# COMBATING CLIENT FINGERPRINTING THROUGH THE REAL-TIME DETECTION AND ANALYSIS OF TAILORED WEB CONTENT

by

KENTON P. BORN

B.S., Kansas State University, 2006
M.S.E., Kansas State University, 2007

AN ABSTRACT OF A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2012

# ABSTRACT

The web is no longer composed of static resources. Technology and demand have driven the web towards a complex, dynamic model that tailors content toward specific client fingerprints. Servers now commonly modify responses based on the browser, operating system, or location of the connecting client. While this information may be used for legitimate purposes, malicious adversaries can also use this information to deliver misinformation or tailored exploits. Currently, there are no tools that allow a user to detect when a response contains tailored content.

Developing an easily configurable multiplexing system solved the problem of detecting tailored web content. In this solution, a custom proxy receives the initial request from a client, duplicating and modifying it in many ways to change the browser, operating system, and location-based client fingerprint. All of the requests with various client fingerprints are simultaneously sent to the server. As the responses are received back at the proxy, they are aggregated and analyzed against the original response. The results of the analysis are then sent to the user along with the original response. This process allowed the proxy to detect tailored content that was previously undetectable through casual browsing.

Theoretical and empirical analysis was performed to ensure the multiplexing proxy detected tailored content at an acceptable false alarm rate. Additionally, the tool was analyzed for its ability to provide utility to open source analysts, cyber analysts, and reverse engineers. The results showed that the proxy is an essential, scalable tool that provides capabilities that were not previously available.

# COMBATING CLIENT FINGERPRINTING THROUGH THE REAL-TIME DETECTION AND ANALYSIS OF TAILORED WEB CONTENT

by

KENTON P. BORN

B.S., Kansas State University, 2006
M.S.E., Kansas State University, 2007

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2012

Approved by:

Major Professor
Dr. David Gustafson

# ABSTRACT

The web is no longer composed of static resources. Technology and demand have driven the web towards a complex, dynamic model that tailors content toward specific client fingerprints. Servers now commonly modify responses based on the browser, operating system, or location of the connecting client. While this information may be used for legitimate purposes, malicious adversaries can also use this information to deliver misinformation or tailored exploits. Currently, there are no tools that allow a user to detect when a response contains tailored content.

Developing an easily configurable multiplexing system solved the problem of detecting tailored web content. In this solution, a custom proxy receives the initial request from a client, duplicating and modifying it in many ways to change the browser, operating system, and location-based client fingerprint. All of the requests with various client fingerprints are simultaneously sent to the server. As the responses are received back at the proxy, they are aggregated and analyzed against the original response. The results of the analysis are then sent to the user along with the original response. This process allowed the proxy to detect tailored content that was previously undetectable through casual browsing.

Theoretical and empirical analysis was performed to ensure the multiplexing proxy detected tailored content at an acceptable false alarm rate. Additionally, the tool was analyzed for its ability to provide utility to open source analysts, cyber analysts, and reverse engineers. The results showed that the proxy is an essential, scalable tool that provides capabilities that were not previously available.
.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ACKNLOWLEDGEMENTS

I would first and foremost like to thank Dr. Gustafson for his roles as committee chair, advisor, teacher, and friend; his help has been greatly appreciated.

I would also like to thank Dr. Andreson, Dr. Boone, Dr. Ou, and Dr. Scoglio for serving on the committee and offering their advice and support.

Lastly, I would like to thank Alice Lund, Wendell Peete, Honeywell FM&T, and the Department of Energy for their help, guidance, and understanding in allowing me to continue my research in this area

# TERMS AND DEFINITIONS

*Client Fingerprint* – The combination of visible and obtainable characteristics that can be used to identify a particular client system.

*Dynamic Web Content* – Web content that changes over time.

*Instantly Dynamic Content* – Web content that changes very frequently regardless of the client fingerprint or elapsed time.

*Tailored Web Content* – Web content that has been modified from an original template because of fingerprintable client information.

*Multiplexing Proxy* – A proxy that can receive a request, send out multiple and possibly modified versions of that request, collect multiple responses, and return a single response back to the client system.

*"The power of individual targeting – the technology will be so good it will be very hard for people to watch or consume something that has not in some sense been tailored for them"*

– Eric Schmidt, Google

# INTRODUCTION

Over the last decade, websites have steadily increased in complexity and dynamism. While they were initially composed of a minimal number of static resources, new technologies and frameworks now provide web developers with advanced capabilities beyond what was achievable with past tool sets.

Similarly, the client systems that access the websites have grown in complexity. This evolution in complexity came about for three reasons: there were many ambiguities in the published standards, developers did not always agree with and accurately follow the standards, and competing browsers in the market created proprietary functionality that could possibly give them an advantage in a saturated market. These issues have led to a plethora of compatibility problems that now plague the web. When Internet Explorer, Chrome, Firefox, Opera, and other browsers are considered along with the shift toward smart phones and other mobile computing devices, it is easy to see why websites are now moving toward dynamically tailored content for specific client fingerprints. While there are many legitimate uses for tailored content such as targeted advertising or browser compatibility, malicious servers can also deliver tailored exploits to users based on the particular location, browser, or operating system a client is using.

A web server can obtain many types of information about the connecting client, collectively referred to as the client fingerprint. Firstly, the server can determine the location of the system. This is typically accomplished using IP address geolocation. Secondly, the server can determine the client's browser. While the *User-Agent* header and *navigator* JavaScript object are the most used methods for browser fingerprinting, it is also possible to exploit the differences in the way various browsers handle the intricacies of building the Document Object Model (DOM). Using client-side JavaScript, it is also possible to fingerprint specific plugins enabled on the client browser. The server is able to determine the operating system of the client. This can be done by either fingerprinting the TCP/IP stack of the packets received from the client, or by parsing the User-Agent string. Tools such as P0f and Nmap (Nmap; P0f) provide powerful capabilities for identifying client operating systems. Lastly, through the use of tracking methods such as cookies, it is possible to identify the client's history. Based on all or part of the client fingerprint, a web server can tailor content with either useful or malicious intent.

Currently, there are no sufficient tools that can detect when a web server delivers tailored content toward specific client fingerprints. While several projects have attempted to detect and alert on changes to websites over time, very little research has been done regarding the real-time detection and classification of tailored information based on the client IP, browser, or operating system. Eli Parser, in a recent TED talk, expressed worry about how information online is being tailored in such a way that people are being put in a filter bubble where they are separated from important and differing points of view (Pariser 2011).

The ability to detect targeted misinformation attempts would be vital to anyone performing open source collection on the Internet. It is currently very difficult to provide a level of trust or validity to information collected on the web. The real-time identification and analysis of tailored web content would be a major accomplishment in this area.

Additionally, a cyber-security analyst would find it advantageous to detect when a server delivers a tailored vulnerability based on the client fingerprint. This would simplify detection, triage, triage and exploit collection, greatly reducing the necessary to reverse engineer malicious websites and build appropriate signatures for intrusion detection systems.

From a non-malicious perspective, the real-time detection of tailored web content would be greatly beneficial to reverse engineering and website testing tasks. It would provide insight into server-side behavior that is often not easily obtainable without the source code. Minor anomalies between web responses are difficult and time-consuming to find without a tool that can quickly analyze and categorize multiple responses simultaneously.

## BACKGROUND

The majority of systems on the web have a fingerprint that can be used to identify and classify that particular device. While privacy enthusiasts have successfully pushed to limit this information, it is still difficult to avoid many of the techniques commonly used for client identification.

An example of a powerful system capable of building robust client fingerprints on the web can be found at https://panopticlick.eff.org (Panopticlick). Panopticlick provides users with the entropy of their browser configuration by analyzing application layer information such as the Hypertext Transfer Protocol (HTTP)

headers, browser plugins, screen size, system fonts, and cookies. Because very few users share identical values across all the fingerprintable attributes, this can be used for monitoring and tracking a particular client without using sessions or cookies.

However, client fingerprints on the web are not limited strictly to tracking users. Various fingerprinting techniques at the IP and transport layer make it possible to identify significant information regarding the user's location and operating system. Additionally, the application layer can often be used to identify client software and configurations. For example, in the case of HTTP, it is typically possible for a server to be able to identify browser and plugin details of the visiting client.

**Location Fingerprinting**

In order for two systems to communicate on the web, they must have the IP address of the corresponding machine. Many databases such as IP2Location (IP2Location 2010) are available that have accumulated information about the geographical location of IP addresses. While not always accurate, many of these services, when provided with an IP address, can offer detail down to the latitude and longitude coordinates of a system.

Geolocation information can be collected from a variety of different techniques, often without the user's knowledge. Firstly, data is often provided from regional Internet registries such as the American Registry for Internet Numbers (ARIN). These registries are assigned blocks of IP addresses and are responsible for allocating smaller blocks of addresses to other groups and companies as necessary. However, this only provides a rough estimate of the location of an IP address owner, and often must be supplemented through other means. For instance, websites that offer location-based services such as package delivery or weather are able to aggregate IP addresses with entered locations and sell this information to companies offering geolocation services.

At the application layer, HTTP headers provide a limited amount of information that can be used for location fingerprinting. With most HTTP requests, a client includes the "Accept-Language" header (RFC2616). This header identifies the language preferences of the individual making the request. While it is rarely used in dynamic website creation, this information provides a very rough estimate of the geographical location of the individual by giving the client's preferred/primary language. While it has limited use for widespread languages such as English, it can work well for identifying locations with a

rare and not as widely used language. Because it only provides a very rough estimate, however, HTTP-Layer location-based fingerprinting is not considered useful compared to alternative methods such as IP geolocation databases.

The most common strategy for mitigating location-based fingerprinting with HTTP traffic is to use a proxy. A proxy is an intermediary device that makes a connection on behalf of a client system. Instead of sending traffic directly to a web server, requests are instead sent to a proxy. When the proxy receives a request, it forwards the request to the actual web server on behalf of the client. The web server, however, will only see the IP address of the proxy, and may have no knowledge of the actual client making the initial request. Once the proxy receives the response, it can then forward the response back to the initial client. While the proxy will require the IP address of the client, the web server will have no knowledge of where the original request was from. This makes proxies ideal for masking the actual location of the client. It should be noted however, that not all proxies work in this manner. One type of proxy, the transparent proxy, leaves the end client information in-tact, and should not be used for location-masking purposes.

One problem with anonymizing proxies is that they are very difficult to implement correctly. For instance, many implementations leak DNS queries directly from the client, only using the proxy for HTTP requests. Additionally, cookies, applets, and JavaScript tricks have all historically allowed the true identify of a client system to be revealed behind a proxy. Because of this, masking identity through a proxy is not always an ideal solution, and should be approached with caution.

Yet another major problem with using proxies to mask identity or location is that the third party providing the proxy must be trusted. Proxies have the ability to not only fingerprint the end user, but also view and modify the traffic entering and exiting the device.

**Operating System Fingerprinting**

Fingerprinting the operating system of a client is useful for many reasons. Firstly, it allows custom exploits to be delivered to targeted systems. Writing malware or exploitation code often requires a tailored technique for the targeted system. Instead of trying many attacks, it is both stealthier and more effective to launch one successful attack against a system. Secondly, detecting operating systems connecting to an internal web server can provide useful statistics about how many and what types of

systems are connected to a network. This can also be an effective way to identify an illegitimate, anomalous connection. Lastly, obtaining the fingerprint of an operating system is useful for social engineering (Nmap). Being able to convey detailed knowledge of a victim's system is a simple way of gaining their trust. Combined with other social engineering techniques, it can be a devastating piece of information that leads to powerful, effective attacks.

Operating system fingerprinting techniques are classified as either active or passive. The primary difference is that active fingerprinting can be detected by the target system, while passive fingerprinting cannot. The additional stealth obtained through passive fingerprinting, however, comes at the cost of both accuracy and availability.

Active operating system fingerprinting is typically performed by sending several probes that test how a target system handles ambiguous details of protocol RFCs. While published RFCs try to be as detailed as possible, there are inevitably small differences that will be seen in the final implementations of that protocol. Because of this, the Internet Protocol (IP), Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Internet Control Message Protocol (ICMP) often contain minor differences on each operating system that can be used as an identifying characteristic.

The Achilles' heel of active operating system fingerprinting is that it is detectable by both the target and monitoring devices in the target's path. Active fingerprinting typically requires a large number of anomalous probes that can be identified and acted upon. Because of this, it is not considered stealthy. While this is not an issue when it is being used to inventory a network, this becomes a large problem when the goal is reconnaissance for system exploitation by an adversary.

Nmap (NMAP) is one of the most popular active operating system fingerprinting tools available. It utilizes multiple probes over each of the standard protocols to accurately gauge the target system. In addition to looking at variations in particular protocol fields, Nmap uses timing techniques for TCP sequence number algorithm detection. Not only does Nmap test each protocol individually, but it also performs overlapping tests that analyze how one protocol may be affected by another. An example of this is the shared IP ID sequence number test, which determines whether or not the target shares sequence number information between transport layer protocols.

Because active operating system fingerprinting can be detected and can be considered abusive, passive operating system was developed as an alternative. With passive detection, there are no probes sent to the

targeted system. Instead, these types of tools will passively sniff legitimate traffic and attempt to identify systems based on the observed traffic characteristics.

The four most commonly used identifiers are the IP "Time To Live" (TTL) field, the IP "Don't Fragment" (DF) bit, the IP "Type of Service" (TOS) field, and the TCP "Window Size" field (Passive Fingerprinting). Because each operating system chooses its own default TTL value, this is often a powerful means of identifying the target system. While this number varies based on the number of hops between the client and detection point, it is typically simple to accurately predict what value it was initially set to. Similarly, operating systems are free to choose their own default TCP window size, and can use both the DF bit and TOS field as they please. This has led to a large number of varying implementations and distinguishing characteristics.

At the application layer, It is also possible to identify a client's operating system through the User-Agent header attached to most HTTP requests (RFC2616). For example, take the following Internet Explorer User-Agent string:

*Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)*

It is easily seen that the request was made from version 7 of the Internet Explorer browser, and the particular platform the browser is running on is a "Windows NT 6.0" system. This method is often very effective for identifying operating systems, wireless devices such as smart phones, and many of the bots and tools that are scraping websites.

Although specific operating system versions always have the same default fingerprint, it is possible to alter the default behavior. While this is seldom done, it is a risk that must be considered. Additionally, tools such as SCAPY allow custom packets to be built and sent from systems, modifying many of the fingerprintable attributes (SCAPY). Four different strategies can be used for mitigating operating system fingerprinting. Firstly, intermediary devices such as HogWash may be used to intercept and scrub fingerprintable characteristics from packets (HogWash). Secondly, operating systems can modify their fingerprint to mimic the fingerprint of other systems. This strategy is seen in tools such as IP Personality and IpMorph (IP Personality; IpMorph). Instead of completely mimicking another system, tools such as Iplog use another strategy that involves simply modifying a subset of packets to confuse fingerprinting tools into incorrectly guessing the operating system (IP Personality; Iplog). Lastly, tools such as Stealh

LKM simply patch the kernel to drop all packets having anomalous characteristics that could be used for fingerprinting (Security Technologies).

HTTP layer operating system fingerprinting can similarly be thwarted by using a false or custom User-Agent string. Additionally, a request is still valid if the User-Agent string is removed from the request entirely. User-Agent strings can be stripped both at the creation of the request or at an intermediary device such as a proxy. While taking these steps will help protect against web servers with malicious intent, it may also break legitimate websites that tailor content for that particular browser or system.

The issues outlined above prevent operating system detection from being fully reliable. Despite this, it has proven to be increasingly effective for both defending and attacking networks. Until less ambiguous specifications can be utilized for more standardized implementations, operating system fingerprinting will continue to be heavily researched and used for both defensive and nefarious purposes.

**Browser Fingerprinting**

Similar to operating system fingerprinting, browser fingerprinting can be applied with either helpful or malicious intent. Browser identification is typically accomplished using either User-Agent strings or the "navigator" JavaScript object that is initialized in most browsers. Another method involves exploiting scripting/DOM compatibility issues between different browsers. All of these methods are widely used for both legitimate and malicious purposes.

The most common method of browser fingerprinting involves analyzing the User-Agent header of HTTP. It is common practice for browsers and tools to attach a User-Agent string to each request. This User-Agent string typically contains information that uniquely identifies the type of browser, version of that browser, and the platform running that particular browser. A popular method for parsing the User-Agent string for necessary information is the HTTP::BrowserDetect Perl module (HTTP::BrowserDetect). This script is passed a User-Agent string and provides detailed information about the tool or browser that made the request. This allows it to be easily integrated and queried in real-time from scripts or other software on the server.

Another common strategy for identifying browsers and platforms is to use the *navigator* object inside JavaScript. Most browsers initialize the "navigator" object to provide information about the browser and

platform of the client. This strategy is used in the popularized BrowserDetect script (BrowserDetect 2010). However, if this information is required on the server, it becomes necessary to relay this information back to the server in an additional message through technologies such as AJAX.

Yet another method involves developers creating scripts in a way such that certain code will only run when it is executed in a particular browser. This type of browser fingerprinting is done by exploiting small differences in how the browsers handle malformed syntax, ambiguous details of the specification, and custom objects only present in a subset of the browsers. For example, the existence of the *document.documentMode* object can be used to identify Internet Explorer 8 or higher, and *window.globalStorage* can be used to uniquely identify Firefox 2 or higher (Using Object Detection to Sniff Out Different Browsers). Object detection is the preferred method of tailoring web pages for specific browsers, and is considered significantly safer than using information from the User-Agent string or navigator object. While future browsers will likely know what objects it must support, current implementations cannot predict what future User-Agent strings it must handle.

