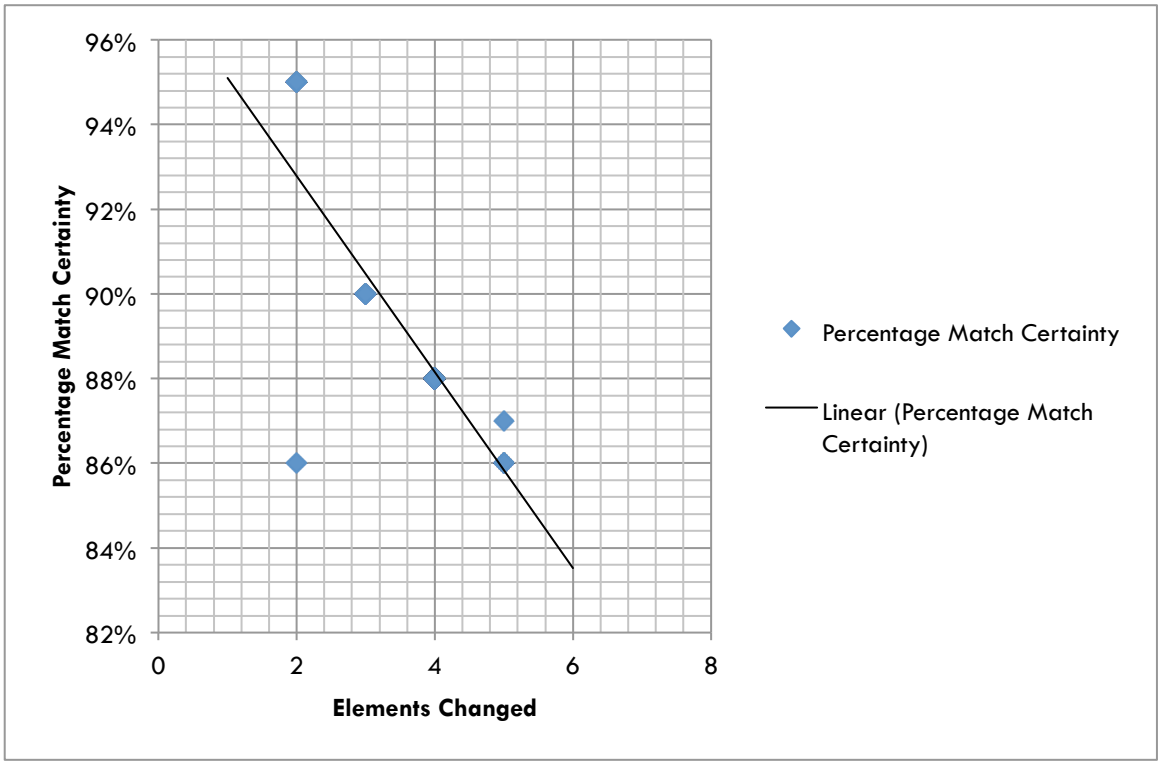

Figure 4-1 Percentage match certainty plotted against elements changed

4.3 Interpretation in relation to the research questions

Having analysed the data that was collected, this section focuses on how it can be interpreted to answer the research questions.

4.3.1 What techniques are available for manipulating the browser attributes?

Raw data from nine HTTP header and fourteen JavaScript attributes, was generated using 'in-situ' experiments with the browsers 'visiting' the fingerprint websites.

The data was analysed and two manipulation techniques were identified. One technique was to manipulate the HTTP header attributes, by altering the content of the HTTP header before it was received by the webserver. The second was to manipulate eight out of fourteen (57.1%) of the JavaScript attributes, by changing the JavaScript code associated with the attribute before being received by the browser. Techniques could not be identified for the remaining six (42.9%) JavaScript attributes because the collection code identified was specific to the fingerprint site, making it difficult to generalise.

Having designed the manipulation techniques, the detection avoidance rules proposed in Table 3-2 were incorporated to derive manipulation tests with the capability to alter the fingerprint and reducing its effectiveness for tracking. These manipulation tests are listed and detailed Table 4-1.

4.3.2 Can these techniques be used to manipulate the attributes without affecting the operation of the browser?

Experiments were conducted to understand if the manipulation techniques would impact the operation of the browser, and 364 tests were performed. Each browser was tested and from the comparison presented in Table 4-4 it can be observed that less that 10% of the manipulations caused the browsers to fail. The browser that had the most failures was Safari, at 9.8%, followed by MSIE at 9.1%, Firefox at 8.0% and Chrome at 7.3%.

At 18.7%, 'BrowserHawk' had the largest number of problems with the manipulations, which was not a surprise as its function is to test the capabilities of the browser. An example provided in Table 4-6 that illustrates this, was that 'BrowserHawk' failed when the operating system was changed from Windows to Mac OS for the browser MSIE, which is a confusing scenario considering that MSIE requires windows to operate.

At 13.2%, 'BBC' had the second largest number of problems with the manipulations, which is considered significant, as it is one of the most popular sites in the UK, and was not designed with browser testing in mind.