While not yet used significantly for browser fingerprinting, clients can be identified and tracked through many browser settings and plugins. Panoptoclick, for example, shows how one can uniquely identify visiting clients through these browser-based characteristics (Panopticlick). Not only does Panopticlick look at the User-Agent string, but it also examines things such as the HTTP_ACCEPT header, the browser plugins, the time zone, the screen size, the system fonts, and the types of cookies that are enabled. An example screenshot of Panoptoclick's output is shown in Figure 1, yielding 20.34 bits of identifying information for the particular browser used.

| Browser Characteristic | bits of identifying information | one in $x$ browsers have this value | |
|---|---|---|---|
| User Agent | 10.58 | 1527.23 | Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit/534.10 (KHTML, like Gecko) Chr |
| HTTP_ACCEPT Headers | 4.04 | 16.41 | text/html, */* ISO-8859-1,utf-8;q=0.7,*;q=0.3 gzip,deflate,sdch en-US,en;q=0.8 |
| Browser Plugin Details | 20.34+ | 1330215 | Plugin 0: Chrome PDF Viewer; Portable Document Format; pdf.dll; (; application/pdf; pdf). Plu Update; npGoogleOneClick8.dll; (; application/x-vnd.google.oneclickctrl.8; ). Plugin 4: Java De (Java Applet; application/x-java-applet; ) (JavaBeans; application/x-java-bean; ) (; application/x-ja ) (; application/x-java-applet;version=1.1.3; ) (; application/x-java-bean;version=1.1.3; ) (; ap application/x-java-applet;version=1.2.1; ) (; application/x-java-bean;version=1.2.1; ) (; application java-applet;version=1.4.2; ) (; application/x-java-bean;version=1.4.2; ) (; application/x-java-appl java-vm-npruntime; ). Plugin 6: Microsoft Office 2010; Office Authorization plug-in for NPAPI brow ). Plugin 8 |
| Time Zone | 4.25 | 18.97 | 480 |
| Screen Size and Color Depth | 6.03 | 65.14 | 1920x1080x32 |
| System Fonts | 20.34+ | 1330215 | Marlett, Arial, Arabic Transparent, Arial Baltic, Arial CE, Arial CYR, Arial Greek, Arial TUR, E DotumChe, Impact, Iskoola Pota, Kalinga, Kartika, Khmer UI, Lao UI, Latha, Lucida Console, Ma Mincho, MS PMincho, MV Boli, Microsoft New Tai Lue, Nyala, Microsoft PhagsPa, Plantagenet C CYR, Times New Roman Greek, Times New Roman TUR, Tunga, Vrinda, Shonar Bangla, Microso Sakkal Majalla, Traditional Arabic, Aharoni, David, FrankRuehl, Levenim MT, Miriam, Miriam Fix Comic Sans MS, Consolas, Constantia, Corbel, Franklin Gothic Medium, Gabriola, Georgia, Pala Eras Light ITC, Eurostile, Felix Titling, Franklin Gothic Book, Franklin Gothic Demi, Franklin Gothi MS Reference Sans Serif, MS Reference Specialty, DejaVu Sans Condensed, DejaVu Serif, G Garamond, Monotype Corsiva, Algerian, Baskerville Old Face, Bauhaus 93, Bell MT, Berlin Sa Kunstler Script, Lucida Bright, Lucida Calligraphy, Lucida Fax, Lucida Handwriting, Magneto, M MT, Tw Cen MT Condensed, Script MT Bold, Rockwell Condensed, Rage Italic, Pristina, Perpetu Gill Sans MT Ext Condensed Bold, Gigi, Franklin Gothic Medium Cond, Forte, Eras Medium ITC, |
| Are Cookies Enabled? | 0.35 | 1.28 | Yes |
| Limited supercookie test | 1.14 | 2.21 | DOM localStorage: Yes, DOM sessionStorage: Yes, IE userData: No |

**Figure 1. Output of the Panoptoclick uniqueness test**

The biggest issue with browser self-identification methods such as User-Agent strings is that they are easily spoofed. Many plug-ins and devices allow custom and misleading headers to be sent in place of the legitimate string, fooling servers into believing the client is running a browser or platform other than what actually made the request. Additionally, the User-Agent string is not required, allowing tools or intermediary devices such as anonymizing proxies to choose not to send the User-Agent string along with the request. However, the downside of using object detection instead of the User-Agent string is that either all of the necessary code for each browser must be sent in the initial response, or browser information has to be sent back to the server through technologies such as AJAX before tailored content can be delivered to the client in a future response.

## PROBLEM STATEMENT

Websites are no longer composed of simple, static HTML web pages. With the complex frameworks and web design in use today, web pages often contain instantly dynamic web content. Two simultaneous visitors to a website may not receive identical responses, making it difficult to determine whether website changes are content-based or simply an insignificant modification of an underlying HTML attribute. Similarly, it is not possible to tell why the change occurred. While it could have been a benign modification, it is also possible that the server returned tailored information for the particular client making the web request.

Web servers return tailored web content for many valid reasons. For instance, a website may offer a location-based service and want to tailor information based on the geolocation data of the connecting IP address. Another example would be tailoring a download to the particular platform or browser connecting to the website. Multiple users with different operating systems would typically be required to select a download appropriate for their system. However, after automatically detecting their operating system on the server side, the correct download could automatically be started, sparing the user of having to complete the selection step. This type of behavior is often seen in browsers when a website aides the user in downloading the appropriate plugin for the particular browser they are using. Advertising is also a large component in the rise of tailored content. By tracking users through their client fingerprint, it is possible to advertise directly to that particular user without requiring sessions or cookies, boosting revenue through higher click-through rates. Not only can advertisements be delivered based on tracking a user, but advertisements, plugins, and upgrades themselves can be tailored toward the client system making the request.

While it is easy to see why the industry is shifting toward tailored content, similar strategies also provide malicious servers with the capability to deliver tailored payloads to users based on the client fingerprint. Since different operating systems and browsers typically require unique exploits for each platform and version, it becomes critical to be able to gain as much information about the system as possible before attempting to exploit it. Discussed previously, it is much less stealthy to launch several attacks in the hopes that one of them work. Intrusion Detection Systems (IDS) can often pick up on the malicious activity and provide network administrators with alerts that may be investigated. Alternatively, it is stealthier to know exactly what exploit and technique is needed before the first attack is sent. This minimizes detection possibilities and provides the malicious actor with a much less detectable foothold into the network.

From a malicious standpoint, browser-based tailoring would be used to exploit vulnerabilities in the particular browser making the request. While most (if not all) browsers are considered to have vulnerabilities, specific exploitable weaknesses typically only apply to a small subset of browsers or versions. Figure 2 (below) shows examples of browser vulnerabilities logged in the National Vulnerability Database (NVD). As one can see, the vulnerabilities are tied to versions of a specific browser rather than being applicable to all of them. For a malicious webserver, this means that it must tailor exploits toward the browser fingerprint that is collected from the client request. While all the exploits for different browsers may be used together, it is likely to increase the probability of being caught by an intrusion detection system or firewall, and it requires giving all the available exploits to anyone investigating the response down the road.

**CVE-2010-3342**

***Summary:*** Microsoft Internet Explorer 6, 7, and 8 does not prevent rendering of cached content as HTML, which allows remote attackers to access content from a different (1) domain or (2) zone via unspecified script code, aka "Cross-Domain Information Disclosure Vulnerability," a different vulnerability than CVE-2010-3348.

***Published:*** 12/16/2010

***CVSS Severity:*** 4.3 (MEDIUM)

**CVE-2010-3776**

***Summary:*** Multiple unspecified vulnerabilities in the browser engine in Mozilla Firefox before 3.5.16 and 3.6.x before 3.6.13, Thunderbird before 3.0.11 and 3.1.x before 3.1.7, and SeaMonkey before 2.0.11 allow remote attackers to cause a denial of service (memory corruption and application crash) or possibly execute arbitrary code via unknown vectors.

***Published:*** 12/10/2010

***CVSS Severity:*** 9.3 (HIGH)

**CVE-2010-4491**

***Summary:*** Google Chrome before 8.0.552.215 does not properly restrict privileged extensions, which allows remote attackers to cause a denial of service (memory corruption) via a crafted extension.

***Published:*** 12/07/2010

***CVSS Severity:*** 4.3 (MEDIUM)

**CVE-2010-4167**

***Summary:*** Untrusted search path vulnerability in configure.c in ImageMagick before 6.6.5-5, when MAGICKCORE_INSTALLED_SUPPORT is defined, allows local users to gain privileges via a Trojan horse configuration file in the current working directory.

***Published:*** 11/22/2010

***CVSS Severity:*** 6.9 (MEDIUM)

**Figure 2. Browser vulnerabilities in the National Vulnerability Database.**

Malicious operating system-based tailoring is also aimed at taking advantage of specific vulnerabilities that only affect a subset of systems. When used in conjunction with browser vulnerabilities, the operating system must be discovered for installing the appropriate malware on the end system as opposed to

exploiting a vulnerability (which has been done in the browser, in this case). However, escalated privileges or persistence cannot always be achieved through the browser vulnerability. In these cases, the malicious actor must use the browser vulnerability as a launching point for an operating system vulnerability that will provide the necessary privileges or foothold in the target system. These actions can be most stealthily performed using the various client fingerprinting methods described above.

Malicious Location-based tailoring, on the other hand, primarily relates to misinformation or psychological operations. The Internet has a powerful ability to sway the opinions of the public. This fact can be exploited by creating websites that alter the response based on where the request is coming from. This technique could be used to hide the true content of a website from opposing groups (for example, foreign entities), or even be used to manipulate specific pieces of an article, providing a targeted adversary with misinformation.

# HYPOTHESIS

A Multiplexing proxy system, through the real-time detection and categorization of tailored web content that has been modified for specific locations, browsers, and operating systems, provides enhanced misinformation, exploit, and web design analytics. This study will analyze the utility of the multiplexing proxy's detection, classification, and visualization methods for three different roles: open source analysts, cyber analysts, and reverse engineers. Both qualitative and quantitative approaches will be taken in an attempt to understand not only the classification accuracy of the tool, but also understand the limits of automated and manual techniques that can be applied to analyzing instantly dynamic and tailored web content.

Open source analysts currently cannot gauge the legitimacy of content returned by the websites they visit during information collection activities. At best, they are able to mitigate the possibilities of targeted manipulation by using anonymization services. However, many of these services are detectable themselves, and require trusting a third party also capable of manipulating the content. Additionally, as demonstrated by Panopticlick, it is very difficult to correctly and fully anonymize a request. The multiplexing proxy simultaneously makes requests from many different locations. By aggregating and analyzing the responses against each other, the proxy can detect changes in one response that are anomalous when compared with the changes seen in all other responses. While it is useful to simply mitigate the ability of an organization to conduct misinformation over the web, the ability to detect misinformation provides critical and actionable information.

Multiplexing requests over varying client fingerprints provides cyber analysts with the ability to more easily trigger tailored malicious content during an investigation. While many malicious websites will use object detection or a shotgun approach instead of passive fingerprinting, there is nothing to stop websites or intermediate proxies from delivering tailored exploits to the client systems making a web request. When a malicious actor uses object detection to deliver an exploit, they typically must provide all their exploits to clients making a request. Sensitive exploits may be preserved by using passive fingerprinting to identify the browser and operating system of clients, only sending the necessary exploit for that particular system. Additionally, many malicious actors may only have one current exploit, limiting their capabilities to only one particular type of system. If the proper browser is not detected for that exploit, malicious behavior may never be triggered. By multiplexing the request to mimic many systems, tailored exploits will be triggered and detected during analysis of the responses.

Current technologies limit the ability of reverse engineers to analyze existing websites. While performing static analysis of web content is possible, it is difficult to easily identify dynamic components of the markup that may change with each request. By using the multiplexing proxy to aggregate multiple responses, a reverse engineer can also analyze instantly dynamic and tailored components. A problem with many analysis tools is that they only concentrate on the resources required for initial rendering. Because the multiplexing proxy continues analyzing the website as additional traffic is generated, future AJAX calls will be captured and analyzed along with the original resources. While just having the ability to analyze dynamic content is useful, the ability to identify content tailored for a specific fingerprint provides a capability that is truly unique to the multiplexing proxy. By identifying these anomalies and providing the user with an interface for analyzing them more deeply, the multiplexing proxy helps identify interesting dynamic behavior that can save a reverse engineer valuable time.

Determining the utility of the multiplexing proxy for each role described above serves as a guide for the correct direction of future research in this area. While many studies have taken a quantitative approach to studying dynamic content over millions of websites, none of them perform a qualitative, manual analysis of instantly dynamic and tailored content. This information will help determine whether or not the ephemeral data can be classified accurately, or whether the number of false positives and false negatives will simply be too vast for an automated approach**.** Analyzing the multiplexing proxy under varying roles answers the question of whether or not it is worth the additional time and effort to use a system capable of detecting and analyzing tailored web responses.

# RELATED WORK

The ephemeral web has created issues for many web searching technologies. Crawlers and web search engines must use more sophisticated techniques to handle the increasingly more dynamic web content. For example, one study monitored 720,000 pages and found that 25% of pages in the .com domain changed within a day (Cho and Garcia-Molina 2000). Other domains such as .gov, however, showed significantly less dynamism. While it only took 11 days for half of the .com domains to change, it took the .gov pages over 4 months to reach the same point. The authors concluded that an incremental crawler would perform significantly better than a batched crawler. Olston and Pandey came to a similar conclusion in their research on recrawl scheduling for search engine optimization (Olston and Pandey 2008). The authors developed web crawling policies that accounted for the longevity of the information, instead of simply performing batched crawling. While they wanted to maintain a suitable "freshness" of each page that has the potential to change, they also aimed to minimize analyzing any ephemeral web content that would likely change before a search engine could harness the information for better results. An example the authors give is a web page displaying a "quote of the day". By the time the web page is analyzed and included in search results, the quote will likely have changed, resulting in a misleading page rank for certain keywords. Both studies concluded that web crawlers must shift from using periodic, batched strategies to complex, incremental crawlers that attempt to only analyze the data that was most likely to have been updated since it was last crawled. By keeping track of the resources that are more likely to change over time, the crawler can more effectively stay up to date, spending less time on parts of the web that are static.

While Cho and Garcia only measured web page differences using a simple MD5 checksum, a larger and more fine-grained study was done monitoring over 150,000 web pages over 11 weeks (Fetterly et al. 2003). Instead of simply using checksums that do not indicate the properties or amount of modified content, the authors combined this technique with a feature vector of syntactic properties of the document using a shingling metric. In this study, the authors found that most changes in websites were trivial and not necessarily a change in content. The first interesting relationship identified was a greater frequency of change in top-level domains. Secondly, they found larger documents often had both a higher frequency of change and degree of change. Lastly, they found that past changes can often be used to predict the size and frequency of future changes.

The dynamic nature of web content was analyzed in another study by looking at the change frequency of 55,000 web pages (Adar et al. WSDM 2009). Shown in table 1, it can be seen that over half the websites

had frequently changing data, and over 10% of the websites contained instantly dynamic content that changed with every request. They attributed the instantly dynamic web content to randomly changing advertisements, page counters, unique identifiers, and server metadata such as how long it took to serve the page. While these sites showed changes in every response, each successive change modified a similar amount of information. This is contrasted with sites such as blogs which will change significantly with each update as new posts push the rest of the content down or off the page.

**Table 1: Changes in sub-hourly crawls**

| Interval | Docs. with change (% of 54816) | | Mean / Median Dice sim. (given change) |
|---|---|---|---|
| | *Change in any sample (upper bound)* | *Change in all samples (lower bound)* | |
| **Instant (0 min.)** | 6303 (11.5%) | 3549 (6.5%) | .937 / .985 |
| **2 minutes** | 10616 (19.3%) | 4952 (9%) | .937 / .982 |
| **16 minutes** | 12503 (22.8%) | 6206 (11.3%) | .927 / .975 |
| **32 minutes** | 13126 (23.9%) | 6445 (11.8%) | .920 / .969 |
| **60 minutes** | 35997 (65.7%) | 22818 (41.6%) | .867 / .950 |

*(Adar et al. WSDM 2009)*

The authors used an XPath extraction approach for analyzing website similarities. A serialized version of the document was created by using a SAX parser against a "cleaned" version of the website (preventing the SAX Parser from failing). For each element encountered, the system would emit the XPath of the element, a hash value of the text within that element, and the hash value of the children of that element. By analyzing each DOM element separately, they could calculate the survivability of each element. This turned out to be an effective approach, seeing a median survival rate of 99.8% after 5 weeks. However, the authors admit that extraction based on XPath failed to perform well on unstable content. By identifying the parts of the page that had instantly dynamic data, they hoped to be able to improve crawler and web searching functionality by focusing only on the relevant content and changes.

Ntoulas et al. studied many aspects of the ephemeral web (Ntoulas et al. 2004). Firstly, the authors looked at the birth and death rate of web pages over time. They found that approximately 8% of downloaded sites each week were new web pages that had not been previously seen. However, this measurement was calculated by simply comparing URLs, making it possible that a previous resource was simply moved to a new URL. Secondly, they calculated the TF-IDF Cosine distance and the word distance for different versions of the same page in an attempt to measure the distribution of degree of change. The authors

found that the large majority of changes to the websites they monitored were minor; not affecting the TF-IDF Cosine distance and word distance in a significant way.

A study on web changes at Seoul National University also looked at how many new URLs were found over time (Kim and Lee 2005). After crawling and accumulating sites for 100 days, they found that 40% of the URLs being crawled were not found in the initial crawl. This study also calculated the download rate, modification rate, and coefficient of age for websites over time. However, like many other studies, there is little done to mitigate websites with instantly dynamic data.

A separate work at Seoul National University focused only on the change in content of specific web pages (Kwon et al. 2006). Unlike the previous study that looked at new pages and content over time, this study focused on methods of comparing two versions of a website for differences. Strategies found in previous studies consisted of byte-to-byte comparison (Hash values), TF-IDF cosine distance and word distance, edit distance, and a shingling metric. These methods were tested for their effectiveness in detecting different types of changes. Each change was classified by being an "add", "drop", "copy", "shrink", "replace", or "move" operation. The shingling metric performed best for "add" and "drop" changes, but was oversensitive to the rest. In contrast, the authors felt that the TF-IDF metric was not sensitive enough to most changes. Word distance only proved effective for "replace" changes, and had reported nothing for text that simply moved. The edit distance metric only differed from the word distance metric in that it treated "move" and "replace" changes similarly. One important aspect of this study is that the markup was removed; not taking into account changes in element attributes or structure.

Dontcheva et al. looked strictly at structural changes in web pages over time (Dontcheva et al. 2007). Their strategy consisted of "cleaning" the websites into a proper XHTML form, removing structurally irrelevant elements, then analyzing the resulting structure of the DOM tree. The authors found that small changes or layout modifications often happened to the leaves of the DOM tree as opposed to major website changes that modified the elements higher up in the tree. Additionally, websites with large amounts of traffic and highly dynamic content tended to have a larger number of structural changes than the less dynamic sites. The authors noted that it was the sites with changes away from the leaf nodes that were increasingly difficult for automated extraction of information. However, their approach did not take into account AJAX or Flash applications that could dynamically modify the structure of the content at any time. Because they stripped out script tags and only analyzed the website's initial response, it is very likely that they did not receive a clear representation of the structure of the DOM tree. Since many websites are making it harder and harder to accurately analyze the content accurately, the authors

concluded that it would be appropriate for the user to play a role in guiding the content extraction and analysis.

As an extension of previous work on revisitation behavior (Adar et al. CHI 2008), Adar et al. analyzed the relationship between revisitation patterns and changes to the website over time (Adar et al. CHI 2009). The authors felt user experience could be enhanced by highlighting the relevant content that likely attracted the user back to the site. For this study, 40,000 websites were crawled every hour, compared against historical crawls for modifications, then analyzed against the access log file. This information was used to help identify the content the user would be revisiting the site to see. While both change tracking and revisitation behavior had been looked at separately in previous studies, this work provided a new look at the correlation between the two. By polling for the user's intentions when visiting websites, the authors found meaningful relationships that could not previously be proven. For instance, sites that had a high rate of change and were visited frequently were often the result of users searching for new information. Conversely, sites that did not have significant change but were visited frequently were users wanting to revisit something they had viewed previously.

Significant work has been done in detecting changes in XML and HTML documents.
A popular method of detecting differences is to solve the longest common subsquence (LCS) problem. Diff, a popular Unix program, popularized this strategy. HtmlDiff , created in 1996, applied the LCS problem to HTML using the Hirschberg algorithm, highlighting HTML changes for visualization. Efficient HTML differencing was further studied by Mikhaiel and Stroulia (Mikhaiel and Stroulia 2005) using a labeled-ordered tree comparison. Many other processes and algorithms have been developed in attempts to find meaningful change detection in structured data, Chawathe and Garcia-Molina (Chawathe and Garcia-Molina 1997) approached the problem by computing a minimum cost edge cover of a bipartite graph. X-Diff (Wang et al. 2003) is an efficient algorithm that detects changes in XML using tree-to-tree correction techniques. This work was extended with X-Diff+ (Xing et al. 2008), which gave a visual representation of how an XML document conformed to its DTD. This research can easily be seen to be applicable considering current popularity of XHTML.