Another observation is that, 'BrowserHawk' failed for the same number of HTTP and JavaScript attributes, whereas 'BBC' did not have problems with any of the HTTP header manipulations. This finding suggests problems could exist with the techniques selected for manipulating JavaScript attributes.

4.3.3 Which attribute variable manipulation techniques can be used to alter a browser fingerprint?

Experiments were conducted to understand if the manipulation techniques could impact the browser fingerprint, and 352 tests were performed. Each browser was tested and from the comparison presented in Table 4-7, the fingerprint was changed in over 40% of tests.

Of the fingerprint sites, most reported a success at changing the fingerprint in over 50% of the tests, whereas 'PetPortal' reported only 28.7%. This finding has already been discussed and a suggestion is

that this is caused because manipulations could not be identified to change the high entropy attributes it used to calculate a fingerprint.

Table 4-11 presents an analysis to understand if some attribute manipulations are more successful than others at changing the browser fingerprint. From the table it can be observed that the least successful manipulation was 'Platform (OS)(13a)' at 6.3%, which only managed to influence the fingerprint site 'HowUniqueAreYou' once, and with the exception of attribute 'UserAgent (6)', all the remaining HTTP header attribute manipulations achieved, 68.8% success or greater.

Table 4-11 Effectiveness of manipulation of changing fingerprint

Attribute Tests	Tests Completed	Fingerprint Change Observed (2 or 3)	Effectiveness of Manipulation for Changing the Fingerprint	Effectiveness assuming 16 tests were undertaken
Accept (1)	16	12	75.0%	75.0%
Accept-Language (2)	16	11	68.8%	68.8%
Accept-Encoding (3)	16	12	75.0%	75.0%
Accept-Charset (4)	4	3	75.0%	**18.8%**
Connection (5)	4	4	100.0%	**25%**
User-Agent (6)	16	7	43.8%	43.8%
User-Agent (7a-l)*	48	48	100.0%	Not applicable
User-Agent (8)	16	16	100.0%	100.0%
User-Agent (9a-f)*	24	18	75.0%	Not applicable
Platform (OS)(13a)	16	1	6.3%	6.3%
Platform (OS)(13b)	16	3	18.8%	18.8%
Screen Resolution (18a)	16	10	62.5%	62.5%
Screen Resolution (18b)	16	10	62.5%	62.5%
Screen Resolution (18c)	16	8	50.0%	50.0%
Timezone (19)	16	5	31.3%	31.3%

Notes: Attributes are only listed if the percentage is >0%, all percentages are rounded to one decimal place. User-Agent(7) & User-Agent(9) are combined as they are multiple aspects of the same manipulation test.

The purpose of this analysis was to understand whether attribute manipulations were effective at altering the browser fingerprint, and a validity issue was identified because not all permutations were tested. To overcome this, a fifth column was added to Table 4-11, giving the predicted outcome if all attribute manipulation permutations had been tested. In this case it can be observed that 'Accept-Charset (4)' and 'Connection (5)' are unlikely to be as effective as had previously been suggested, with scores dropping from 75% and 100% to 18.8% and 25% respectively.

4.3.4 Can the manipulation techniques make the browser fingerprint less effective?

To understand whether the manipulation techniques can make the fingerprint less effective, a feature of the site 'HowUniqueAreYou', to recognise returning browsers, was used. A fingerprint is calculated from a 'visiting' browser, and then compared to ones from previous visits. If a match of 85% or greater is achieved the browser is reported as being recognised. Of 87 tests, the fingerprint was recognised on 85 occasions, and the percentages are recorded in Appendix Table A-5. An analysis was conducted to review matches of 90% or lower (see Table 4-9) and whilst it was observed that ten out of the fourteen (71.4%) occurrences were related to the high entropy attribute 'User-Agent', there were no obvious patterns.

One explanation for the lack of patterns was that the attribute manipulation tests had been designed to operate on a single element. So additional experiments were conducted, to test the hypothesis that the more attribute elements that are changed, the less effective the resultant fingerprint. A problem with these experiments was that insufficient elements were available for testing all browsers, suggesting that these results could only be indicative. One notable observation that can be made, by

comparing the percentage match for 'User-Agent (7 and 8) on Firefox (from Table 4-9), and Tests 2 and 3, (from Table 4-10), is that the manipulation of some elements of the attribute appeared to impact the fingerprint more than for others.

4.4 Interpretation in relation to the research aim

The aim of this research was to investigate the extent to which it may be possible to reduce the effectiveness of tracking using a browser fingerprint.

From the literature review, twenty-five attributes were identified and attribute manipulation tests were derived for seventeen (68.0%). These tests were first used to understand if the manipulations could be used without effecting the operation of the browser, and findings illustrated 18.7% was the highest failure rate. They were then tested to understand if the browser fingerprint could be altered, reducing its effectiveness for tracking. A success of 51% was observed for three out of four fingerprint sites, the other had a 28.7% success. This is explained, as the sites are likely to use different attributes to construct the fingerprints.

A feature of one site, 'HowUniqueAreYou' was that it could report whether or not it recognised a returning browser, and in fifteen out of 87 (17.2%) tests the certainty of a match was less than 90%, reducing to two (2.3%) for a certainty of less than 85%. No obvious patterns were identified, so manipulation tests were designed and conducted, using multiple aspects of high entropy attributes. Repeat visits indicated that certainty decreased consistently from 95% with two element changes, down to less than 85% when six elements were changed. Unfortunately a lack of available aspects meant that MSIE and Firefox could not be tested for all occurrences making these findings inconclusive.

Chapter 5 Conclusions

5.1 Conclusions about the research questions

This section provides a discussion of conclusions with regard to the research questions:

5.1.1 What techniques are available for manipulating the browser attributes?

Two attribute manipulation techniques were identified and tested using thirty-seven attribute manipulations (see Table 4-1). The first technique was to replace the attribute data associated with the HTTP header before it is received by the webserver, and the second was to replace JavaScript code containing system variables that are used to collect the attribute.

Some JavaScript attributes could not be tested using the identified technique, because either custom code was found that could not be generalised or because external Java or Flash modules had been used.

5.1.2 Can these techniques be used to manipulate the attributes without affecting the operation of the browser?

In laboratory experiments, 90% of the 364 fingerprint manipulation tests, did not effect the operation of the browser. Four Internet sites were used for testing and issues with the manipulations only occurred for two. The site with the highest failure rate at 18.7%, 'BrowserHawk', tests browser functionality and had issues with attributes that are altered using both types of manipulation techniques. Compared to 'BBC', one of the most popular websites in the UK, at 13.2%, which only experienced problems with the JavaScript manipulation techniques.

One conclusion that can be drawn from experiments is that: for popular Internet sites, the HTTP header manipulation techniques may be used without effecting the operation of the browser, and that problems are likely to be experienced using the techniques to manipulate JavaScript attributes.

5.1.3 Which attribute variable manipulation techniques can be used to alter a browser fingerprint?

Laboratory experiments, involving 352 tests, were conducted incorporating both manipulation techniques, and for three out of four fingerprint sites, over 50% of them altered the fingerprint. The exception 'PetPortal' reported only 28.7% success, which was explained because the selection of attribute manipulations was less relevant to this fingerprint site.

To put these findings into context, it must be considered that of the twenty-five attributes identified from literature, manipulation tests were only derived for seventeen (68.0%) of them, and of which only twelve (48.0%) had any success at changing the fingerprint (See Table 4-11). This demonstrates that there is no simple answer to this question, and that each fingerprinting site is likely to have its own attribute selection criteria for generating a fingerprint. However, it has been possible to demonstrate that the browser fingerprint can be altered using both manipulation techniques.

5.1.4 Can the manipulation techniques make the browser fingerprint less effective?

The final research question was designed to understand whether the manipulation techniques could be used to alter the fingerprint, making it less effective for tracking. The avoidance techniques detailed in Table 3-2 were applied to derive manipulation tests and data was collected as browsers 'visited' the fingerprint site 'HowUniqueAreYou'. Eighty-seven manipulation tests were conducted and on fourteen (16.1%) occasions the fingerprint was altered sufficiently to reduce the certainty of a match to below 90% (see Table 4-9 for details), and on two (2.3%) occasions it was not recognised at all, giving a strong indication that these techniques, if developed, have the capability to reduce the effectiveness of the fingerprint for tracking.

An unexpected outcome was that no patterns were observed, in particular the results showed no advantage to manipulating high entropy attributes. To understand if this was because the attribute manipulation tests only altered individual elements, further experiments were designed to test the hypothesis, that making multiple changes to an attribute could improve the impact on the fingerprint.

Additional experiments were conducted using the high entropy attribute 'User-Agent'. The findings demonstrated that the more elements changed the less recognisable the fingerprint, where at six elements it reduced to less than 85%. Unfortunately whilst these findings indicated that increasing the number of elements reduced the effectiveness of the fingerprint, a lack of available attribute elements meant that not all browser permutations could be tested, making these findings inconclusive. Furthermore this experiment required that the versions of operating system and browser be reduced, to avoid being detected as a returning browser. Whilst this operation appeared to defeat the fingerprinting site, it is not something that could be sustained without a complex strategy, if at all.

5.2 Conclusions about the research aim

The aim of this research was to investigate the extent to which it may be possible to reduce the effectiveness of tracking using a browser fingerprint.

From secondary research fingerprint attributes suitable for tracking were identified, with a weighting to indicate how much they were likely to contribute to its effectiveness for tracking. Furthermore it had been observed that attributes changed over time, as users updated their computers. With the aim of making changes that were possible but unlikely, to avoid being recognised, two manipulation techniques were identified, aligned to HTTP header and JavaScript attributes.