Because LCS can be computationally expensive, AT&T developed an algorithm called TopBlend that solved the heaviest common subsequence problem using the Jacobson-Vo algorithm (Chen et al 2000). TopBlend was used as part of the AT&T Difference Engine (AIDE), which allowed users to visualize website changes over time (Douglis et al. 1998). Using a web-crawler, the engine collects temporal versions of web pages and highlights differences between them using TopBlend.

Another tool, Zoetrope, was created specifically for visualizing and exploring the ephemeral web (Adar et al. UIST 2008). Using a crawler and a database, the tool builds a collection of documents and snapshots of websites over time. When the user requests the website, he or she is allowed to explore the website through different "lenses", providing the user with a historical view of subsets of the document intertwined with updated information. It can be used both for studying underlying text and structure, or it can be used to examine the user-facing graphics over time.

The goal of this research differs from the AIDE and Zoetrope in that it attempts to detect real-time tailored content instead of simply looking at temporal changes. While these tools only compare version of the website at different points in time (and using the same fingerprint), this research compares versions of the website retrieved at almost identical points in time from varying fingerprints.

While AIDE and Zoetrope focused on tracking and comparing the text of sites, Greenberg and Boyle took a different approach to displaying historical changes to users (Greenberg and Boyle 2006). In their work, they allowed the user to select regions of interest on web pages. This was stored in a bitmap representation and compared against similarly clipped regions of the website at later times. When significant change is detected in the bitmap, the user is notified, and can either browse the history of images that have been captured, or can have them played back in a video stream. While allowing users to select a region of interest is a useful approach, using bitmaps of clipped regions will be limiting and ineffective in many cases. Often, modifications will greatly alter the size or positioning of relevant data. While it will effective notify the user of the change, it may have difficult displaying those changes to the user.

Liu et al. also attempted to display website differences through visualization tools (Liu et al. 2000). However, their focus was on competitor website comparisons for business intelligence. In their work, they used hierarchical clustering and visualization to explore similar information found while crawling each of the sites. The hierarchical clustering allowed the user to choose the granularity of similarity they felt was appropriate for the information they were researching.

Many websites and programs offer services that monitor websites and alerts users when changes are detected. WebCQ, for example, is a notification system developed at the Georgia Institute of Technology. The notification system crawls through websites looking for changes and notifying users based on the type of changes it sees (Liu et al. 2000). Similar to AIDE, it has the ability to visually display any changes

to the user. However, it differs in that its purpose is as a notification system as opposed to a query system that can work in real-time. While some notification services will only notify the user that a change was made, many will highlight the changes, attempting to show them in line with the modified content. Additionally some services will perform searches for relevant keywords or phrases, only notifying the user when the content is something they are particularly interested in. Below, table 2 and table 3 show some of the most popular web-based and desktop-based services available.

**Table 2. Web-based website monitoring services (Tracking Changes)**

| NAME | WEBSITE |
|---|---|
| Change Detection | http://www.changedetection.com/ |
| ChangeDetect | http://www.changedetect.com/ |
| Femtoo | http://femtoo.com/ |
| FollowthatPage | http://followthatpage.com/ |
| Infominder | http://www.infominder.com/ |
| Page2RSS | http://page2rss.com/ |
| Watch That Page | http://www.watchthatpage.com/ |
| Websnitcher | http://websnitcher.com/ |

**Table 3. Desktop-based website monitoring services (Tracking Changes)**

| NAME | WEBSITE |
|---|---|
| Copernic Tracker | http://www.copernic.com/en/products/tracker/ |
| Internet Owl | http://www.internetowl.com/ |
| Update Patrol | http://www.updatepatrol.com/ |
| Update Scanner for Firefox | http://updatescanner.mozdev.org/en/index.html |
| Website Watcher | http://aignes.com/ |

A real-time comparative web browser (CWB) was demonstrated in 2003 that allowed the user to browse and compare different web pages containing similar content (Nadamoto and Tanaka 2003). This browser displays and synchronizes multiple web pages based on passage relevance on the two pages. As the user scrolls through an article, similar content is found on the other webpage and displayed simultaneously. This was extended with a multi-language functionality in the Bilingual Comparative Web Browser (B-

CWB). This extension focused on bilingual browsing, attempting to accomplish the same feats when the visible websites are not in the same language (Nadamoto et al. 2005). However, the CWB or B-CWB does not focus on changes to one particular website. Alternatively, it detects similar contexts between different websites and visually maps them. In contrast, this research focuses on the differences of one website as viewed from varying client fingerprints.

The Selenium project was created by the testing community to mitigate the difficulty of testing web applications over multiple browsing platforms (Selenium). Selenium provides an API for invoking web requests in varying browsers and programmatically testing the responses. This approach can utilize a cloud-like architecture, simultaneously testing different browsers at scale. However, the Selenium framework's primarily goal is to ensure proper functionality in multiple browsers as opposed to comparing the responses received from various browser fingerprints.

A similar project out of Mozilla Labs, TestSwarm, also tried to solve the problem of testing web applications over multiple browsing platforms (TestSwarm). However, TestSwarm differed from Selenium in that it used crowdsourcing to achieve scalable and diverse testing. Like Selenium, however, its goal is not the real-time detection of tailored web content.

No projects have been found attempting to tackle the problem with a similar approach. It is believed that the multiplexing proxy uses a unique and powerful strategy to achieve the real-time detection of tailored web content.

# METHODOLOGY

The real-time detection of tailored web content requires knowing the responses for varying client fingerprints at the same point in time. A multiplexing proxy accomplishes this by receiving requests and multiplexing them to the web server using a combination of both duplicate and varying client fingerprints (figure 3, below). As the responses are received back at the proxy, they can be aggregated in a data store where they may be compared and analyzed against the original response. The results may then be sent back to the client in the form of a simple HTTP response. By multiplexing HTTP requests and only modifying the information obtainable through passive fingerprinting, the tool can present the user with detailed analysis of how web server responses change due to fingerprintable attributes of the client making the request.

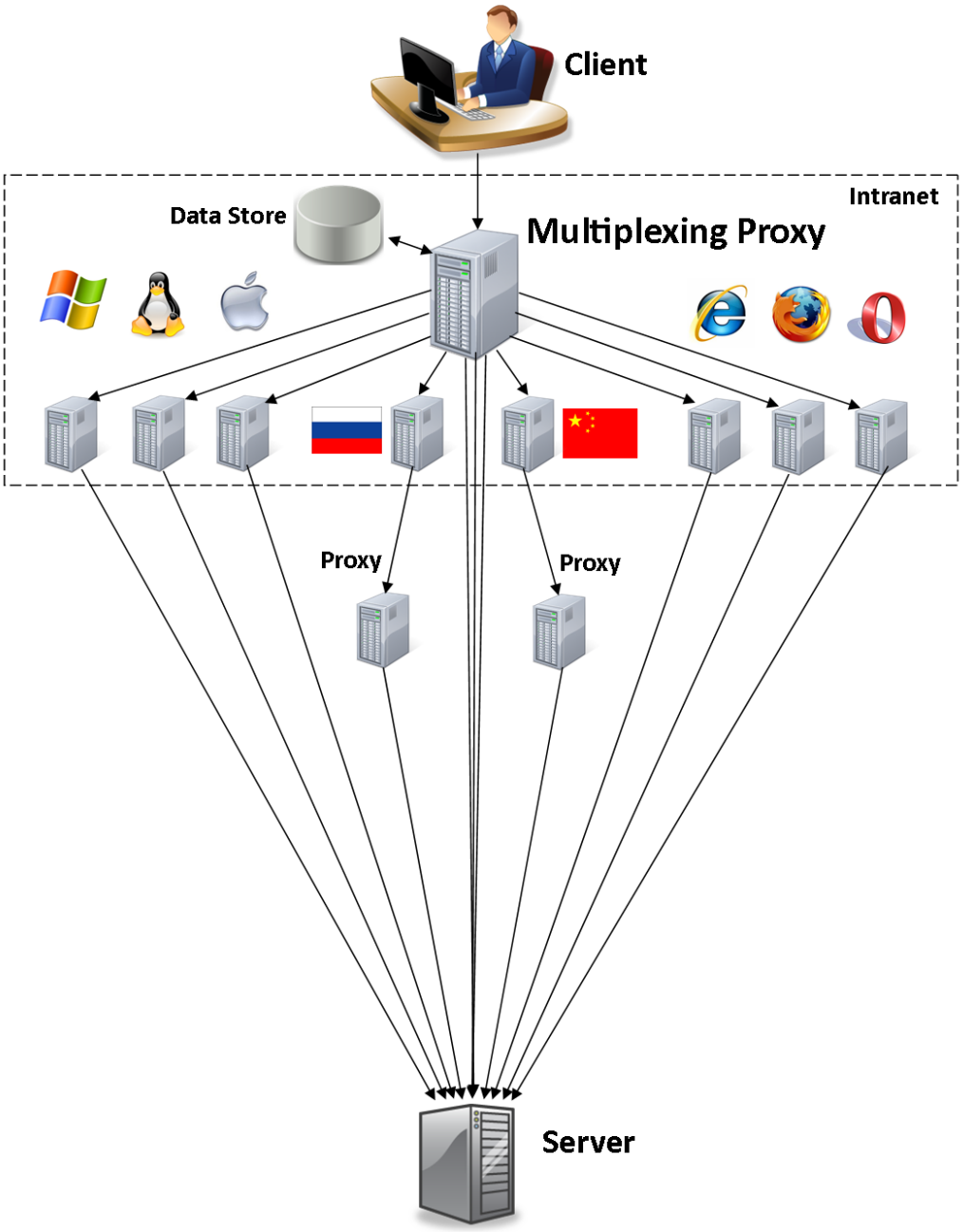



**Figure 3. The Multiplexing Proxy System**

**SYSTEM FLOW**

The flow of the multiplexing proxy system involves six steps:

1. The client configures the multiplexing proxy, determining both how many duplicate requests should be made, and determining what fingerprintable information should be modified or spoofed. This ranges from modifying the User-Agent string to rerouting the traffic through an external proxy.

2. When the multiplexing proxy receives a request, it identifies the user and analyzes any custom headers that were added from any client tool in the multiplexing system. If custom headers exist, they are stripped from the request before it is multiplexed.

3. Using the custom configuration for the identified user, the proxy multiplexes incoming requests. This involves both making duplicate requests and modified requests.

4. As responses are returned to the multiplexing proxy, they are held in a data store until it is time to analyze them against the original response. The data store can range from memory to a long-term database. Optionally, initial calculations and analysis not requiring the other responses can be performed and stored along with the response.

5. The multiplexing system compares and analyzes the responses against the original response using the techniques outlined in the *Analysis* section, below.

6. After the analysis is performed, the multiplexing system relays this information to the user.

One option for providing the user with the analysis is to modify the original response by highlighting the areas where changes were detected. For example, a red border could be added around images or text that was modified. Additionally, content can be injected around the borders of the original content that summarizes the changes "inline" with the response. Yet another option is having the client send a separate request to a web service that provides detailed analysis of the initial request. An example of this option is a browser plugin that loads the original request in one tab and simultaneously loads the analysis in a separate tab. This provides the user with uninhibited browsing while still allowing them to see the full analysis at any time. This strategy can be supplemented with a toolbar that provides a condensed summary of the analysis.

An important aspect of multiplexing requests is only modifying one fingerprintable attribute in each of the requests in step three. If more than one attribute is altered, it becomes difficult to disambiguate why the change occurred. For example, if the modified browser fingerprint elicited a change, but using a proxy in another location had no effect on the response, we can then give significant confidence to the likelihood that data was being tailored strictly toward the browser fingerprint used when a modified response was received.

For each operating system in the configuration, a request can either be created from that particular operating system, or the TCP/IP stack can be altered on a system to mimic the chosen platform.

There are several possible options for handling configurations requiring browser fingerprint modifications. Firstly, the user can be given choices of browsers to mimic, allowing the proxy to choose appropriate User-Agent strings. Another option is to allow the user to choose (or enter) his or her own User-Agent strings to use. Lastly, the proxy can take the original request and send it to processes (possibly on varying or virtual machines) that load the request in the selected browsers (or something that can emulate them). While this last strategy is the most difficult, it could lead to advantages when searching for tailored resources brought in through scripted object detection or instantly dynamic URLs.

The location modifications require the use of additional proxies outside of the primary intranet hosting the multiplexing proxy system. For each location chosen in the configuration, requests must be routed through external proxies at those particular locations. Optionally, modifications can also be made to HTTP headers that give hints as to the location of the user (i.e. the Accept-Language header).

**SYSTEM COMPONENTS**

A viable architecture must either multiplex the requests using a robust client-side tool or a functionality enhanced proxy. A proxy seemed like the most viable solution because of the smaller amount of information and code that would have to be maintained at the client. Performing the multiplexing at a proxy allows specific details to remain hidden from the end user. Information such as the network of external countries may be sensitive and very dynamic. If a proxy was not used, the client would be required to communicate directly with the external proxies providing IP address manipulation. Additionally, the client would be required to emulate the TCP/IP stacks of all the necessary operating

systems for the chosen configuration. The proxy strategy, however, allows requests to simply be made by a system in the same intranet as the proxy that is capable of handling the TCP/IP stack required by the configuration.

Because of the popularity of proxies, it was likely that an existing proxy could be leveraged for this project. Researching the design and source code of several open source projects led to the discovery of the Pro-Active Web filter (PAW). PAW not only has a built-in proxy, but also provides clean interfaces for accessing and manipulating the requests and responses as they flow between the client and the server. Access to the requests allows custom HTTP headers to be analyzed and stripped, and provides an easy location to multiplex the request based on the user's configuration. Similarly, having access to the responses provides an easy location to insert the response into a data store and perform initial analysis. If an inline injection strategy is desired for reporting anomalies, this also gives a convenient location to perform the necessary response modifications.

A widely usable multiplexing system would require allowing the users to customize the behavior of the proxy to fit their needs. While a general tailoring detection capability would be useful, there may be many circumstances that would require a specific browser or location to be tested. For instance, it may be of direct interest for someone to determine whether traffic coming from another country was treated the same way as traffic coming from the United States. Since each user requires a different configuration, it is necessary to tie configurations with specific users or requests.

At first, having the user pass configuration information to the proxy through a browser plugin seemed like a good approach. The toolbar would allow the user to select operating systems, browsers, and locations to use for the following requests. As requests were made, the browser plugin would intercept them, adding custom HTTP headers to the request that described the configuration. When the proxy received the requests, it would multiplex the requests according to this configuration.

Although building the ability to modify the configuration into the plugin would work well, a web service provides better maintainability and is more easily distributed. One of the biggest issues with the plugin strategy is that it would require heavy maintenance and GUI development for every browser that would be supported. Since available proxies, user-agent strings, and operating systems would likely change often, it would be easiest to manage this through a web service that could be updated and enhanced without having to make significant updates to client-side tools.

While moving the configuration to a web service removes a significant amount of functionality and maintenance from the client-side, a browser plugin still seems like the best approach for enhancing the client-side platform. Browser plugins are ideal for improving browsing experiences and providing easy portability, maintainability, distribution, and security. By piggy-backing on top of a browser, the system can utilize the browser's built-in functionality for handling websites, rendering images, and running code. Additionally, browser plugins allow the user to continue using the browsing environment they are familiar with instead of introducing them to a new, custom tool. Most browsers offer a rich API that allows functionality to be added quickly and easily. For example, a toolbar button could easily be added that would allow the multiplexing functionality to be turned on and off by simply clicking a button. Unless the multiplexing proxy was easy and pain-free to use, adoption of the proxy would likely fall short. Since multiplexing all HTTP requests is not desirable (for example, POST requests to a shopping cart), the system requires a button that could disable the proxy quickly.

Browser plugins and their affiliated APIs also provide a simple method of grouping together the resources for a particular website. By assigning an identifier to all resources fetched after opening a new website in the browser, the resources for a website can be analyzed together and displayed back to the user in a more concise and understandable form. Browser plugin APIs allow this information to be easily retrieved and passed to the proxy through a custom User-Agent header that can be stripped at the proxy.

Lastly, implementing the client-side as a browser plugin allows it to piggy-back on the functionality provided by other existing plugins. For example, combining the multiplexing proxy with Adblock Plus filters requests to common advertisement domains and tracking services (Adblock Plus). Because advertisements and tracking are heavily used and often change with each request, filtering them greatly enhances and cleans the analysis of websites. While it seems like our goal is partially diminished by this, in reality, the websites themselves typically have little control over the content returned by the embedded third party elements. Other plugins such as NoScript provide additional security to the end user (NoScript). Because the majority of malicious responses utilize JavaScript, it is common to disable JavaScript on untrusted websites. However, using a plugin like NoScript reduces the analysis that can be performed on resources that are obtained dynamically after a website loads in the browser.

Displaying the analysis in tab separate from the original request appears to be the best approach for in depth result analysis and visualization. While it would be possible to display differences by injecting information into the original response, this strategy has many shortcomings that could be easily mitigated with another approach. One major problem is that website content is now much more dynamic than it

used to be. Since many websites are no longer static, an area of the website that was flagged as containing tailored content may later be replaced by matching content through scripts, AJAX, and DHTML. This adds a temporal element that makes it difficult to easily highlight differences. Also, a significant amount of logic is contained in elements and code that is not visible to the user when browsing the website. This would make it both difficult to properly display to the user "inline", and would often confuse less technical users that would not understand some of the underlying syntax behind websites. While the analysis could also be displayed around the borders of the webpage, it would inhibit the user's view of the website and may make it difficult to always distinguish between the two.

Whenever the plugin detects the user visiting a new website, it automatically loads the analysis of that particular website in a toolbar. When requested, more detailed analysis is presented in a separate tab from the original content. This strategy allows the user to either view in depth analysis of the content, or continue browsing in a non-inhibited manner. The analysis is retrieved from a web service running parallel with the proxy that has access to the same data store containing the responses and analysis of the multiplexed requests. The web service uses the Google Web Toolkit (GWT). GWT allows the user to program in Java, but compile the client-side code into JavaScript that automatically handles inconsistencies between different browsers. Since the proxy was also being developed in Java, the GWT framework seemed like an appropriate choice.

The multiplexing system does not make all the multiplexed requests through the proxy itself. Instead, it relies on separate, distributed agents that receive commands from the proxy. These agents may either exist remotely or on the local machine running the proxy. As requests are generated at the multiplexing proxy, they are placed in a queue. Using a round-robin algorithm, these requests are forwarded to agents that have connected to the proxy system. The agent is responsible for completing the request, performing the initial analysis on the results, and inserting the results into the data store. This process may involve the agent forwarding the request through another external proxy, if necessary. The one exception to distributing the requests among the agents in a round-robin fashion is when operating system modifications are requested. At startup, each agent specifies the operating system TCP/IP stack they can handle. For these types of request modifications, the agents are iterated from least recently used (LRU) to most recently used (MRU) until one with the appropriate operating system is found. The distributed agent architecture was chosen because of the heavy amount of traffic going through the system. For each request that comes in, several requests must be created. If the system has many users concurrently making requests with large configurations, the distributed strategy will allow the system to more easily handle the

load. Since agents can be added or removed at any time, this makes the system scale in a very simple and efficient way.