Experiments were conducted, to understand if they could reduce the effectiveness of the fingerprint for tracking, without impacting the operation of the browser. The findings indicated that, the HTTP header attributes were less prone to causing an impact to the operation of the browser, and that in 13.8% of tests the fingerprint was altered sufficiently to reduce the certainty of a match to less than 90%. More issues were observed manipulating JavaScript attributes, and of the fourteen JavaScript attributes identified, techniques could only be derived for eight (57.1%). The browser also experienced an impact in 13.2% of tests and on only two (2.3%) occasions did the certainty of the fingerprint match reduce to so the fingerprint was not recognised. A lack of expected patterns in the findings, led to additional experiments being conducted to understand if increasing the number of attribute elements could reduce the certainty match predictably. Unfortunately, whilst it was possible to demonstrate that the fingerprint was not recognised when six elements were manipulated at the same time, the findings were inconclusive because not all browsers had sufficient elements available for testing.

The findings support the conjecture that altering the fingerprint attributes in a way that was legitimate but unexpected, is likely to reduce the effectiveness of a fingerprinting site recognising a returning browser. However it is important to consider that this approach would be complex to manage because attribute elements need to be changed to appropriate values each time a fingerprinting site is 'visited'.

5.3 Further Work

This research has demonstrated success in reducing the fingerprints effectiveness using two manipulation techniques that were derived from a restricted set of attributes and browsers. This indicates that increasing the scope could add depth to the understanding, and potentially increase the number of viable manipulation techniques.

The analysis in 4.2.1 showed that the attribute 'User-Agent' from MSIE had much less entropy than from the other browsers. This was previously unreported and indicates that changes are occurring in the browser evolution that could make some attributes less effective for tracking. Further research should

be conducted to understand this phenomenon, and potentially looking for opportunities to reduce the entropy of other attributes without impacting the operation of the browser.

5.4 Implications of the research

Prior to this research the focus had been on techniques to generate a fingerprint that could be used for tracking a computer or browser, and harbouring the premise that a unique fingerprint was a bad thing. This research has focused on investigating techniques that may be used to make the fingerprint less effective for tracking, by taking advantage of the natural changes a computer may encounter over time, to change the fingerprint and avoid being tracked.

5.5 Reflections on the experience of the research process

The research process has been a rewarding and frustrating experience requiring the leap from being a designer with the aim of delivering a specific solution to address a problem, over to a researcher who is looking to understand more about a phenomenon.

During the process, skills have been learned in a number of disciplines:

Critical thinking and not taking things on face value, which sounds easy, but to ensure that there is substance behind what is being said or described is difficult.

Articulating an idea, and making a point that can be clearly understood without the inclusion of colloquial terms is harder than it appears, but needs mastering if these ideas are to be propagated effectively.

Dissertation word count: 14885

References

AdTruth (2011) *SteelHouse Turns to AdTruth for High Performance, Pro-Privacy Cookie Augmentation* [online], SCOTTSDALE, Ariz, PRNewswire. http://www.prnewswire.com/news-releases/steelhouse-turns-to-adtruth-for-high-performance-pro-privacy-cookie-augmentation-133431228.html (Accessed 11th September 2012).

Aggarwal, G., Bursztein, E., Jackson, C. and Boneh, D. (2010) 'An analysis of private browsing modes in modern browsers', *19th Usenix Security Symposium 2010,* 11th August 2010, Washington DC, USA, USENIX Association. pp. 1-15.

Alexa (2012) *Statistics Summary for bbc.co.uk* [online], San Franciso, US, Alexa Internet, Inc. http://www.alexa.com/siteinfo/bbc.co.uk (Accessed 10th November 2012).

Angwin, J. (2010) *The Web's New Gold Mine: Your Secrets* Wall Street Journal. http://online.wsj.com/article/SB10001424052748703940904575395073512989404.html (Accessed 4th March 2012).

Angwin, J. and Valentino-Devries, J. (2010) *Race Is On to 'Fingerprint' Phones, PCs* [online], European Edition, The Wall Street Journal. http://online.wsj.com/article/SB10001424052748704679204575646704100959546.html (Accessed 10th June 2012).

Apple (2012) *Apple Product Security* [online], Cupertino, CA USA, https://ssl.apple.com/support/security/ (Accessed 19th December 2012).

Baviskar, S. and Thilagam, P. S. (2011) 'Protection of Web User's Privacy by Securing Browser from Web Privacy Attacks', *International Journal of Computer Applications in Technology,* Vol 2(4), No 1, pp. 1051-1057.

Boda, K., Földes, Á., Gulyás, G. and Imre, S. (2011) 'User Tracking on the Web via Cross-Browser Fingerprinting', *NordSec'11 Proceedings of the 16th Nordic conference on Information Security Technology for Applications,* Lecture Notes in Computer Science, Springer-Verlag Berlin, Heidelberg. pp. 31-46.

Bohi, H. (2010) 'BEHAVIORAL TARGET MARKETING: Virtual stalker or virtual coach?', *Alaska Business Monthly,* Vol 26, No 2, pp. 10-14.