When a response is received at an agent, initial analysis is performed before inserting it into the data store. This requires first examining the "content-encoding" response header to make sure the response does not need decompressed. After standardizing the response content, a combination of the "response-type" header and the content may be used to classify the response type. Since the header may be altered or blank, the content itself is analyzed to ensure it matches the header. For instance, if the "response-type" header specifies that it is an image, the first few bytes of the response are checked to make sure it matches the image type reported in the header. Similarly, strategies are used for detecting HTML, scripts, and many other content types that may be useful for analysis and filtering.

**RESOURCE CLASSIFICATION**

A heavy amount of dynamic web content does not necessarily increase the probability of tailored web content. If every response contains different content, it is more likely that it is the result of instantly dynamic content as opposed to tailored web content. This makes it essential to analyze the results according to the actual modifications that were made to the client fingerprint at the multiplexing proxy. For instance, if only three of ten responses were classified as dynamic, but they were all on User-Agent string modifications, this is much more telling than if every response returned dynamic content. In this case, the resource can immediately be classified as being tailored toward specific browsers, while the more dynamic resource remains ambiguous. The more dynamic case typically requires manual analysis through visualization tools.

Attempting to classify a website modification as either malicious or non-malicious is a failing approach. While this would be possible in obvious cases such as the addition of obfuscated JavaScript, the range of malicious activity is simply too vast. For example, changing one character in a URL could be the sole cause of an exploit being launched instead of something benign. Similarly, the hash of an image returned from a URL may change because the URL is to a dynamic advertisement, or because an image was formulated in a way to exploit a known vulnerability in the browser's image parser.

This problem is exacerbated from the point of view of an open source information analyst who is interested in the actual content on the website. A dynamic number in the content of a website could be

anything from a price change to simply text at the bottom of the page relaying how long building a response took or how many comments there have been on an article. Without being able to fully understand the surrounding content of the website, it is not possible to put a response modification in the appropriate context.

The problems outlined above provide the groundwork for classifying not on whether or not the modification was malicious, but instead classifying the type of dynamic content returned from various web pages. Firstly, it is possible to classify a website as containing instantly dynamic content by making several simultaneous requests with an identical client fingerprint and comparing the responses. If the responses to duplicate requests all match, but there are modifications to the responses of requests with an altered client fingerprint, then it is possible to deduce whether or not the changes were based on a modification to the browser, the operating system, or the location of the client. These changes can be classified as browser anomalies, operating system anomalies, and location anomalies, respectively.

Table 4, below, presents the set of classifications that can be assigned to each resource collection.

**Table 4: Classifications assigned to resource collections**

| Classification | Description |
|---|---|
| *Static* | No changes were detected |
| *Instantly Dynamic* | Changes were detected between requests with a duplicate fingerprint |
| *Browser Anomaly* | A change was attributed to a modified User-Agent string |
| *OS Anomaly* | A change was attributed to a TCP/IP stack modification |
| *Location Anomaly* | A change was attributed to the IP address of the requesting system |

The accuracy of the classifications depends heavily on the ability to correctly identify whether or not a resource is static or instantly dynamic based on identical, simultaneous requests. If we know that the same response is returned every time for that particular client fingerprint, then any change in the response to a modified client fingerprint allows for the immediate identification of a tailored resource. However, if every response to identical requests appears to be instantly dynamic, one can expect that the responses to modified client fingerprint requests will also not match the original response.

Because the correct identification of tailored versus instantly dynamic content relies almost solely on the ability to identify static resources through duplicate requests, this allows us to reduce the problem of classification accuracy to a more simple form. Essentially, accuracy will be heavily based on whether or

not a resource can be correctly identified as either static or instantly dynamic through simultaneous duplicate requests, regardless of what is returned when using a modified client fingerprint.

This approach will be explored through both theoretical and empirical analysis. Firstly, it should be possible to derive equations that provide a lower bounds on the probability of misclassification based on the number of duplicate requests used at the multiplexing proxy. It should then be possible to test these results by visiting a preselected set of websites using several multiplexing configurations. The number of misclassifications will be measured for each website by consecutively testing it with configurations using one, two, three, four, and five duplicate requests. This will provide a baseline for evaluating the necessary number of requests required to meet a user's desired accuracy level.

**RESPONSE CLASSIFICATION**

The most critical piece of the multiplexing system is its ability to obtain meaningful and actionable results from response comparisons. However, it must also be able to convey these results to the user in an easily readable and non-inhibiting way. Because of the ambiguities and complexities involved with dynamic web content, options must be provided to further explore detected changes in a fast and concise manner.

While the position and amount of changed text are two attributes that jump out as something to that would be high on the list of things to analyze they actually provides very little information with regards to malicious modifications. A malicious change can be as large as an executable hidden in obfuscated JavaScript, or it can be as small as a URL with one modified character. The malicious change may even be in a resource such as an image that cannot be analyzed for contextual meaning. Because of these difficulties, statistics about the change itself actually provides very little insight toward the analysis of response modifications. Instead, it is more appropriate to analyze the page as a whole and provide visual tools to help an analyst gauge the nature of the change manually.

One straightforward analytic is to simply determine whether the content of one response exactly matches the content of another response (a byte-for-byte comparison). However, repeatedly comparing large amounts of content can be a very costly. Calculating hash values for every response provides a fixed, small representation of a large, variable amount of data. When comparing two responses, it is sufficient to determine whether or not they have identical hash values. An MD5 checksum provides a sufficient 128 bits of entropy. This gives negligible probability of collision while providing a quick method of comparing two responses.

The dynamic nature of the web creates problems when using a simple hash function to determine whether or not the same content was returned. Many of the frameworks and content management systems inject unique identifiers into elements in many of the responses. While these unique identifiers do not modify the visible content or behavior of the website, the hash values will not match. To solve this problem, another analytic involves calculating a hash value over a cleaned and stripped version of the website using the Jericho HTML Parser (Jericho HTML Parser). This analytic strictly looks at the content that should have a visible impact to the user. For example, if a website injects tracking information into the "id" field of an HTML element, this will be stripped out of the response before the hash value is calculated. This provides the user with both a true content hash value and a stripped content hash value. When the original hash values do not match, it is possible to compare whether or not the stripped hash values of responses match. If they do, it is likely that the changes detected in the responses do not contain visible content that was tailored based on the client fingerprint.

Comparing only the structure of the web responses provides a third analytic that compliments the first two. One major cause of instantly dynamic content is the inclusion of unique identifiers into URLs, scripts, and HTML element attributes. A website changing its DOM structure is typically representative of a larger change in the content of the website. Incorporating detection of changes in the DOM structure will help more accurately classify the type of change that is occurring as opposed to simply focusing on the discovery of a change itself. This can be accomplished by hashing the concatenation each element encountered in the DOM using a SAX parser. An alternative approach would be comparing XPath value calculations for each element.

Yet another useful analytic is to look at the overall length of the responses. While the length of specific modifications themselves typically provides very little information, the overall change in length can be insightful. The most difficult case to handle is when there is both instantly dynamic and tailored content in the same response. While some of the responses for that particular resource may only contain the instantly dynamic data, some of the responses may also have information that was tailored to a modified client fingerprint. An interesting approach to discovering this type of resource is to search for outliers in the response lengths. While the length for all the websites may be different, it is likely that a significantly tailored resource's length will greatly differ when compared to the resources that were simply instantly dynamic. This approach works well because it applies to all resource types as opposed to a specific one such as HTML or scripts. Additionally, it is less computationally expensive than measurements such as

word distance. It is both a quick and accurate measurement that can be computed across all types of responses.

Each of these analytics can be computed at a very low cost. Firstly, the length is immediately known after receiving a response, making it an ideal attribute for simple analytics. Additionally, the byte-to-byte comparison and stripped comparison can be performed by simply calculating a hash value over the content and stripped content, respectively. This makes it unnecessary to do deep inspection of individual elements or differences, saving this type of analysis until it is considered necessary by the user. Each of these calculations can be stored in the database and be easily reused without having to perform the calculations on actual responses again. This significantly reduces the necessary cost and complexity in the system.

Using the response analytics described above, it is possible to determine whether or not modifying the client fingerprint has a significant impact on the content. Table 5, below, details the possible ways a response can be classified after being compared to the responses of the original and duplicate requests.

**Table 5: Classifications assigned to each multiplexed response**

| Classification | Description |
|---|---|
| *Matching* | The response matched the original response in a byte-to-byte comparison |
| *Stripped Matching* | The response's visible user content matched the original |
| *Not Matching* | The two responses were not matching or stripped matching |
| *Length Anomaly* | The response had an anomalous length compared to the original and duplicate responses |
| *Failed* | The server refused to serve the client request |

The varying response classifications are critical for determining whether content is instantly dynamic or tailored toward a particular client fingerprint. Responses to duplicate requests provide a baseline that can be used for comparing the responses of requests with a modified client fingerprint. The classifications in table 5 provide a hierarchy for determining how anomalous a particular response is compared to the rest. Naturally, a request that failed must be considered the most anomalous. If a response was received, the next most interesting case is a significant length anomaly. This is followed in the hierarchy by a response that differs from the original response in some way. Slightly lower in the hierarchy is a response that matches the original response when stripping out benign content not visible to the end user. Lastly, if

none of these classifications accurately describe the response, it must match the original response byte-for-byte.

Tailored responses are found by identifying the responses that are classified at least one step higher in the hierarchy than the highest classified response to a duplicate request. This approach identifies anomalous responses despite dynamic website behavior. For instance, if every response to duplicate requests comes back as not matching the original response (they contain instantly dynamic content), tailored content can still be identified through a response that additionally has an anomalous length in comparison to the original and duplicate responses.

A website consists of not only the initial web pages, but also the resources necessary for rendering that page. Therefore, it makes sense to display the analysis as a collection of resources from the visited site that are ranked according to their dynamism and classification. Providing quick access to the most "interesting" content allows the user to further explore the results using one of the many visualization tools provided by the multiplexing proxy web service.

Firstly, the user can view the modified content stripped away from the static content (the differences). However, because context is often important, the user may also view the highlighted modifications in line with the original response. Yet another way of viewing the differences is to only look at the inline differences of stripped content, only showing text that would have been visible to the end user. Additionally, compressed and malformed HTML is mitigated by providing a cleaned and formatted version of the text.

Instead of using one of the structured "diff" algorithms seen in previous research, an unstructured algorithm was selected due to the inconsistency and abuse of many MIME types. By using an unstructured "diff" algorithm, the same algorithm could be applied to any textual response no matter what the actual type of the resource was. This provided a widely usable algorithm that did not require MIME type identification and verification.

## ROLE ANALYSIS

The primary objective of this study is to determine the usefulness of the multiplexing proxy for open source analysts, cyber analysts, and reverse engineers. While each of these roles would utilize the tool for

a different purpose, they all share a common interest in detecting tailored web content toward a specific client fingerprint.

**Open Source Analyst**

The multiplexing proxy helps an open source analyst assign a level of trust and validity in the information obtained during information collection. This assumption can be tested by using the proxy to visit several websites in varying genres. Shown in table 6, eight genres have been chosen for testing the classification accuracy of the multiplexing proxy.

**Table 6: Website genres for open source collection analysis**

| Genre | Description |
|---|---|
| *News / News Aggregate* | Reputable websites that provide news or links to news |
| *Business* | Websites belonging to a particular business |
| *Personal* | Websites belonging to an individual without a business focus |
| *Blog* | Corporate or individual websites that take the form of a blog |
| *Forum* | Websites based on threads of information posted from various users |
| *Search Engine* | Websites that provide a broad website search functionality to users |
| *Reference/Image/Video* | Websites that provide resources and information on a narrow topic |
| *Other* | Websites that did not fit well in other genres (i.e. social networking) |

Organizing the websites into specific genres will provide the ability to analyze patterns in the classification distributions for each particular genre. The tool will be most effective for genres that have a minimal amount of instantly dynamic content. Non-static classifications will be investigated and documented for further analysis. This will closely mimic the necessary steps that would be taken by an open source analyst using the multiplexing proxy.

In order for the tool to be considered useful for the open source analyst role, the information and trust gained must outweigh the additional effort and time necessary to analyze the results provided by the multiplexing proxy. By running a series of tests over many genres of information and many types of websites, it will be evident whether or not the additional complexity and time requirements are offset by the amount of information and trust a user gains.

**Cyber Analysts**

Cyber analysts are able to evaluate malicious websites with much greater efficiency using the multiplexing proxy. By visiting a website with many varying fingerprints, it is possible to trigger malicious behavior that may not have been seen, otherwise. However, demonstrating this capability cannot be accomplished easily. Firstly, obtaining a database of live, malicious websites is very difficult. Secondly, there is significant risk in rendering malicious websites in a browser without a robust system involving virtual machines. Because of these issues, mock websites will be built that demonstrate what the user would see when the techniques outlined in table 7 are used.

**Table 7: Exploitation techniques**

| Technique | Description |
|---|---|
| JavaScript Injection | Inject malicious JavaScript into a previously benign response |
| I-Frame Injection | Introduce an I-Frame that points to a malicious domain |
| URL Modification | Modify a URL to point to a malicious domain |
| Maliciously Crafted Image | An image modification that exploits a specific browser vulnerability |
| Multiple Vulnerabilities | A combination of multiple vulnerabilities from above |

Each technique will be demonstrated by modifying an existing website with a spoofed implementation of each of the above techniques. The website will be visited and analyzed with the multiplexing proxy to show the classifications and visualizations available in each of the above scenarios.

**Reverse Engineers**

The multiplexing proxy allows reverse engineers to easily analyze both instantly dynamic and tailored content. This provides insights into server-side behavior that are often not easily obtainable without the source code. Minor anomalies between web responses are difficult and time-consuming to find without a tool that can quickly analyze multiple responses simultaneously. The utility of the proxy can be demonstrated by analyzing and investigating websites collected for the classification accuracy study. Tailored and instantly dynamic resources will be analyzed to determine whether extracting and visualizing the dynamic content provides valuable insight or unnecessary overhead. The utility of the tool is dependent on the value of the information that can easily be extracted with the tool. If the resources classified as instantly dynamic or tailored frequently provide no useful information, then the multiplexing

proxy cannot be considered an effective tool for this role. However, if the multiplexing proxy identifies interesting properties of the website not easily seen without the tool, it will have proven its usefulness.

## METHODOLOGY ANALYSIS

Fully analyzing the potential of the multiplexing proxy requires testing not only its potential to provide descriptive and accurate resource classifications, but also its ability to provide information to its users that is not more easily obtained through a simpler approach. Outlined above, this study will:

1) Produce a set of resource classifications appropriate for real-time analytics
2) Quantify the accuracy of the resource classifications
3) Determine what types of websites are most affected by instantly dynamic or tailored content
4) Develop visualization tools for analyzing dynamic web content
5) Examine the usefulness of the tool for open source analysts, cyber analysts, and reverse engineers.

Each of these steps will provide additional information or tools that were not previously available. This work will provide guidance for future research in this area. By identifying the roles that benefit the most from this type of tool, future efforts can be directed toward the areas benefiting the most from the tool.

# LIMITATIONS

The first limitation of the multiplexing proxy in its current form is the lack of support for secure HTTP (HTTPS). Because the proxy acts as a middle-man between the start and end destinations, it is unable to view or modify any of the encrypted Transport Layer Security (TLS) or Secure Socket Layer (SSL) traffic between the two end systems. In order to accomplish this, the proxy would have to hijack the handshake using its own X509 certificate. Once the secure HTTP request was received at the proxy, it could be decrypted and analyzed. At this point, the proxy would then open new SSL/TLS connections with the end host, completing secure transactions on behalf of the client. However, this is a large privacy risk and is not a necessary step for research-oriented work.

Secondly, the multiplexing proxy does not currently operate on HTTP POST requests. Multiplexing POST requests can have many adverse effects. For example, POST requests are often used for adding items to a shopping cart or submitting a comment to a forum. If the functionality of the POST request is not idempotent, it will cause unintended actions on behalf of the user. Because many users may not be aware of the consequences of a multiplexed POST request, it may be safest to simply remove this functionality.

Another limitation is the lack of analysis based on HTTP cookies. A beneficial extension to the experiment would be running tests both with and without site-specific cookies to analyze how it affects the dynamism of the website. In the multiplexing proxy's current form, all the cookies sent in the original request are reused in the modified client requests. This would theoretically allow session-tracking across multiplexed requests. While this maintains consistency with only modifying one fingerprintable attribute with each request, it would also be interesting to see how removing the site-specific cookies affect the responses.

Lastly, the ability to detect and modify operating system fingerprints is currently not robust. With the current implementation, I have the knowledge of what operating system is connecting to the proxy, and can exploit that knowledge by not requiring the proxy to fingerprint the client. A large scale system that supports many clients would require additional fingerprinting at the proxy so that the requests could all be multiplexed appropriately. Additionally, a much more robust network of computers would be required to support the necessary number of operating systems. This also applies to the remote proxies where the operating system fingerprint cannot currently be controlled. An alternative approach to routing through multiple operating systems is to integrate the agents with a tool such as IpMorph that can dynamically

alter the system's fingerprint in user space depending on the needs at the time. The end use and deployment of the system would deem which of these two strategies was most appropriate.

# RESULTS

Quantitative and qualitative analysis revealed many strengths and weaknesses of the multiplexing proxy. Both the theoretical and empirical classification accuracy studies provided sufficient evidence to warrant future work in this area. However, fully utilizing the tool in its current form requires the user to possess a technical background. When the user comprehends the results without confusion, the multiplexing proxy is a truly powerful tool. However, a user with little knowledge about the web and languages behind it may get easily confused and misled by the results. While the proxy provided significant utility for many tasks, it was also evident that more work will be required before it is more widely usable in a production environment. Despite this, it has proven to be a unique and invaluable tool that provides information and assurance that has not previously been easily attainable.

## THEORETICAL CLASSIFICATION ACCURACY

Classifying websites accurately depends on the ability to distinguish between instantly dynamic web content and tailored web content. This is most easily accomplished by creating duplicate requests at the multiplexing proxy. Without duplicate requests, it is not possible to gain a ground truth of whether or not the web server returns as static response. If the original response matches the responses for each of the duplicate requests, it increases the probability that the particular web resource is static for that particular fingerprint. This allows us to appropriately classify changes seen in the responses for other client fingerprints. However, if the multiplexing proxy does not send any duplicate requests to a web server with dynamic content, responses will likely be seen that both match and vary from the original response. This will falsely give the impression that responses were tailored toward a specific client fingerprint instead of being instantly dynamic.

The two major cases of misclassification involve classifying an instantly dynamic website as either static or tailored toward a specific browser. Both of these cases occur when, by chance, all the duplicate request responses match the original response. When this happens, if the responses for a varying client fingerprint match the original response, the resource will be classified as static. Otherwise, it will be classified as tailored.

**Probability of Misclassification**

Say, for example, the multiplexing proxy is making three duplicate requests and two modified requests to a web server that returns one of three stochastically independent web responses with probabilities .50, .25, and .25, respectively. Misclassifying an instantly dynamic website as tailored would require all three responses to duplicate requests to match the response to the original request. If this happens, the website will either be misclassified as static or tailored toward a specific fingerprint. This means that to have a chance of misclassification, it would require either the response version with a 50% chance of being returned to be chosen four times in a row, or one of the two response versions with a 25% chance of being returned being chosen four times in a row.

Let *P(M)* be the probability that all the duplicate request responses match the original response:

Then:

$$P(M) = .5^4 + .25^4 + .25^4 = .0703$$

For the resource to be incorrectly, classified as static, the two responses to varying client fingerprints much also match the original response. This equates to having two more responses that much match the original and duplicates.

Let *P(S)* be the probability that an instantly dynamic web resource will be incorrectly classified as static:

Then:

$$P(S) = .5^6 + .25^6 + .25^6 = .0161$$

Alternatively, if one of the two varying client fingerprint responses does not match, then it will result in a tailored classification instead of an instantly dynamic classification.

Let *P(T)* be the probability that at least one instantly dynamic web resource will be incorrectly classified as tailored. This can be obtained by taking the probability of the original and duplicate requests matching and multiplying this with the difference between one and the probability of both varying client fingerprint responses matching.

Then:

$$P(T) = .5^4(1 - .5^2) + .25^4(1 - .25^2) + 25^4(1 - .25^2) = .0542$$

It is straightforward to see that $P(M) = P(S) + P(T)$. Given two, it is possible to calculate the third through simple arithmetic.

While the values calculated above seem high, it should be noted that many instantly dynamic websites will not be limited to only a few different responses with high probability. Instead, many sites will have a unique response for every request, effectively pushing the probability of the same response being returned near 0.

Although the calculations above were made for a specific multiplexing proxy configuration, they can be generalized to account for other configurations.

When the membership function of an element in a set is an indicator function that returns an integer [0,1], it holds that the probability of an event occurring is equal to the expected value of that particular event. Since the expected value is defined as the weighted average of all possible values of a random variable, we have that the probability of an event occurring is equal to this weighted average.

Let $P(X)$ be the probability of an event X occurring;
Let $E[X]$ be the expected value of the random variable X;
Let $p_i$ be the probability of the i'th unique value
Let $x_i$ be the value of random variable X seen with probability $p_{i;}$

If membership is well-defined under an appropriate indicator function:

Then:

$$p_1 + p_2 + \ldots + p_k = 1.$$

And:

$$E[X] = P(X) = x_1p_1 + x_2p_2 + \ldots + x_kp_k$$

Let $R$ be the number of unique responses that may be returned by a website;
Let $D$ be the number of duplicate requests sent;
Let $P(i)$ be the probability of the web server returning i'ith unique response;
Let $P(M)$ be the probability of all the responses to duplicate requests matching the original response;

Then:

$P(M)$ is equal to the weighted average of all the responses matching the response to the original request for each possible unique response

Or:

$$P(M) = \sum_{i=1}^{R} P(i) \wedge (D + 1)$$

Note that $P(i)$ is raised to the $D+1$ power to give us $P(m_i)$, the probability of the same particular response being returned in both the original request and each of the duplicate requests.

Because most websites return different versions randomly and in an equally probable manner, a discrete uniform distribution can be expected during sampling. In this case, the weighted average turns into a simple average, and the equation simplifies to the following:

$$P(M) = 1/(R \wedge D)$$

Table 8, below, provides the probability of all duplicates matching the original response for many values of *R* and *D* (assuming the server returns each response with equal probability).

**Table 8: *P(M)* for varying values of R and D**

| P(M) | | R | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **D** | **1** | 1 | 0.5 | 0.3333 | 0.25 | 0.2 | 0.1667 | 0.1429 | 0.125 |
| | **2** | 1 | 0.25 | 0.1111 | 0.0625 | 0.04 | 0.0278 | 0.0204 | 0.0156 |
| | **3** | 1 | 0.125 | 0.0370 | 0.0156 | 0.008 | 0.0046 | 0.0029 | 0.0020 |
| | **4** | 1 | 0.0625 | 0.0124 | 0.0039 | 0.0016 | 0.0008 | 0.0004 | 0.0002 |
| | **5** | 1 | 0.0313 | 0.0041 | 0.0010 | 0.0003 | 0.0001 | 5.95E-5 | 3.05E-5 |
| | **6** | 1 | 0.0156 | 0.0014 | 0.0002 | 0.0001 | 2.14E-5 | 8.5E-6 | 3.81E-6 |
| | **7** | 1 | 0.0078 | 0.0005 | 6.1E-5 | 1.28E-5 | 3.57E-6 | 1.21E-6 | 4.77E-7 |
| | **8** | 1 | 0.0039 | 0.0002 | 1.53E-5 | 2.56E-6 | 5.95E-7 | 1.73E-7 | 5.96E-8 |
| | **9** | 1 | 0.0020 | 5.08E-5 | 3.81E-6 | 5.12E-7 | 9.92E-8 | 2.48E-8 | 7.45E-9 |
| | **10** | 1 | 0.0010 | 1.69E-5 | 9.54E-7 | 1.02E-7 | 1.65E-8 | 3.54E-9 | 9.31E-10 |

With the exception of R=1, the *P(M)* values represent the probability that instantly dynamic content will be classified as either static or tailored.

The web resource will be incorrectly classified as static when the multiplexed requests with a varying fingerprint also have matching responses. However, if the varying fingerprint returns a different response, the web resource will then be classified incorrectly as being tailored toward that particular fingerprint.

The number of duplicate request responses matching the original response can be modeled as a series of binary outcomes, with D representing the number of trials in a binary distribution. The only case that will cause a misclassification is when all the trials are consistent with the original response.

Let *V* be the number of varying client fingerprint requests other than the original request or duplicates;
Let *P(S)* be the probability that an instantly dynamic web resource will be incorrectly classified as static;

Then:

$$P(S) = \sum_{i=1}^{R} P(i)^{(D + V + 1)}$$

Notice that this is the same formula as P(M), but with the addition of varying client fingerprints to the calculation of $P(m_i)$.

If the probability of each possible response is equal, then it simplifies to the following:

$$P(S) = 1/(R \wedge (D + V))$$

Alternatively, if one of the two varying client fingerprint responses does not match, then it will result in a tailored classification instead of an instantly dynamic classification.

Let *P(T)* be the probability that at least one instantly dynamic web resource will be incorrectly classified as tailored. Once again, this can be obtained by taking the probability of the original and duplicate requests matching and multiplying this with the difference between one and the probability of the varying client fingerprint responses all matching.

Then:

$$P(T) = \sum_{i=1}^{R} ((P(i))^{(D + 1)} * (1 - (P(i)^{V}))$$

If the probability of each possible response is equal, then it simplifies to the following:

$$P(T) = P(M) * (1 - 1/(R \wedge V))$$

Or more explicitly:

$$P(T) = 1/(R \wedge D) * (1 - 1/(R \wedge V))$$

**Simultaneous Instantly Dynamic and Tailored Data**

Another possibility for incorrect classification is seen when a set of web responses contains both instantly dynamic data and tailored data simultaneously. In this scenario, multiple responses to duplicate requests may contain unique content, influencing the classifier to mark differences in the modified fingerprint responses as being the result of instantly dynamic content instead of tailored content.

Let *P(i)* be the probability that the i'th response matches the response to the original request
Let *P(inst)* be the probability that a resource will be classified as instantly dynamic;
Let *P(tail)* be the probability that a user will receive a tailored response for a request with a varying client fingerprint;
Let *P(misc)* be the probability of misclassification due to both instantly dynamic data and tailored content;

Then:

$$P(inst) = (1 - P(M))$$

And:

$$P(tail) = 1 - \prod_{i=1}^{V} (1 - P(i))$$

Finally:

$$P(misc) = P(inst) * P(tail)$$

Or more explicitly:

$$P(misc) = (1 - P(M)) \left(1 - \prod_{i=1}^{V} (1 - P(i))\right)$$

It should be noted that in many cases, *P(tail) = 1*. Because of this, the probability of a misclassification in this scenario is significantly higher than was seen for misclassifications due to all the duplicate responses

matching. This enforces the necessity of additional measures to determine whether or not a resource is tailored regardless of classifying the response as instantly dynamic.

### Summary

The formulas derived above calculate the theoretical probability of misclassification for resources both containing and not containing instantly dynamic data. These formulas show that sufficient accuracy can be achieved by configuring the proxy to use multiple duplicate requests.

Greater accuracy can be gained by also running calculations off of a stripped version of the content. Because a significant amount of instantly dynamic content is found in comments, markup, and scripts, it is possible to distinguish significant changes in tailored responses from insignificant changes due to instantly dynamic content by attempting to strip out the less relevant content before comparisons.

Additionally, it is possible to perform comparisons between the differences in each of the web responses to obtain a measure of how "anomalous" the change is compared to others. This could be done using many different techniques such as looking at the number of differences, the TF-IDF Cosine distance, the word distance, or even simply the overall size of the response. By looking at the distribution of the values across all the web responses, it should be possible to identify outliers that do not seem congruent with the rest of the responses, further increasing the accuracy of the classifications.

## EMPIRICAL CLASSIFICATION ACCURACY

While the theoretical accuracy of the classification model was calculated above, it assumes a ground truth about the responses returned from the website. In practice, this will be an unknown.

Similar to the theoretical classification accuracy, the empirical accuracy relies primarily in the ability to distinguish between static and instantly dynamic content through the use of duplicate requests. If instantly dynamic content is not detectable before looking at the responses for modified client fingerprints, it is not possible to tell the difference between tailored content and instantly dynamic content. Additionally, detecting tailored content is dependent on using the exact client fingerprint that would trigger the tailored

response. Therefore, empirical analysis will concentrate on the misclassification of instantly dynamic content.

In order to empirically examine the potential for misclassification, a large sample of popular websites was taken from the Alexa Top Sites list (Top Sites). The most popular websites were divided into eight categories: *News/News Aggregate, Business, Personal, Blog, Forum, Search Engine, Reference/Image/Video,* and *Other*. This gave an unbiased baseline that could be used to measure the ephemeral properties of popular websites. The categories that did not contain many domains at the top of Alexa's list were supplemented with a relatively small number of domains found by using results from search engine queries for that particular category, as well as a few personally known sites.

Social media has greatly affected how websites dynamically integrate with one another. Because of the popularity of sites such as Facebook and Twitter, it has now become common practice for major websites to add resources to web pages that allow users to bridge the gap between the website and the social media tools. Social media widgets are often instantly dynamic, slightly modified with each response.

Additionally, many high traffic websites collect revenue by integrating ads into their websites using a cross-domain ad service. Unfortunately, these ads are often instantly dynamic. Many tools are available that block the most popular ad services. AdBlock Plus, for example, is a popular browser plugin for Firefox that helps block the most popular ad services. Often, the site using the ad service does not have control of the ads that are displayed to the end user. Because of this, Ad Block Plus is used during testing to prevent ads from skewing the results for each website.

Lastly, an aspect of the web that greatly affects the dynamic nature of websites is visitor analytics. Many websites make requests to resources at domains known to collect information on visitors. Requests collecting user statistics can most easily be identified through both domain name analysis, as well as through a ping-like behavior that is often attributed to user-tracking.

Social media, ad services, and visitor analytics all significantly contribute to the amount of dynamic content coming from a website. For example, tools such as Google Analytics, Facebook Connect, and AdSense are commonly found while browsing the web. However, since many websites do, in fact, rely on cross-domain information, it is not sufficient to simply ignore all cross-domain requests. Therefore, for this study, only cross-domain requests to known, popular tracking, social media, and ad service tools were

filtered from the results. This mirrors the strategy of filtering based on a white list of the most commonly integrated tools. This was done through a combination AdBlock Plus and manual filtering.

Table 9, below, provides a high level overview of the data collected for empirical analysis.

**Table 9: Website Categories**

| Category | Number of websites | Number of Resources | Instantly Dynamic Resources |
|---|---|---|---|
| News / News Aggregate | 22 | 1821 | 35 |
| Business | 44 | 2428 | 37 |
| Personal | 12 | 226 | 2 |
| Blog | 16 | 950 | 9 |
| Forum | 11 | 327 | 11 |
| Search Engine | 12 | 187 | 10 |
| Reference/Image/Video | 18 | 987 | 21 |
| Other | 15 | 704 | 19 |
| **TOTAL** | **150** | **7630** | **144** |

As one can see, there was just under an average of one instantly dynamic resource per website after filtering analytics, social media, and ad services. Websites categorized as *News/News Aggregate*, *Business*, *Reference*, and *Other* tended to have more dynamic content than the average website. This correlates with past research saying that larger, frequently visited websites tend to be more dynamic than other sites.

All 150 websites were analyzed five times. Initially, the multiplexing proxy was configured to multiplex and analyze the original request along with one duplicate request. The number of duplicate requests was increased with each successive test up to 5 duplicate requests. The multiplexing proxy classified each resource as either static or instantly dynamic. The results were manually analyzed to help provide a ground truth and ensure accuracy of the tool.

**Overall Statistics**

The classification accuracy of the multiplexing proxy was directly proportional to the number of duplicate requests made. Table 10, below, provides classification statistics from combining the data sets for each of the eight categories.

**Table 10: Overall Classification Statistics (7630 resources)**

| # Duplicate Requests | Static | Instantly Dynamic | Misclassified | % Error |
|---|---|---|---|---|
| 1 | 7509 | 121 | 23 | 0.301 |
| 2 | 7499 | 131 | 13 | 0.170 |
| 3 | 7488 | 142 | 2 | 0.0262 |
| 4 | 7488 | 142 | 2 | 0.0262 |
| 5 | 7487 | 143 | 1 | 0.0131 |

The results in table 10 show that there is significant improvement in the classification accuracy as more duplicate requests are used. Adding a second duplicate request nearly halved the number of misclassifications, and using three duplicate requests lowered the initial 23 misclassified resources down to only two misclassified resources. However, using more than three duplicate requests appeared to have diminishing returns. While four duplicate requests showed no improvement over three duplicate requests, there was one resource that was identified using 5 duplicate requests that was not caught when using four duplicate requests.

The diversity between different types of websites makes a strong case for analyzing each of the particular genres separately. The following sections go into the specific details of the test results for each individual categorization of website. While the exact categorization of many websites is ambiguous and debatable, a best effort was made to place each website in its most relevant category.

**News / News Aggregate Websites**

Websites that fell into the *News/News Aggregate* tended to be heavier in resources than many other categories. This is partly attributed to the large number of images and scripts often seen on this type of site. Collecting data from 22 websites resulted in 1821 resources, 35 of which were classified as instantly dynamic.

**Table 11: Analysis of News/News Aggregate Websites**

| Website | Resources | Number of Detected Instantly Dynamic Resources | | | | | Actual |
|---|---|---|---|---|---|---|---|
| | | 1 Dup | 2 Dup | 3 Dup | 4 Dup | 5 Dup | |
| news.yahoo.com | 40 | 3 | 3 | 3 | 3 | 3 | 3 |
| news.google.com | 34 | 2 | 2 | 2 | 2 | 2 | 2 |
| www.bbc.co.uk/news | 90 | 0 | 0 | 1 | 1 | 1 | 1 |
| www.cnn.com | 96 | 0 | 0 | 0 | 0 | 0 | 0 |
| cnet.com | 130 | 1 | 1 | 1 | 1 | 1 | 1 |
| www.nytimes.com | 80 | 1 | 1 | 1 | 1 | 1 | 1 |
| www.huffingtonpost.com | 88 | 1 | 1 | 1 | 1 | 1 | 1 |
| www.digg.com | 31 | 1 | 1 | 1 | 1 | 1 | 1 |
| www.reddit.com | 35 | 1 | 1 | 1 | 0 | 1 | 1 |
| www.foxnews.com | 146 | 0 | 0 | 2 | 2 | 2 | 2 |
| www.reuters.com | 120 | 5 | 5 | 5 | 5 | 5 | 5 |
| time.com/time | 97 | 1 | 1 | 1 | 1 | 1 | 1 |
| www.latimes.com | 82 | 0 | 0 | 0 | 0 | 0 | 0 |
| www.forbes.com | 126 | 1 | 1 | 1 | 2 | 2 | 2 |
| www.wired.com | 79 | 0 | 0 | 0 | 0 | 0 | 0 |
| www.gizmodo.com | 58 | 3 | 3 | 3 | 3 | 3 | 3 |
| www.nydailynews.com | 111 | 2 | 2 | 2 | 2 | 2 | 2 |
| www.gawker.com | 47 | 3 | 3 | 4 | 4 | 4 | 4 |
| www.businessweek.com | 75 | 1 | 0 | 1 | 1 | 1 | 1 |
| www.cbsnews.com | 132 | 0 | 2 | 2 | 2 | 2 | 2 |
| www.npr.org | 77 | 0 | 0 | 0 | 0 | 0 | 0 |
| www.msn.com | 47 | 1 | 1 | 2 | 2 | 2 | 2 |

Averaging over 1.5 instantly dynamic resources per website, News/News Aggregate websites contained more instantly dynamic content than any other category. Additionally, this category contained the largest amount of filtered content. The majority of the websites contained heavy amounts of social media, tracking scripts, and ads that were not calculated into the results above. Since these are often instantly

dynamic, including these resources would have greatly increased the number of instantly dynamic resources reported for this particular category of website.

Studying the misclassified resources presents a common pattern: the number of correctly identified instantly dynamic resources rises as the number of duplicate requests increases. The resources that were misclassified only using one or two requests are often correctly identified as instantly dynamic once the number of duplicate requests increases to three or more. However, as can be seen with *www.businessweek.com*, it is possible that a resource will be correctly classified as instantly dynamic with one duplicate request, but will be incorrectly classified with two duplicates. Additionally, while *www.reddit.com* correctly classified every resource with one, two, three, and five duplicate requests, it had incorrectly classified a resource as static when using four duplicate requests. This is one of only two cases where using four duplicate did not successfully classify all resources.

**Business Websites**

Similar to the *News/News Aggregate* category, *Business* websites had more false positives than most categories. Data from 44 websites resulted in 2428 resources, 37 of which were classified as instantly dynamic.