Broenink, R. (2012) 'Using Browser Properties for Fingerprinting Purposes', *16th biannual Twente Student Conference on IT,* 27th January 2012, Enschede, The Netherlands, Twente University Press. pp. 169-176.

Dingledine, R., Mathewson, N. and Syverson, P. (2004) 'Tor: The second-generation onion router', *13th USENIX Security Symposium,* August 9-13, 2004, San Diego, CA USA, USENIX Association. pp. 303-320.

eBizMBA (2012) *Top Most Popular Video Websites | November 2012* [online], New York, USA, eBizMBA Inc. http://www.ebizmba.com/articles/video-websites (Accessed 10th November 2012).

Eckersley, P. (2010) 'How Unique Is Your Web Browser?', *LePETS'10 Proceedings of the 10th international conference on Privacy enhancing technologies,* Lecture Notes in Computer Science, Berlin, Germany, Springer Berlin / Heidelberg. pp. 1-18.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T. (1999) *RFC2616: Hypertext transfer protocol–HTTP/1.1* [online], Fremont, California, USA, The Internet Society. http://tools.ietf.org/html/rfc2616 (Accessed 29th March 2012).

Giles, J. (2011) 'The Real You', *New Scientist,* Vol 212, No 2836, pp. 50-53.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I. H. (2009) 'The WEKA data mining software: an update', *SIGKDD Explor. Newsl.,* Vol 11, No 1, pp. 10-18.

Information Commissioner's Office (2012) *Enforcing the revised Privacy and Electronic Communications Regulations* http://www.ico.gov.uk/for_organisations/privacy_and_electronic_communications/~/media/documents/library/Privacy_and_electronic/Practical_application/enforcing_the_revised_privacy_and_electronic_communication_regulations_v1.pdf (Accessed 10th January 2013).

Isaac, M. (2012) *TED 2012: New Browser Add-On Visualizes Who Is Tracking You Online* [online], San Francisco, USA, WIRED. http://www.wired.com/epicenter/2012/02/ted-mozilla-collusion/ (Accessed 28th February 2012).

Keil, F., Schmidt, D., Burgiss, H., Rian, L. and Rosenfeld, R. (2012) *Privoxy - Home Page* [online], Boston USA, Privoxy Developers. http://www.privoxy.org/ (Accessed 6th November 2012).

Keith, J. and Sambells, J. (2010) *DOM scripting: web design with JavaScript and the Document Object Model*, friends of ED.

Madrigal, A. (2012) *I'm Being Followed: How Google and 104 Other Companies Are Tracking Me on the Web* http://www.theatlantic.com/technology/archive/2012/02/im-being-followed-how-google-151-and-104-other-companies-151-are-tracking-me-on-the-web/253758/ (Accessed 8th June 2012).

Mayer, J. R. (2009) *Any person... a pamphleteer": Internet Anonymity in the Age of Web 2.0, Faculty of the Woodrow Wilson School of Public and International Affairs,* Senior Thesis, Princeton University.

Mowery, K., Bogenreif, D., Yilek, S. and Shacham, H. (2011) 'Fingerprinting information in javascript implementations', *IEEE Symposium on Security and Privacy,* May 26th 2011, Oakland, California, USA, pp. 1-11.

Mozilla (2012) *Security Centre* [online], Mountain View, CA USA, Mozilla. http://www.mozilla.org/security/ (Accessed 19th December 2012).

Open University T802 (2010) *Methodology and Techniques*, Milton Keynes, Open University.

Perry, M., Clark, E. and Murdoch, S. (2011) *The Design and Implementation of the Tor Browser [DRAFT]* [online], United States, http://www.torproject.org/projects/torbrowser/design/ (Accessed 13th June 2012).

Purcell, K., Brenner, J. and Rainie, L. (2012) *Search Engine Use 2012* [online], Washington, DC USA, Pew Research Center's Internet & American Life Project. http://www.pewinternet.org/Reports/2012/Search-Engine-Use-2012.aspx (Accessed 28th September 2012).

Said, H., Al Mutawa, N., Al Awadhi, I. and Guimaraes, M. (2011) 'Forensic analysis of private browsing artifacts', *Innovations in Information Technology (IIT), 2011 International Conference on,* 25-27 April 2011, pp. 197-202.

Schmücker, N. (2011) *Web Tracking, Department of Telecommunication Systems,* SNET2 Seminar Paper - Summer 2011, Telekom Innovation Laboratories TEL 19, Berlin, Berlin University of Technology.

Shannon, C. (1948) 'A Mathematical Theory of Communication', *The Bell System Technical Journal,* Vol 27, No 1, pp. 379-423.

Sipior, J. C., Ward, B. T. and Mendoza, R. A. (2011) 'Online Privacy Concerns Associated with Cookies, Flash Cookies, and Web Beacons', *Journal of Internet Commerce,* Vol 10, No 1, pp. 1-16.

Smith, S. (2006) *Review: BrowserHawk 9* [online], US, ASP Alliance. 10th November 2012).

StatCounter (2012a) *Top 5 Browsers from Sep 2011 to Sep 2012* [online], Dublin, IE, http://gs.statcounter.com/ - browser-ww-monthly-201109-201209 (Accessed 29th Oct 2012).

StatCounter (2012b) *Top 5 Operating Systems from Nov 2011 to Nov 2012* [online], Dublin, IE, http://gs.statcounter.com/ - os-ww-monthly-201111-201211 (Accessed 19th December 2012).

Stocks, M. (2012) *Device Fingerprinting System, Informatics - Computer Science,* Dissertation, University of Sussex.

The Tor Project, I. (2012) *Tor: Overview* [online], http://www.torproject.org/, The Tor Project, Inc. http://www.torproject.org/about/overview.html.en (Accessed 10th September 2012).

Touchton, C. (2012) *The Best Interactive Websites And Fun Things To Do Online When Bored* [online], New York, USA, Squidoo. http://www.squidoo.com/interactive-websites (Accessed 10th November 2012).

Tsukayama, H. (2012) *Facebook IPO: How does Facebook make its money?* http://www.washingtonpost.com/business/technology/facebook-ipo-how-does-facebook-make-its-money/2012/02/01/gIQAL03yiQ_story.html (Accessed 2nd March 2012).

Vander Veer, E. A. (2004) *JavaScript for dummies, 4th Edn*, Indianapolis, Indiana, Wiley Publishing Inc.

Wikipedia (2012) *JavaScript* [online], http://www.wikipedia.com, http://www.wikipedia.com/wiki/JavaScript (Accessed 22th March 2012).

Xie, Y., Yu, F. and Abadi, M. (2009) 'De-anonymizing the internet using unreliable ids', *SIGCOMM '09 Proceedings of the ACM SIGCOMM 2009 conference on Data communication,* New York, NY, USA, ACM. pp. 75-86.

Yen, T.-F., Huang, X., Monrose, F. and Reiter, M. (2009) In *Conference on Detection of Intrusions and Malware & Vulnerability Assessment, 2009*, Vol. 5587 (eds) Flegel, U. and Bruschi, D. Springer Berlin / Heidelberg, pp. 157-175.

Yen, T. F., Xie, Y., Yu, F., Yu, R. P. and Abadi, M. (2012) In *Network and Distributed System Security Symposium 2012*, Vol. San Diego, California, CA United States.

Appendices

Table A- 1 Summary of raw HTTP Header attribute data

Header	**MSIE Values**
Accept:	text/html, */* text/html, application/xhtml+xml, */*
Accept-Language:	en-us en-US
User-Agent:	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)
Accept-Encoding:	gzip, deflate
Connection:	Keep-Alive
Header	**Firefox Values**
User-Agent:	Mozilla/5.0 (Windows NT 6.1; rv:16.0) Gecko/20100101 Firefox/16.0
Accept:	text/html, */* text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language:	en-US,en;q=0.5
Accept-Encoding:	gzip, deflate
Connection:	keep-alive
Header	**Chrome Values**
Connection:	keep-alive
User-Agent:	Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.11 (KHTML, like Gecko) Chrome/23.0.1271.64 Safari/537.11
Accept:	text/html, */* text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding:	gzip,deflate,sdch
Accept-Language:	en-US,en;q=0.8
Accept-Charset:	ISO-8859-1,utf-8;q=0.7,*;q=0.3
Safari Header	**Safari Values**
User-Agent:	Mozilla/5.0 (Windows NT 6.1) AppleWebKit/534.57.2 (KHTML, like Gecko) Version/5.1.7 Safari/534.57.2
Accept:	text/html, */* text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language:	en-US
Accept-Encoding:	gzip, deflate
Connection:	keep-alive