**Table 12: Analysis of Business Websites**

| Website | Resources | Number of Detected Instantly Dynamic Resources | | | | | Actual |
|---|---|---|---|---|---|---|---|
| | | 1 Dup | 2 Dup | 3 Dup | 4 Dup | 5 Dup | |
| **www.microsoft.com** | 63 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.conduit.com** | 43 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.apple.com** | 38 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.aol.com** | 54 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.mozilla.com** | 10 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.adobe.com** | 62 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.godaddy.com** | 45 | 1 | 0 | 1 | 1 | 1 | 1 |
| **www.skype.com** | 63 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.walmart.com** | 65 | 2 | 3 | 3 | 3 | 3 | 3 |
| **www.partypoker.com** | 54 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.dell.com** | 32 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.usps.com** | 90 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.hp.com** | 37 | 1 | 0 | 1 | 1 | 1 | 1 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **www.ups.com** | 12 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.pconline.com** | 26 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.avg.com** | 55 | 1 | 1 | 1 | 0 | 0 | 1 |
| **www.babylon.com** | 25 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.target.com** | 87 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.nfl.com** | 49 | 1 | 1 | 0 | 1 | 1 | 1 |
| **www.opendns.com** | 22 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.nba.com** | 117 | 0 | 1 | 1 | 1 | 1 | 1 |
| **plentyoffish.com** | 31 | 1 | 2 | 2 | 2 | 2 | 2 |
| **match.com** | 57 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.salesforce.com** | 42 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.amazon.com** | 73 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.ebay.com** | 38 | 2 | 2 | 2 | 2 | 2 | 2 |
| **www.groupon.com** | 27 | 2 | 1 | 2 | 2 | 2 | 2 |
| **www.fedex.com** | 10 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.comcast.com** | 63 | 1 | 0 | 1 | 1 | 1 | 1 |
| **www.expedia.com** | 24 | 1 | 1 | 1 | 1 | 1 | 1 |
| **barnesandnoble.com** | 128 | 0 | 0 | 0 | 0 | 0 | 0 |
| **samsung.com** | 153 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.verizon.com** | 23 | 1 | 2 | 2 | 2 | 2 | 2 |
| **www.ibm.com** | 56 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.newegg.com** | 119 | 0 | 1 | 1 | 1 | 1 | 1 |
| **www.att.com** | 65 | 1 | 2 | 2 | 2 | 2 | 2 |
| **www.themeforest.com** | 80 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.mysql.com** | 45 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.dyndns.com** | 9 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.oracle.com** | 60 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.gap.com** | 51 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.toysrus.com** | 103 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.costco.com** | 76 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.southwest.com** | 46 | 0 | 0 | 0 | 0 | 0 | 0 |

Both www.avg.com and www.nfl.com had interesting misclassifications. In the case of www.nfl.com, every resource was correctly classified with both one and two duplicate requests, but failed to appropriately classify a resource as instantly dynamic with three requests. Even more improbable was www.avg.com being correctly classified with one, two, and three duplicate requests, but being incorrectly classified with both four and five requests. This was very anomalous, and resulted in the only misclassification with five duplicate requests.

Like News/News Aggregate category websites, Business websites can benefit from having dynamic resources that change when users revisit the website almost instantly. A good example of this is having an inline ad that displays one of the popular products of the business. There is little incentive to display the same product to a customer every time, so it makes since to scroll between a few, popular products. Unfortunately, this is the type of resource that is most likely to be misclassified when a low number of duplicate queries are used. This can be seen in the high number of misclassifications under one and two duplicate requests.

**Personal Websites**

While *Personal* websites have historically been simplest in terms of the number of resources and dynamic content, the growing trend of content management systems and sophisticated website builders is quickly affecting this category. Despite this, websites that fall under the *Personal* website category still identified as the most simple of all the categories. It must be noted, however, that personal blogs were grouped into the *Blog* websites category, not making a strong appearance under *Personal* websites. Over 12 personal websites, 226 resources were seen with a total of 2 being classified as instantly dynamic.

**Table 13: Analysis of Personal Websites**

| Website | Resources | Number of Detected Instantly Dynamic Resources | | | | | Actual |
|---|---|---|---|---|---|---|---|
| | | 1 Dup | 2 Dup | 3 Dup | 4 Dup | 5 Dup | |
| **www.kentonborn.com** | 15 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.youdzone.com** | 22 | 0 | 0 | 0 | 0 | 0 | 0 |
| **people.cis.ksu.edu/~xou** | 5 | 0 | 0 | 0 | 0 | 0 | 0 |
| **cis.ksu.edu/~dan** | 8 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.eece.ksu.edu/~caterina** | 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.cricketwalker.com** | 34 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.ellenspace.net/marlyn.html** | 10 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.mintran.com** | 25 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.stuarthobday.com** | 33 | 0 | 0 | 0 | 0 | 0 | 0 |
| **colly.com** | 26 | 1 | 1 | 1 | 1 | 1 | 1 |
| **adellecharles.com** | 14 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.brandwood.com** | 30 | 0 | 0 | 0 | 0 | 0 | 0 |

Very few instantly dynamic resources appeared across the set of *Personal* websites. Additionally, there were no misclassifications across the resources. Within this category of website, it is uncommon to have

an instantly dynamic resource with only a few variations. Typically, any instantly dynamic resources on a personal website are a form of widget that contains a unique component in every response. For example, *www.cis.ksu.edu/~dan* has a weather widget that returns instantly dynamic content in every response. This makes it very simple to identify the instantly dynamic content with only one duplicate request.

### Blog Websites

Similar to *Personal* websites, *Blog* websites contained resources that were very simple to classify. Since many blogs use a popular content management system (CMS), it was important to ensure the test included an example using each of the popular frameworks. While some of the results in table 14 show the CMS investigated, the actual website that was used to explore the CMS appears below this. Collecting data from 16 websites resulted in 950 resources, 9 of which were classified as instantly dynamic.

**Table 14: Analysis of Blog Websites**

| Website | Resources | Number of Detected Instantly Dynamic Resources | | | | | Actual |
|---|---|---|---|---|---|---|---|
| | | 1 Dup | 2 Dup | 3 Dup | 4 Dup | 5 Dup | |
| **www.blogspot.com**<br>**nation-health.blogspot.com** | 71 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.wordpress.com**<br>**larissameek.com** | 85 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.livejournal.com**<br>**community.livejournal.com/ourbedrooms** | 36 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.smashingmagazine.com** | 43 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.schneier.com** | 9 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.crickwalker.com/blog** | 95 | 1 | 1 | 1 | 1 | 1 | 1 |
| **esbueno.noastokes.com** | 42 | 2 | 2 | 2 | 2 | 2 | 2 |
| **www.calicott.com** | 37 | 0 | 0 | 0 | 0 | 0 | 0 |
| **ma.tt** | 47 | 2 | 2 | 2 | 2 | 2 | 2 |
| **www.behoff.com** | 53 | 1 | 1 | 1 | 1 | 1 | 1 |
| **srobbin.com** | 24 | 0 | 0 | 0 | 0 | 0 | 0 |
| **jenifferdake.net/blog** | 41 | 0 | 0 | 0 | 0 | 0 | 0 |
| **tumblr.com**<br>**Allisonweiss.tumblr.com** | 60 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.brianjeremy.com** | 44 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.brianyerkes.com** | 36 | 0 | 0 | 0 | 0 | 0 | 0 |
| **hunsonisgroovy.com** | 227 | 0 | 0 | 0 | 0 | 0 | 0 |

While blogs typically have a significantly high number of resources, the content is typically not instantly dynamic. Most of the content is stored in a database and retrieved by the back-end to build a response at query-time. While the response itself is built dynamically, there is not a particular reason for it to be instantly dynamic. Similar to personal websites, any instantly dynamic content that was found had unique components in every response. Therefore, every instantly dynamic resource was correctly identified with only one duplicate request.

**Forum Websites**

While it may not be immediately intuitive, forums are often designed in a very similar way to blogs. Thread contents are stored in a back-end database and allow responses to be dynamically built and returned the user at request-time. When testing forums, it is important to concentrate on the actual content of a forum itself as opposed to the introduction web page. Therefore, each test consisted of navigating to an actual thread or event in each forum for the test. The topics/threads used are given below each forum website in table 15. Over 11 websites, 327 resources were tested with 11 of them being identified as instantly dynamic.

**Table 15: Analysis of Forum Websites**

| Website | Resources | Number of Detected Instantly Dynamic Resources | | | | | Actual |
|---|---|---|---|---|---|---|---|
| | | 1 Dup | 2 Dup | 3 Dup | 4 Dup | 5 Dup | |
| **www.stackoverflow.com** | 16 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.yelp.com …/events/Oakland-much-ado-about…** | 64 | 3 | 4 | 4 | 4 | 4 | 4 |
| **www.freelancer.com www.freelancer.com/sellers/** | 64 | 1 | 2 | 2 | 2 | 2 | 2 |
| **craigslist.org sfbay.craigslist.org/wan/** | 6 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.4chan.org boards.4chan.org/v/res/91074…** | 41 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.explore-ideas.com** | 6 | 1 | 1 | 1 | 1 | 1 | 1 |
| **forums.thestranger.com …/forumdisplay.php?s=ecc515…** | 28 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.12stepforums.net activeboard.com/forum.spark?forum…** | 34 | 1 | 1 | 1 | 1 | 1 | 1 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **www.hypernews.org …/HyperNews/get/government.html** | 26 | 0 | 0 | 0 | 0 | 0 | 0 |
| **forums.whyweprotest.net** | 18 | 1 | 0 | 1 | 1 | 1 | 1 |
| **z4invisionfree.com/PhelpsAnonymous index.php?s=5b301dba0c…** | 24 | 1 | 1 | 1 | 1 | 1 | 1 |

Unlike the *Blog* category, several instantly dynamic resources were misclassified when using a low number of duplicate requests. However, when analyzing the results using three or more duplicate requests, it can be seen that every instantly dynamic resource was correctly classified.

There are several reasons why a forum may contain more instantly dynamic content than a blog. Unlike blogs that typically have one thread and must maintain an order, a forum may benefit from occasionally shuffling new threads and topic areas on certain pages. Additionally, forums often operate off of the revenue of ads, and may have inline ads that would not be blocked through a plugin such as AdBlock Plus. Several forums also contained cross-domain widgets with instantly dynamic content. An example of this is *www.yelp.com* integrating with a mapping service.

### Search Engine Websites

Ever since the popularity of Google's search interface, search engines have attempted to migrate from complex interfaces to a minimalistic design. Because of this, there are typically a small number of resources affiliated with websites in the *Search Engine* category when compared to other highly popular sites such as the *News/News Aggregate* and *Business* websites. Data from 12 websites resulted in 187 resources, 10 of which were classified as instantly dynamic.

**Table 16: Analysis of Search Engine Websites**

| Website | Resources | Number of Detected Instantly Dynamic Resources | | | | | Actual |
|---|---|---|---|---|---|---|---|
| | | 1 Dup | 2 Dup | 3 Dup | 4 Dup | 5 Dup | |
| **www.google.com** | 9 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.yahoo.com** | 34 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.baidu.com** | 6 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.bing.com** | 17 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.yandex.com** | 15 | 0 | 0 | 0 | 0 | 0 | 0 |
| **go.com** | 18 | 1 | 1 | 1 | 1 | 1 | 1 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **ask.com** | 27 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.about.com** | 26 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.mybrowserbar.com** | 5 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.altavista.com** | 13 | 1 | 2 | 2 | 2 | 2 | 2 |
| **www.indeed.com** | 8 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.mywebsearch.com** | 9 | 1 | 1 | 1 | 1 | 1 | 1 |

Most of the instantly dynamic content under this category was correctly classified with only one duplicate request, hinting at the possibility that most of the resources had a unique component in every response instead of simply two or three variations like was seen in many websites under the *News/News Aggregate* and *Business* categories. The only misclassification encountered was under the condition of using one duplicate request for analysis.

**Reference/Image/Video Websites**

At first glance, many of the websites that fell under the *Reference/Image/Video* category may also have been classified under the Search Engine category. The primary difference between these two categories is the location of the indexed content. While search engines typically provide links to cross-domain information, the Reference/Image/Video websites usually aggregate content under their own domain and focus on a more narrowly focused area. Once again, it made more sense in this category to run the test on the content itself as opposed to the introductory web pages. The actual topics and items used for the test are given under the website itself in table 17. Over 18 websites in this category, a total of 987 resources were collected, 21 of which were classified as instantly dynamic.

**Table 17: Analysis of Reference/Image/Video Websites**

| Website | Resources | Number of Detected Instantly Dynamic Resources | | | | | Actual |
|---|---|---|---|---|---|---|---|
| | | 1 Dup | 2 Dup | 3 Dup | 4 Dup | 5 Dup | |
| **www.wikipedia.com**<br>**en.wikipedia.org/wiki/Expected_value** | 68 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.imdb.com**<br>**www.imdb.com/title/tt0106308/** | 81 | 2 | 2 | 2 | 2 | 2 | 2 |
| **www.weather.com**<br>**…/weather/today/Livermore+CA+94550** | 39 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.youtube.com** | 20 | 2 | 2 | 2 | 2 | 2 | 2 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **…/watch/v=CD2LRROpph0** | | | | | | | |
| **www.flickr.com<br>…/photos/malteschmidt/556211…** | 35 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.dailymotion.com/us** | 61 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.ehow.com<br>…/about_5052109_educational-expen…** | 92 | 2 | 2 | 2 | 2 | 2 | 2 |
| **sourceforge.net<br>…/projects/filezilla/** | 26 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.download.com<br>download.cnet.com/AVG-Anti-…** | 68 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.w3schools.com<br>…/js/default.asp** | 21 | 0 | 1 | 1 | 1 | 1 | 1 |
| **metacritic.com<br>…/movie/source-code** | 36 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.Istockphoto.com<br>…/search/text/llama/source…** | 78 | 1 | 1 | 1 | 1 | 1 | 1 |
| **articlebase.com<br>…/business-opportunities-articles/…** | 75 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.w3.org<br>…/TR/2004/REC-DOM-Level-3-…** | 6 | 0 | 0 | 0 | 0 | 0 | 0 |
| **www.accuweather.com<br>…/us/ca/Livermore/94550/…** | 105 | 1 | 2 | 2 | 2 | 2 | 2 |
| www.livescore.com<br>www.liverscore.com/basketball/ | 9 | 1 | 0 | 1 | 1 | 1 | 1 |
| www.wikihow.com<br>…/Make-a-Pirate-Sock-Puppet | 96 | 1 | 1 | 1 | 1 | 1 | 1 |
| www.whitepages.com<br>**…/store-locator/Target** | 71 | 3 | 3 | 3 | 3 | 3 | 3 |

The results from the *Reference/Image/Video* category do not provide any particularly surprising results. All three misclassifications occurred when the analysis used either one or two duplicate requests. While there was a significantly high number of instantly dynamic resources, most of these had unique components that changed with every request, and were easily identifiable with only one duplicate request.

### Other Websites

Many popular websites were difficult to categorize, falling on a border between two or three different possibilities. However, many of them were popular enough that they should be included in the test.

Collecting data from 15 different websites resulted in 704 resources, 19 of which were classified as instantly dynamic.

**Table 18: Analysis of Other Websites**

| Website | Resources | Number of Detected Instantly Dynamic Resources | | | | | Actual |
|---|---|---|---|---|---|---|---|
| | | 1 Dup | 2 Dup | 3 Dup | 4 Dup | 5 Dup | |
| **www.mediafire.com** | 16 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.espn.go.com** | 77 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.fileserve.com** | 18 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.gamespot.com** | 69 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.ign.com** | 129 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.drupal.com** | 14 | 1 | 0 | 1 | 1 | 1 | 1 |
| **www.softpedia.com** | 106 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.multiupload.com** | 37 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.facebook.com** | 15 | 3 | 3 | 3 | 3 | 3 | 3 |
| **www.twitter.com** | 64 | 2 | 2 | 2 | 2 | 2 | 2 |
| **www.linkedin.com** | 15 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.multiply.com** | 23 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.myspace.com** | 23 | 1 | 1 | 1 | 1 | 1 | 1 |
| **www.scribd.com** | 80 | 2 | 2 | 2 | 2 | 2 | 2 |
| **www.xkcd.com** | 18 | 1 | 1 | 1 | 1 | 1 | 1 |

The results from the *Other* category had less misclassifications than had been expected. Due to the popularity and nature of many of the websites, misclassification rates approaching that of the *News/News Aggregate* and *Business* categories were predicted. Contrarily, only one resource was misclassified. This occurred when analyzing *www.drupal.com* with only two duplicate requests.

**Summary**

Similar to the theoretical classification accuracy, the empirical classification accuracy showed that a small number of duplicate requests could provide sufficient accuracy. The majority of misclassifications were seen in the *News/News Aggregate* and the *Business* websites. The typical scenario involved a website with a resource only having two or three variations that were returned in a seemingly random fashion. As was discussed in the theoretical classification section, this type of website provides the highest probability

of a misclassification. It makes sense for both news and business websites to have a dynamic resource capable of changing in subsequent responses. This style of website will appeal more to a person revisiting the website after a short period of time, making it look as if the content is being updated more often than it actually is.

When three or more duplicate requests were used, nearly every resource was correctly identified as instantly dynamic. With three duplicate requests, only two resources would have been incorrectly classified as static. While increasing this to four duplicate requests did not provide any increase in accuracy, there was only one misclassification using 5 duplicate requests. These results indicate that significant diminishing returns will be experienced with anything greater than three duplicate requests. Depending on the ramifications of a misclassification, using more than three duplicate requests seems unnecessary.

The surprising statistic is the number of correctly classified instantly dynamic resources when only one or two duplicates were used. While this would have been anomalous for websites only containing two or three variations of the resource, it actually does not conflict with the theoretical classification. In many of these cases, the resource contained a counter, unique id, timestamp, or some other form of dynamic content that prevented the website from returning the same content twice in nearly every case. This effectively reduced the chance of a misclassification to a value approaching 0.

**OPEN SOURCE ANALYST**

Censorship and misinformation have seen significant time in the spotlight over the last few years. One of the most popular cases has been the ongoing internet censorship of Chinese citizens. Starting in 1996 with the Temporary Regulation of Computer Information Network International Connection, China has continued to pass regulations to control citizen's access to the web (Qiu 1999). More recently, attention has been given to "the Great Firewall", a technology that blocks access to any websites the government considers sensitive. Censored websites range over a vast number of topics such as democracy, religion, news, protest, chat, and porn. This was given additional attention around the time of the Beijing Olympics when the Great Firewall was loosened due to the rise of foreigners in the area (James 2009). However, censorship was once again tightened when "Charter 8", a petition challenging the government, was being spread through internet channels.

Cases like the one above are not restricted to China. Similar censorship is being seen all over the world in places like Cuba, Iran, North Korea, and Egypt (Internet Censorship 2011). The reasons for censorship vary, but the most common censorship is seen with pornography, media sharing, social networking, politics, religion, and circumvention methods.

Closely related to censorship is misinformation. Not only do some countries block citizens from retrieving information from the Internet, but it is not uncommon for misinformation to be spread over the web. While it is possible to create a website that delivers poor information to anyone who visits the site, it is also possible to tailor the information on a website to particular citizens or organizations at specific locations around the world. This strategy can be used by both governments and terrorist organizations to provide misinformation, disinformation, propaganda, and false leads to adversaries (Bolz et al. 2005).

While detecting websites that deliver false information to everyone is not a feasible problem to tackle without a knowledge base and natural language processing, it is possible to detect websites that deliver varying information based on a specific client fingerprint. This can be done by using the multiplexing proxy to analyze the results of a web query from multiple, varying locations around the world. If a response was modified because of where the request originated from, it will be detected when it is analyzed against the other responses.

The following case studies use proxies located in the United States, China, Russia, and India. The specific proxies used have been collected from various websites such as proxy-list.org that publish information about publicly available proxies all over the world (Proxy List). Before use, each proxy was validated by visiting websites with static responses and ensuring the appropriate response was returned.