Table A- 2 Summary of raw JavaScript attribute data

Attribute	**JavaScript code used to extract attribute data for Panopticlick**
Plugins and Versions	A custom JavaScript library PluginDetect.js is used. MSIE does manual detection for Quicktime, DevalVR, Shockware, Flash, WindowsMediaplay and Silverlight, other browsers use **navigator.plugins** to get the list of plugins.
Timezone	JavaScript function **Date().getTimezoneOffset()**
Screen Resolution	JavaScript variables: **screen.width, screen.height** and **screen.colorDepth**
Fonts List (All)	Shockwave Flash library used to collect fonts
Partial Supercookie	Tests use of **localStorage** and **sessionStorage** global objects and userdata class for MSIE
Cookies Enabled	- Unknown -
Attribute	**JavaScript code used to extract attribute data from LetMeTrackYou**
JavaScript Enabled	Set statically set to 1 from the JavaScript code
JavaScript Version	Multiple version dependant JavaScript language code is run, each setting a variable to be the version is relates to. The result is a variable set to the highest version available from the browser.
Platform (OS)	JavaScript variable: **navigator.platform**
Charset	JavaScript variable: **document.defaultCharset**
Language	JavaScript variable: **navigator.language**
Screen Resolution	JavaScript variables: **window.screen.width, window.screen.height** and **window.screen.colorDepth**
Cookies Enabled	JavaScript variable: **navigator.cookieEnabled**
Java Support	JavaScript variable: **navigator.javaEnabled()**
Timezone	JavaScript function **Date().getTimezoneOffset()**
Plugins and Versions	A custom JavaScript library PluginDetect.js is used. MSIE does manual detection for Quicktime, DevalVR, Shockware, Flash, WindowsMediaplay and Silverlight, other browsers use **navigator.plugins** to get the list of plugins.
Fonts List (All)	Shockwave Flash library used to collect fonts
Attribute	**JavaScript code used to extract attribute data from HowUniqueAreYou**
Plugins and Versions	A custom JavaScript library PluginDetect.js which uses **navigator.plugins**
Fonts List (All)	Shockwave Flash library used to collect fonts
Screen Resolution	JavaScript system variables: **window.screen.width, window.screen.height** and **window.screen.colorDepth**
Timezone	JavaScript date variable, method: mydate.**getTimezoneOffset()**
Attribute	**JavaScript code used to extract attribute data from PetPortal**
Platform (OS)	JavaScript variables: **navigator.appVersion**
Screen Resolution	JavaScript variables: **screen.width** and **screen.height**
Timezone	JavaScript function **Date().getTimezoneOffset()**
Detected Fonts	JavaScript code processes a list of fonts and tests to see if they are installed.
User-ID	- Unknown -

Table A- 3 Raw HTTP Attribute Manipulation Tests

Attribute Tests
Accept (1)
Replace 'Accept: */*' with 'Accept: text/html, */*' or 'text/html,' with 'text/html, text/xml,'
Accept-Language (2)
Replace 'en-US' with 'en-GB'
Accept-Encoding (3)
Replace 'gzip' or 'gzip,' with ''
Accept-Charset (4)
Replace 'ISO-8859-1' with 'ISO-8859-2'
Connection (5)
Swap 'keep-alive' and 'Keep-Alive'
User Agent (6)
Replace 'Windows NT 6.1' with 'Macintosh; Intel Mac OS X 10_8_2'
User Agent (7)
Replace MSIE 9.0 with 'Firefox/16.0'
Replace MSIE 9.0 with 'Chrome/23.0'
Replace MSIE 9.0 with 'Version/5.1.7'
Replace 'Firefox/16.0' with 'MSIE 9.0'
Replace 'Firefox/16.0' with 'Chrome/23.0'
Replace 'Firefox/16.0' with 'Version/5.1.7'
Replace 'Chrome/23.0' with 'MSIE 9.0'
Replace 'Chrome/23.0' with 'Firefox/16.0'
Replace 'Chrome/23.0' with 'Version/5.1.7'
Replace 'Version/5.1.7' with 'MSIE 9.0'
Replace 'Version/5.1.7' with 'Firefox/16.0'
Replace 'Version/5.1.7' with 'Chrome/23.0'
User Agent (8)
Replace 'Windows NT 6.1 ' for 'Window NT 6.0'
User Agent (9)
Replace 'MSIE 9.0 ' for 'MSIE 8.0'
Replace 'Firefox/16.0 ' for 'Firefox/15.0'
Replace 'Chrome/23.0 ' for 'Chrome/22.0'
Replace 'Version/5.1.7 ' for 'Version/5.1.6'
Replace '/537. ' for '/536.'
Replace '/534.57.2 ' for '/534.56.1'

Table A- 4 Raw JavaScript Manipulation Tests

Attribute Tests
Platform (OS) (13)
Replace navigator.platform with 'MacIntel'
Replace navigator.appVersion with '5.0 (Macintosh)
Charset (14)
Replace document.defaultCharset with 'ISO-8859-1'
Language (15)
Replace navigator.language with 'en-US
Cookies Enabled (16)
Replace navigator.cookieEnabled with null
Java Support (17)
Replace navigator.javaEnabled() with 'null'
Screen Resolution (18)
Replace window.screen.height or screen.height with '780'
Replace window.screen.width or screen.width with '1024'
Replace window.screen.colorDepth or screen.colorDepth with '32
Timezone (19)
Replace Date().getTimezoneOffset() with '55' or Replace .getTimezoneOffset() with .getUTCMilliseconds()
Plugins and Versions (20)
Replace navigator.plugins with 'null'
Replace navigator.plugins with sessionStorage
Partial Supercookie (23)
Replace localStorage. with notlocalStor.
Replace sessionStorage. with notSessionStor.