*The next section requires an understanding of the tool's output and analysis.*
*Please see "Appendix A" for this information.*

**Tailored Content**

*www.bbc.co.uk/news/*

http://www.bbc.co.uk/news/

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html | f8c5891aeed82fd5744bed76ac882fa7 | 83420 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | text/html | bdac4f604e3fd0b5509b6674bbca8b98 | 83420 | view | text | cleaned | stripped | diff | inline_diff | | |
| Duplicate | duplicate | text/html | bdac4f604e3fd0b5509b6674bbca8b98 | 83420 | view | text | cleaned | stripped | diff | inline_diff | | |
| Duplicate | duplicate | text/html | bdac4f604e3fd0b5509b6674bbca8b98 | 83420 | view | text | cleaned | stripped | diff | inline_diff | | |
| Location | China | text/html | 69920ec096b15dd63256d07ccc77e43d | 97475 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | Great Britain | text/html | 1d8df2c337339e53c7864dfe63a651c8 | 89940 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | Russia | text/html | a9d3eedffff3df361ee14b7e29acd3b3 | 97475 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |

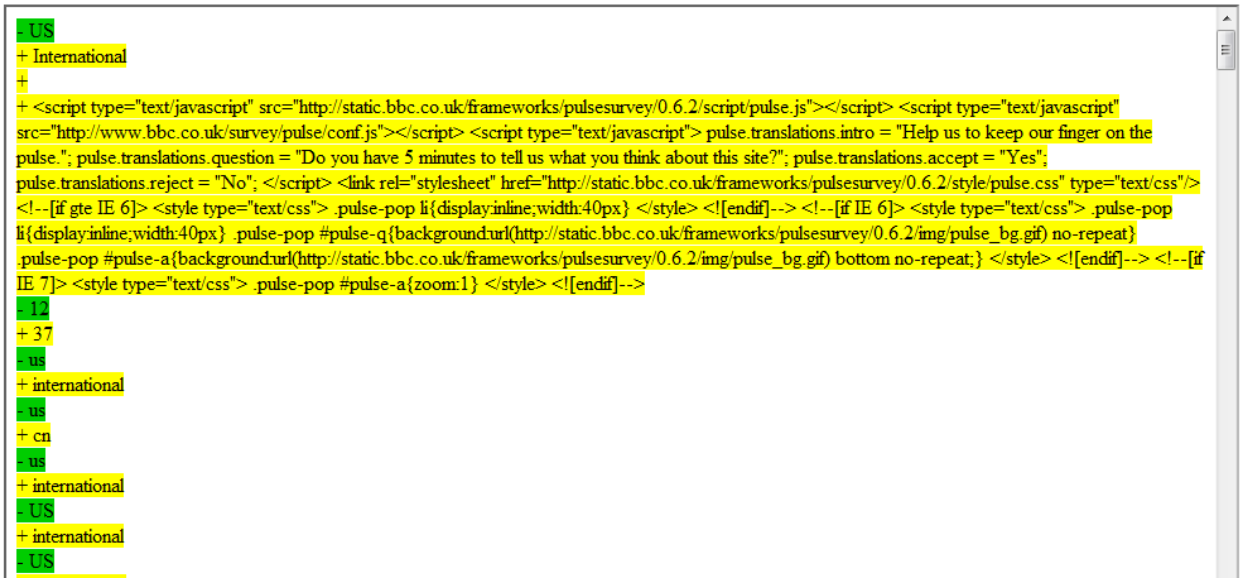



**Figure 4. Tailored responses to requests from the United States**

Figure 4, displaying a "diff" between the responses of a Chinese-based request and a US-based request, shows how the server responds differently depending on whether or not the request came from the US. While the original response contained content tailored toward the United States, the request from China was tailored toward International news. The length anomaly seen in every request sent through a remote proxy provides confidence that this is not a random occurrence. Figure 5, below, shows how the response to the US-based request contained news about Obama, but the response to the overseas request contained overseas news.

```
+ y
- 310162066" href="/news/uk-politics-13539137"><img src="http://news.bbcimg.co.uk/media/images/52956000/jpg/_52956077_012066726-1.jpg"
alt="President Obama addresses cabinet" /><span class="headline heading-13">As it happened: Obama day two</span></a> <br class="ie-clear" />
</div> </div>
+ 279477229" href="/news/uk-scotland-north-east-orkney-shetland-13534802">Fraser wins Arlene murder appeal</a> </li> <li> <a class="story"
rel="published-1306344635591" href="/news/uk-scotland-13551670">Ash unlikely to hit Scots flights</a> </li> </ul> </div>
+ geo-
- unit ">
+ section column-2 ">
-
- 2
+ 4
- heading-16"><a href="/news/magazine/">Magazine</a></h2
+ geo-digest-section-header"><a class="story" href="/news/northern_ireland/">Northern Ireland</a
+
-
+ </h4>
+
-
+ <ul>
+
- <div class=" stacked-144"> <a class="headline-anch
+ <li> <a class="st
+ y
- 287686903" href="/news/uk-13372452"><img src="http://news.bbcimg.co.uk/media/images/52240000/jpg/_52240414_twitter2.jpg" alt="Photo:
Nicholas Kamm - Getty Images Sport" /><span class="headline heading-13">Could Twitter threaten the justice system? </span></a> <br class="ie-clear"
/> </div> </div>
+ 336121058" href="/news/uk-northern-ireland-13550363">Howell is sentenced over assaults</a> </li> <li> <a class="story" rel="published-
1306324356781" href="/news/uk-northern-ireland-13543364">Murderer&#039;s Stormont job condemned</a> </li> </ul> </div>
```

**Figure 5. News articles according to client location**

*www.godaddy.com*

## http://www.godaddy.com/

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=utf-8 | 5171e3cd33ab31633971847cd6d6f4d5 | 265259 | view | text | cleaned | stripped | | |
| Duplicate | duplicate | text/html; charset=utf-8 | fbd9cafa9d57652a1471daa572d9231 | 265259 | view | text | cleaned | stripped | diff | inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8 | c351e75573a5935617707c7d3c46b7d0 | 265259 | view | text | cleaned | stripped | diff | inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8 | e69a6a2dba8774ce8802294e3cfc51a0 | 265259 | view | text | cleaned | stripped | diff | inline_diff |
| Location | China | text/html; charset=utf-8 | 5f1fc8af9f5213db06018d08623b7ca4 | 265280 | view | text | cleaned | stripped | diff | inline_diff |
| Location | Great Britain | text/html; charset=utf-8 | a20bb03f273f13818831ef4aa6c1c2e | 265361 | view | text | cleaned | stripped | diff | inline_diff |
| Location | Russia | text/html; charset=utf-8 | dc96cf8556327aeff4aaae42671f6dc | 265281 | view | text | cleaned | stripped | diff | inline_diff |

```
.linksBlock li { display: inline; float:left; text-align: center; margin: 0px 0px 0px 23px;cursor: pointer; }
.linksBlock span { font-weight: bold; text-decoration: underline; }
</style>
<ul class="linksBlock five">
<li title="Domains for just $1.99&#163;1.23*" style="padding-top: 8px;margin-left:0px;" onclick="domainPromos.execute(event, th
just<br /><span>$1.99&#163;1.23*</span></li>
<li title=".COM Transfers $7.49&#163;4.64*" style="padding-top: 8px;" onclick="location.href='http://www.godaddy.com/domain
transfer.aspx?ci=8908'">.COM Transfers<br /><span>$7.49&#163;4.64*</span></li>
<li title="Save on 6+ Domains" style="padding-top: 8px;" onclick="location.href='http://www.godaddy.com/domains/searchbulk.asp
6+ Domains<br /><span>Plus FREE Privacy</span></li>
<li title=".CO domains $11.99&#163;7.42 first year" onclick="location.href='http://www.godaddy.com/tlds/co-domain.aspx?tld=cc
src="http://img1.wsimg.com/fos/icn/1/57405_icn_co.png" style="padding-right: 2px;border: 0px none;" alt=".CO" title=".CO" />Do
/><span>$11.99&#163;7.42</span></li>
```

**Figure 6. Tailored Prices based on location**

The figure above demonstrates a clear-cut case of content being tailored toward specific client locations. In this case, the prices are being altered in the responses for requests from Great Britain. Once again the consistency of length anomalies for requests sent through a remote proxy provides sufficient information to determine that there is a tailored response based on the location of the client.

```
<li title="Why choose Go Daddy?" class="vid-why" onclick="openVideo('http://www.godaddy.com/shared/video/videos.aspx?ci=36895&
show_vid=GDHPHowTo&pagetype=HPpod3&vid_title=Why+choose+Go+Daddy&vid_name=Products%2fFOS_2_NONUS_640x360_large.flv&
alt_vid_name=Products%2fFOS_2_NONUS_640x360_iPad.mp4&vid_caption=http%3a%2f
%2fa1848.g.akamai.net%2f7%2f1848%2f13927%2fv001%2fgodaddysof1.download.akamai.com%2f13755%2fProducts%2fcc_whychoose.txt',
'http://a1848.g.akamai.net/7/1848/13927/v001/godaddysof1.download.akamai.com/13755/Products/FOS_2_NONUS_640x360_iPad.mp4');">
<div>Why choose<br />Go Daddy?</div></li>
```

**Figure 7. Tailoring for Non-US requests**

*www.indeed.com*

**http://www.indeed.com/**

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html;charset=UTF-8 | dfd971a3176f75c4840c4688ec7c747a | 13996 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | text/html;charset=UTF-8 | 9bed476e117f74e1aec62aba79c21db6 | 13996 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html;charset=UTF-8 | a07695434deafc366f37cacce403e902 | 13996 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html;charset=UTF-8 | 54dcc5fea204e0056f0ff2135b531aa5 | 13996 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | China | text/html;charset=UTF-8 | a214405e547a31ed8e16c075b9005597 | 14083 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | Great Britain | text/html;charset=UTF-8 | 8306fdc5c093711fe15871cff78b4c51 | 14080 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | Russia | text/html;charset=UTF-8 | c70c89181ed8a80da065947ff90f4640 | 13976 | view | text | cleaned | stripped | diff | inline_diff | | |

```
+ <script type="text/javascript">window.location.href='http://www.indeed.co.uk/?r=us';</script>
- in0k2i2m8
+ ou0k3i6jf
- in0k2i2m8
+ ou0k3i6jf
- function pf(f) { document.cookie = 'ippfl=Pleasanton%2C+CA; path=/'; }
- in0k2i2m8
+ ou0k3i6jf
- pf();
- value="Pleasanton, CA"
- ?l=Pleasanton%2C+CA
- in0k2i2m8
+ ou0k3i6jf
+ <p style="text-align: center">For UK jobs, visit <a href="http://www.indeed.co.uk/">Indeed UK</a></p>
- in0k2i2m8
+ ou0k3i6jf
- in0k2i2m8
+ ou0k3i6jf
```

**Figure 8. Script injection due to client location**

The differences in Figure 8 show many obvious examples of location-based tailoring. In this example, information about California is replaced with content about the UK. The length anomalies seen for all of the requests through a remote proxy make it apparent that this also happens for many other locations.

Looking closely at some of the differences reveals the intent of the tailoring. Figure 9, below, shows how a script is injected into the response to redirect clients located in China to a domain tailored for that particular location.

```
<p style="text-align: center">For jobs in China, visit <a href="http://cn.indeed.com/">Indeed China</a></p>
```

**Figure 9. Redirection script injected into responses for clients located in China**

Because some clients will have scripts turned off or blocked by a plugin, the response additionally contained text and links directing the user to the correct site. This can be seen in Figure 10, below.

```
<p style="text-align: center">For jobs in China, visit <a href="http://cn.indeed.com/">Indeed China</a></p>
```

**Figure 10. Additional attempt to redirect Chinese users.**

*www.bing.com*

http://www.bing.com/

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=utf-8 | 208c81fc1ad9c99ad9c3ccb38547cb8b | 32253 | view | text | cleaned | stripped | | | |
| Duplicate | duplicate | text/html; charset=utf-8 | edaf35f039715ed47513db59125862df | 32253 | view | text | cleaned | stripped | diff | inline_diff | |
| Duplicate | duplicate | text/html; charset=utf-8 | 94982b91d889aff17b575bc08bbcb367 | 32253 | view | text | cleaned | stripped | diff | inline_diff | |
| Duplicate | duplicate | text/html; charset=utf-8 | 260490cd1bf9cac103278ef8efdbe6f8 | 32253 | view | text | cleaned | stripped | diff | inline_diff | |
| Location | China | text/html; charset=utf-8 | 695887160f424ce185146b047cea151b | 32261 | view | text | cleaned | stripped | diff | inline_diff | |
| Location | Great Britain | text/html; charset=utf-8 | 74fb2df7866ea26c2538c577b3a81823 | 26728 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | Russia | text/html; charset=utf-8 | 8264601dab6063f45ec96220f0d28b8a | 32261 | view | text | cleaned | stripped | diff | inline_diff | |

Bing Explore | MSN | Hotmail Sign in ~~▼ | Rewards | United States | Preferences Bing~~| United Kingdom | Preferences Bing Show all Only from United Kingdom Images Videos Shopping News Maps ~~Travel Entertainment~~ Search History Visual Search   »   »   »   » ◄ ► © ~~Evaporating salt water reservoirs of Teguidda n Tessoumt, Niger — George Steinmetz/Corbis Facebook friends Now on Bing, try social search Spring shopping spree? Earn gift cards with Bing Popular now Kim Kardashian · Area 51 · Tornadoes · Mark Haines · Nancy Grace © 2011 Microsoft | Privacy | Legal | Advertise | About our ads | Help | Tell us what you think X Make better decisions with the help of your friends Learn more~~Young sand cat in United Arab Emirates -- Xavier Eichaker/Photolibrary Purr-fect Pick out your favourite kitty Feline good Watch a cat play piano On this day in 1982â?¦ Which English club, one of only four to lift it, won the European Cup? © 2011 Microsoft | Privacy | Legal | Advertise | Help | Feedback

**Figure 11. Tailored Search Engine Menu Based on Location**

Figure 11, above, shows *Bing* tailoring the search engine menu based on the location of the client. The analysis showed all three remote requests receiving a response with an anomalous length. While some of the data toward the end of the response initially appeared to be random articles, it is apparent from looking at the consistency in the lengths of the responses to duplicate requests that this is not the case. Without analytics that can detect this type of anomaly, manual inspection of responses may have overlooked the location-based tailoring.

*www.avg.com*

**http://www.avg.com/**

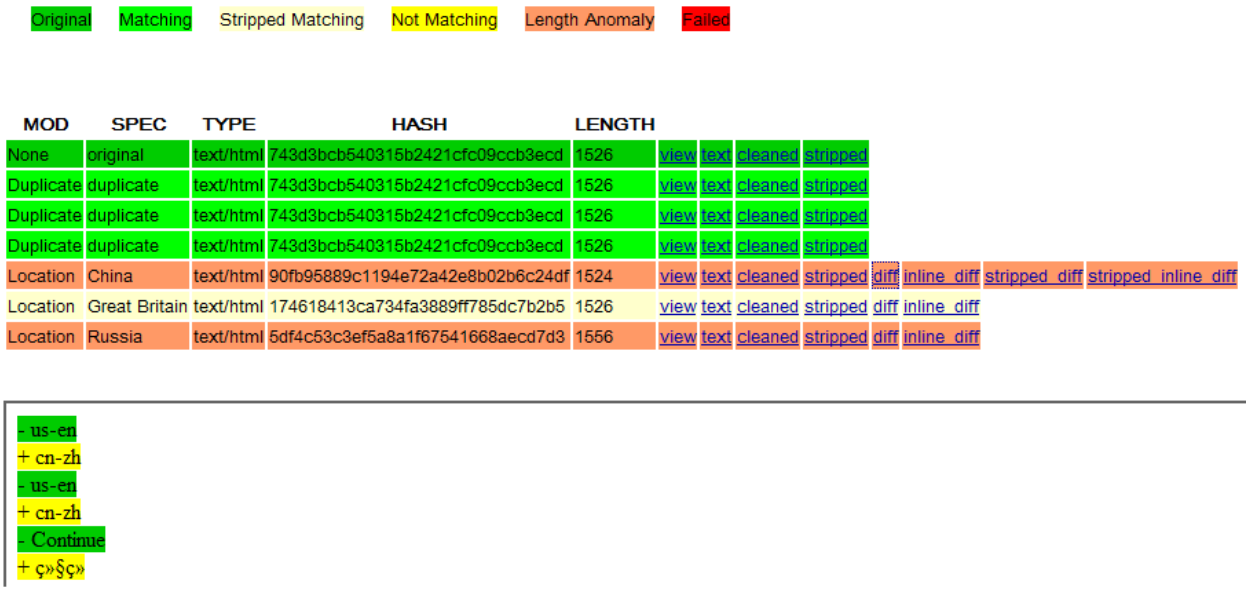



**Figure 12. Tailored links and content**

While the location-based tailoring performed by *AVG* was minimal, it was obvious due to the high number of responses that matched the original response. Each of the responses to a request through a remote proxy had multiple links containing a location-based substring. Additionally, the language of the "continue" option provided to the user accounted for the location of the request.

*www.wired.com*

http://www.wired.com/images/index/2011/05/mba_bg.jpg

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | |
|---|---|---|---|---|---|
| None | original | image/jpeg | b56f50d6c2c7f08320a016aafa2c8668 | 16567 | view |
| Duplicate | duplicate | image/jpeg | b56f50d6c2c7f08320a016aafa2c8668 | 16567 | view |
| Duplicate | duplicate | image/jpeg | b56f50d6c2c7f08320a016aafa2c8668 | 16567 | view |
| Duplicate | duplicate | image/jpeg | b56f50d6c2c7f08320a016aafa2c8668 | 16567 | view |
| Location | China | image/jpeg | b56f50d6c2c7f08320a016aafa2c8668 | 16567 | view |
| Location | Great Britain | image/jpeg | b56f50d6c2c7f08320a016aafa2c8668 | 16567 | view |
| Location | Russia | image/jpeg | 1414c8e09ded56bd95d17edff91a9205 | 9454 | view |

**Figure 13. Image from www.wired.com**

An interesting image resource was found on *www.wired.com*. While the same image was returned for nearly all of the responses, the request for the same URL sent through a proxy in Russia returned a different image. Figure 14, below, shows the tailored response.

**Figure 14. Tailored image from the same URL**

While instantly dynamic images are not uncommon in websites, it is typically performed by modifying the URL in the HTML response instead of through dynamically returning different images from the same URL. To examine the possibility that it may have been a random occurrence due to an instantly dynamic image, the case study was repeated, and had a similar result.