Table A- 5 Returning browser data from JavaScript attribute manipulations from 'HowUniqueAreYou'

Attribute Tests	Percentage Match for the browsers			
	MSIE	Firefox	Chrome	Safari
Accept (1)	97%	86%	98%	98%
Accept-Language (2)	86%	100%	98%	98%
Accept-Encoding (3)				
Accept-Charset (4)			98%	
Connection (5)				
User-Agent (6)	87%	100%	100%	98%
User-Agent (7a)	<85%			
User-Agent (7b)	86%			
User-Agent (7c)	87%			
User-Agent (7d)		87%		
User-Agent (7e)		87%		
User-Agent (7f)		87%		
User-Agent (7g)			100%	
User-Agent (7h)			93%	
User-Agent (7i)			93%	
User-Agent (7j)				100%
User-Agent (7k)				95%
User-Agent (7l)				95%
User-Agent (8)	86%	98%	98%	<85%
User-Agent (9a)	90%			
User-Agent (9b)		96%		
User-Agent (9c)			93%	
User-Agent (9d)				100%
User-Agent (9e)			92%	
User-Agent (9f)				88%
Platform (OS) (13a)	100%	100%	100%	100%
Platform (OS) (13b)	100%	86%	100%	100%
Charset (14)	100%	100%	100%	100%
Language (15)	100%	100%	100%	100%
Cookies Enabled (16)	100%	100%	100%	100%
Java Support (17)	100%	100%	100%	100%
Screen Resolution (18a)	93%	94%	100%	94%
Screen Resolution (18b)	93%	92%	92%	93%
Screen Resolution (18c)	93%	92%	94%	92%
Timezone (19)	87%	85%	99%	99%

Table A- 6 Results from multiple element manipulation experiments using 'HowUniqueAreYou'

Test 1	**User-Agent (8) + User-Agent (9a) – two elements Swapping: MSIE 9.0, Win 6.1 for:**	**Iteration**	**% Match**
	MSIE 8.9, Win6.0	1	86%
	MSIE 8.8, Win5.9	2	95%
	MSIE 8.7, Win5.8	3	95%
	MSIE 8.6, Win5.7	4	95%
	MSIE 8.5, Win5.6	5	95%
Test 2	**User-Agent (8) + User-Agent (9b) – three elements Swapping: Firefox/16.0/rv:16.0/, Win 6.1 for:**	**Iteration**	**% Match**
	Firefox/15.9/rv:15.9 Win 6.0	1	90%
	Firefox/15.8/rv:15.8, Win 5.9	2	90%
	Firefox/15.7/rv:15.7, Win 5.8	3	90%
	Firefox/15.6/rv:15.6, Win 5.7	4	90%
	Firefox/15.5/rv:15.5, Win 5.6	5	90%
Test 3	**User-Agent (8) + User-Agent (9b) – four elements Swapping: Firefox/16.0/rv:16.0/Gecko/20100101, Win 6.1 for:**	**Iteration**	**% Match**
	Firefox/15.9/rv:15.9/Gecko/20100100, Win 6.0	1	88%
	Firefox/15.8/rv:15.8/Gecko/20100011, Win 5.9	2	88%
	Firefox/15.7/rv:15.7/Gecko/20100010, Win 5.8	3	88%
	Firefox/15.6/rv:15.6/Gecko/20100001, Win 5.7	4	88%
	Firefox/15.5/rv:15.5/Gecko/20100000, Win 5.6	5	88%
Test 4	**User-Agent (8) + User-Agent (9d) + User-Agent (9f) – four elements Swapping: Safari/5.1.7/534.57.2, Win 6.1 for:**	**Iteration**	**% Match**
	Safari/5.1.6/533.56.2, Win6.0	1	88%
	Safari/5.1.5/533.55.1, Win5.9	2	88%
	Safari/5.1.4/533.54.0, Win5.8	3	88%
	Safari/5.1.3/533.53.9, Win5.7	4	88%
	Safari/5.1.2/533.52.8, Win5.6	5	88%
Test 5	**User-Agent (8) + User-Agent (9c) + User-Agent (9e) – five elements Swapping: Chrome/23.0.1271.64 Safari/537.11, Win 6.1 for:**	**Iteration**	**% Match**
	Chrome/20.0.1270.64 Safari/523.11, Win6.0	1	87%
	Chrome/20.9.1269.64 Safari/522.11, Win5.9	2	86%
	Chrome/20.8.1268.64 Safari/521.11, Win5.8	3	86%
	Chrome/20.7.1267.64 Safari/520.11, Win5.7	4	86%
	Chrome/20.6.1266.64 Safari/519.11, Win5.6	5	86%
Test 6	**User-Agent (8) + User-Agent (9c) + User-Agent (9e) – six elements Swapping: Chrome/23.0.1271.64 Safari/537.11, Win 6.1 for:**	**Iteration**	**% Match**
	Chrome/22.0.1270.63 Safari/523.10, Win6.0	1	<85%
	Chrome/21.9.1269.62 Safari/522.09, Win5.9	2	<85%
	Chrome/21.8.1268.61 Safari/521.08, Win5.8	3	<85%
	Chrome/21.7.1267.60 Safari/520.07, Win5.7	4	<85%
	Chrome/21.6.1266.59 Safari/519.06, Win5.6	5	<85%