**Blocked Domains**

***news.google.com / www.partypoker.com***

**http://news.google.com/**

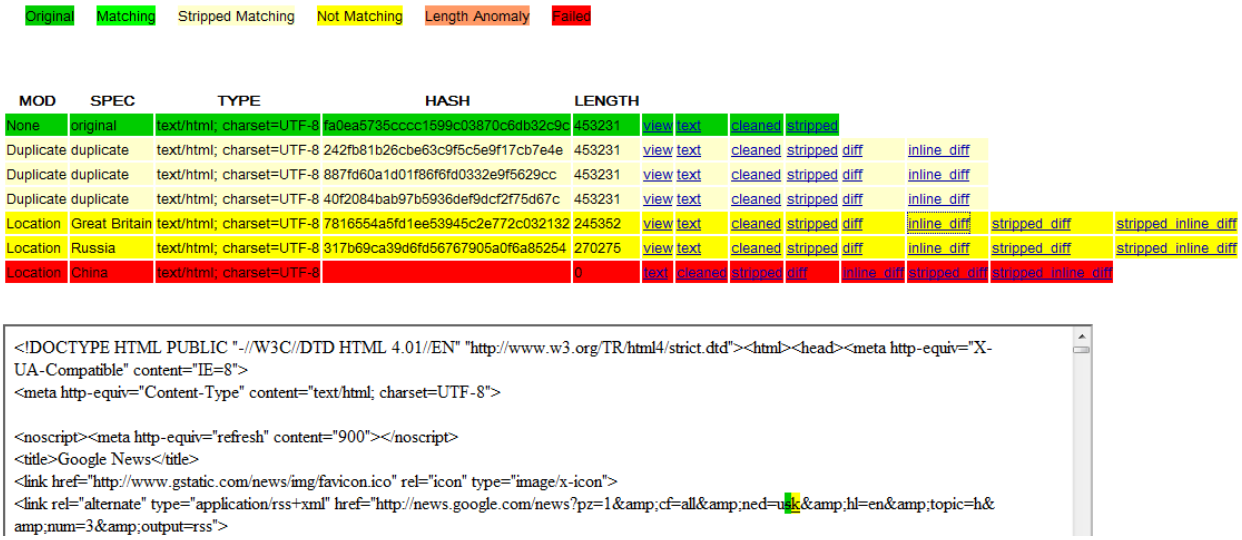
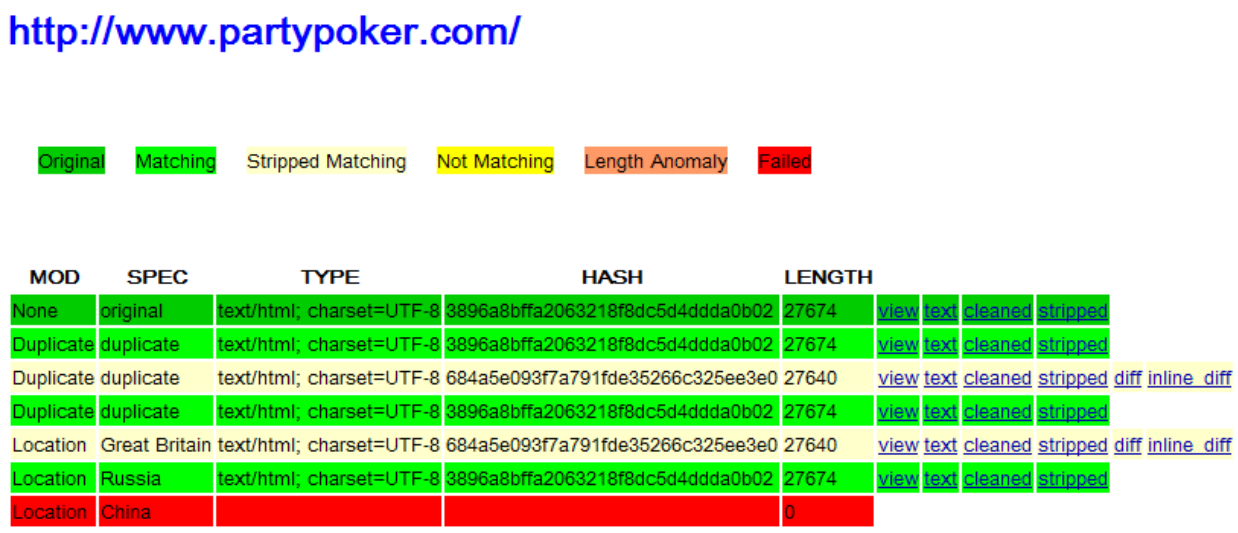



**Figure 15. Tailored and blocked responses based on client location**

Figure 15 shows the responses to all of the duplicate requests as matching the original response after stripping out benign attributes and elements. However, the requests sent through a remote proxy received a response that either did not match or failed. This clearly shows anomalous behavior based on the location of the client. The requests sent through proxies in Great Britain and Russia received a response containing small, tailored changes. However, the request routed through China was blocked. Mentioned previously, China is known for their heavy censorship of pictures, news, and content they consider dangerous or misaligned with the information they would like to be spread. This is a straight-forward case of the "Chinese firewall" in action.

Figure 16 shows similar blocking for Chinese-based requests to www.partypoker.com. While the website appears to randomly return one of two different responses, an anomaly is quickly noticed in the request sent through a proxy in China.

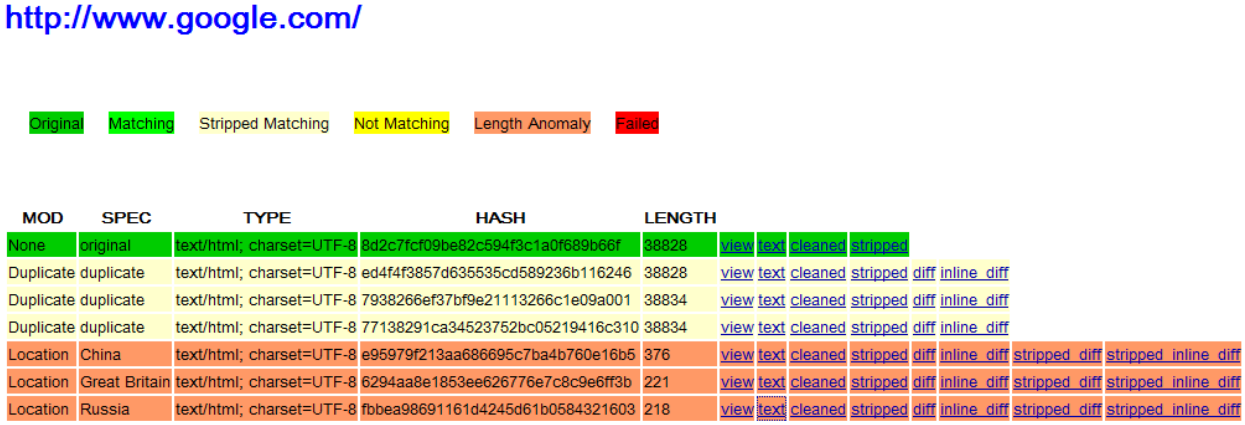



| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=UTF-8 | 3896a8bffa2063218f8dc5d4ddda0b02 | 27674 | view | text | cleaned | stripped | | |
| Duplicate | duplicate | text/html; charset=UTF-8 | 3896a8bffa2063218f8dc5d4ddda0b02 | 27674 | view | text | cleaned | stripped | | |
| Duplicate | duplicate | text/html; charset=UTF-8 | 684a5e093f7a791fde35266c325ee3e0 | 27640 | view | text | cleaned | stripped | diff | inline_diff |
| Duplicate | duplicate | text/html; charset=UTF-8 | 3896a8bffa2063218f8dc5d4ddda0b02 | 27674 | view | text | cleaned | stripped | | |
| Location | Great Britain | text/html; charset=UTF-8 | 684a5e093f7a791fde35266c325ee3e0 | 27640 | view | text | cleaned | stripped | diff | inline_diff |
| Location | Russia | text/html; charset=UTF-8 | 3896a8bffa2063218f8dc5d4ddda0b02 | 27674 | view | text | cleaned | stripped | | |
| Location | China | | | 0 | | | | | | |

**Figure 16. The Great Firewall of China blocking a domain**

**302 Response**

*www.google.com*




| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=UTF-8 | 8d2c7fcf09be82c594f3c1a0f689b66f | 38828 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | text/html; charset=UTF-8 | ed4f4f3857d635535cd589236b116246 | 38828 | view | text | cleaned | stripped | diff | inline_diff | | |
| Duplicate | duplicate | text/html; charset=UTF-8 | 7938266ef37bf9e21113266c1e09a001 | 38834 | view | text | cleaned | stripped | diff | inline_diff | | |
| Duplicate | duplicate | text/html; charset=UTF-8 | 77138291ca34523752bc05219416c310 | 38834 | view | text | cleaned | stripped | diff | inline_diff | | |
| Location | China | text/html; charset=UTF-8 | e95979f213aa686695c7ba4b760e16b5 | 376 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | Great Britain | text/html; charset=UTF-8 | 6294aa8e1853ee626776e7c8c9e6ff3b | 221 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | Russia | text/html; charset=UTF-8 | fbbea98691161d4245d61b0584321603 | 218 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |

```
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="http://www.google.ru/">here</A>.
</BODY></HTML>
```

**Figure 17. Tailored 302 Response for Russia**

One of the most common strategies for providing location-based content is to use a "302 Found" response, redirecting the user to a new URL that contains a tailored HTML response. Figure 17 shows how Google uses this strategy to forward Russian clients to *www.google.ru*, a URL tailored for them. The length anomaly seen for every request through a remote proxy makes it apparent that this is a common scenario. Figure 18 confirms this, similarly showing a tailored redirect response sent to the proxy located in Great Britain.

```
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="http://www.google.co.uk/">here</A>.
</BODY></HTML>
```

**Figure 18. Tailored 302 Response for Great Britain**

Google uses a different strategy for a JavaScript resource required by the original HTML response. In this case, tailored JavaScript is included directly in the response. This can be seen in Figure 19, below.

http://www.google.com/extern_chrome/1018d3c18568d7dd.js

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/javascript; charset=UTF-8 | fdb474f42b7aa76213b809650b887734 | 31017 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | text/javascript; charset=UTF-8 | fdb474f42b7aa76213b809650b887734 | 31017 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | text/javascript; charset=UTF-8 | fdb474f42b7aa76213b809650b887734 | 31017 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | text/javascript; charset=UTF-8 | fdb474f42b7aa76213b809650b887734 | 31017 | view | text | cleaned | stripped | | | | |
| Location | China | text/javascript; charset=UTF-8 | d98464d520f6cdc7b3d7dd5cfaff9495 | 30716 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | Great Britain | text/javascript; charset=UTF-8 | d19984334b0a9d84a25fede303f41c42 | 33498 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | Russia | text/javascript; charset=UTF-8 | d98464d520f6cdc7b3d7dd5cfaff9495 | 30716 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | United States | text/javascript; charset=UTF-8 | 351219c4b34b594c2bca919c4c5ebace | 33501 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |

```
- 1018d3c18568d7dd
+ d224a158d22da450
- .soi{background:#ebeff9;line-height:22px;padding:0 4px;position:static;vertical-align:middle}.soi a{white-space:nowrap}.soi img{margin-top:-3px;vertical-
align:middle}.soi .lsbb{display:inline-block;height:20px;margin-bottom:4px}.soi .lsb{background-repeat:repeat-x;font-size:small;height:20px;padding:0 5px}
- overflow:hidden;text-overflow:ellipsis;width:100%.
- //ssl
+ http://www
- //ssl
+ http://www
-
+ id\x3d\x22scroll_col\x22\x3e \x3cdiv
+ \x3c/div\x3e
- -US
```

**Figure 19. Tailored JavaScript for China**

Once again the length anomalies combined with visual inspection show a significant amount of tailoring based on the location of the client.

***www.gawker.com / www.gizmodo.com***

**http://gawker.com/**

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=utf-8; | cd02fe7b41a4e9afc4514f45eea66a48 | 84405 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | text/html; charset=utf-8; | ec05bbcf6e08dde7903eac92206816e0 | 84405 | view | text | cleaned | stripped | diff | inline_diff | | |
| Duplicate | duplicate | text/html; charset=utf-8; | c92501efaad30261f8d616cad7d72026 | 84405 | view | text | cleaned | stripped | diff | inline_diff | | |
| Duplicate | duplicate | text/html; charset=utf-8; | 9bdea97579ac157a2526a121c17e9b45 | 84405 | view | text | cleaned | stripped | diff | inline_diff | | |
| Location | China | text/html; charset=utf-8; | ed3f443edbae8c4c6a516a246ad856ff | 84405 | view | text | cleaned | stripped | diff | inline_diff | | |
| Location | Great Britain | text/html; charset=iso-8859-1 | 64b5d284b0bb98a0ece1fd1b3e088d1e | 204 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | Russia | text/html; charset=utf-8; | 3b8512dee13fb9f67b4af6962035df77 | 84405 | view | text | cleaned | stripped | diff | inline_diff | | |

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://uk.gawker.com">here</a>.</p>
</body></html>
```

**Figure 20. Tailored redirect for United Kingdom**

Gawker also used the “302 found” strategy to redirect specific client fingerprints. However, this website appears to only redirect a subset of the clients that Google did. In this case, only the request sent through a proxy in Great Britain was redirected. This emphasizes the importance of a diverse set of multiplexed requests covering a breadth of client configurations. If the request from Great Britain was not present, it would have appeared as though the website did not ever tailor information based on the location of the client. The “302 Found” response was immediately identified as a length anomaly as the response dropped from 84405 bytes to 204 bytes. Shown in figure 21, a similar case was seen with www.gizmodo.com.

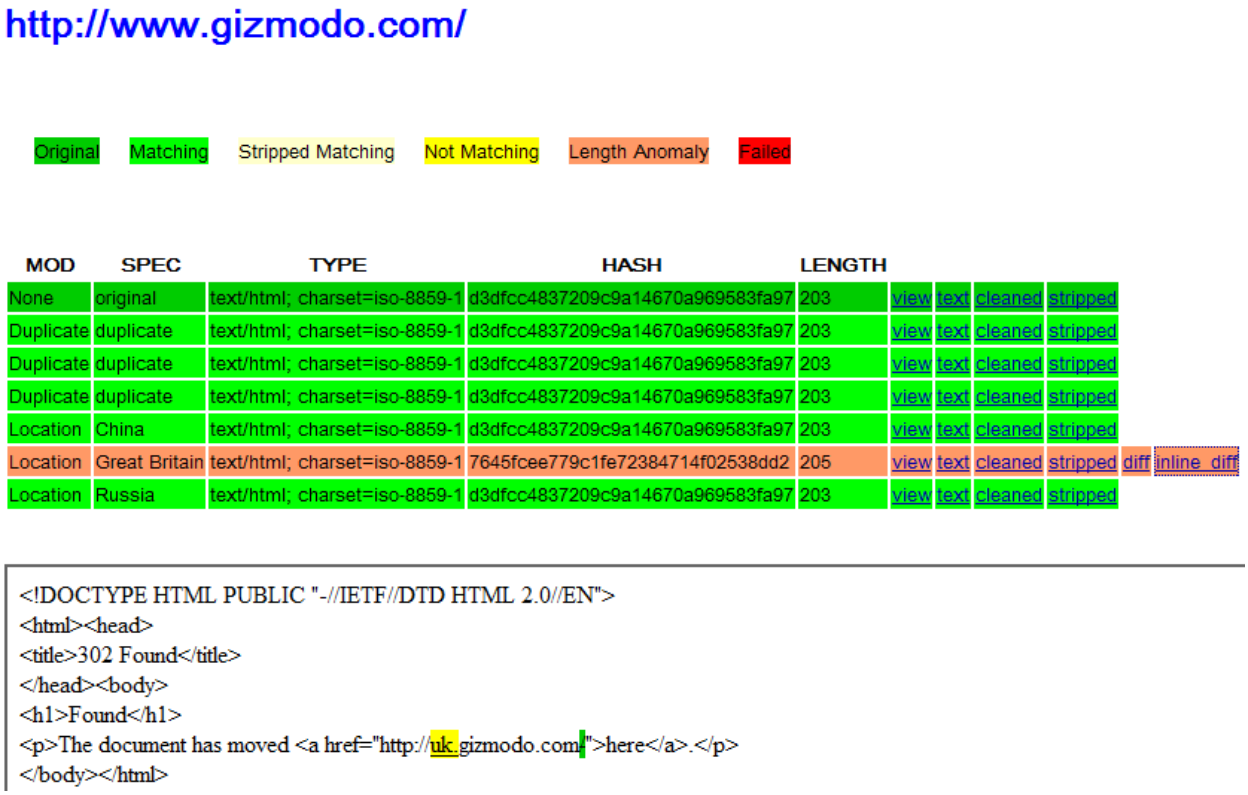



**Figure 21. Another tailored redirect for United Kingdom**

In this case, all of the responses were redirects. However, clients located in Great Britain received a tailored redirect, causing the length anomaly. Additionally, Gizmodo's website contained a third party resource that returned tailored content. This can be seen figure 22, below.


**Figure 22. Location-based tailoring in third party content**

Figure 22 shows how tailored JavaScript is returned from Google based on geolocation of the IP address used in the request. Inspecting the content hints at Gizmodo integrating with Google Earth to provide location-based tailored data on their website.

**Fingerprint Collection**

*www.att.com*

http://www.att.com/

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=UTF-8 | 4ff13a01fdf67bc6682ccfee03bcbac8 | 42723 | view | text | cleaned | stripped | | |
| Duplicate | duplicate | text/html; charset=UTF-8 | c20d855450cd00ccc5888a700314ef86 | 42723 | view | text | cleaned | stripped | diff | inline diff |
| Duplicate | duplicate | text/html; charset=UTF-8 | 6f4adcfcfbe39d13ac7a5cb62e55b691 | 42723 | view | text | cleaned | stripped | diff | inline diff |
| Duplicate | duplicate | text/html; charset=UTF-8 | ae3471472109a5cbab5c016fc6089d71 | 42723 | view | text | cleaned | stripped | diff | inline diff |
| Location | China | text/html; charset=UTF-8 | f59314f6b594af1147999f7c95c771fa | 42296 | view | text | cleaned | stripped | diff | inline diff |
| Location | Great Britain | text/html; charset=UTF-8 | 1dd0750b666e02e6a04fef8ed4f2392b | 42290 | view | text | cleaned | stripped | diff | inline diff |
| Location | Russia | text/html; charset=UTF-8 | 79bf24e1d80cd69e9178b221b7b28e7d | 42723 | view | text | cleaned | stripped | diff | inline diff |

```
<!-- akamai headers here -->
<meta name="DCSext.wt_aka_georegion" content="24647" />
<meta name="DCSext.wt_aka_country_code" content="USCN" />
<meta name="DCSext.wt_aka_region_code" content="CASH" />
<meta name="DCSext.wt_aka_city" content="PLEASANTON" />
<meta name="DCSext.wt_aka_dma" content="807" />
<meta name="DCSext.wt_aka_pmsa" content="5775" />
<meta name="DCSext.wt_aka_msa" content="7362" />
<meta name="DCSext.wt_aka_areacode" content="925" />
<meta name="DCSext.wt_aka_county" content="ALAMEDA" />
<meta name="DCSext.wt_aka_fips" content="06001SHANGHAI" />
<meta name="DCSext.wt_aka_lat" content="37.692201.22" />
<meta name="DCSext.wt_aka_long" content="-121.8937746" />
<meta name="DCSext.wt_aka_timezone" content="PST" />
<meta name="DCSext.wt_aka_zip" content="94566+9458GMT+8" />
<meta name="DCSext.wt_aka_continent" content="NAAS" />
<meta name="DCSext.wt_aka_throughput" content="vhigh" />
<meta name="DCSext.wt_aka_bw" content="52000" />
<meta name="DCSext.wt_aka_network" content="comcast" />
<meta name="DCSext.wt_aka_asnum" content="336517621" />
<meta name="DCSext.wt_aka_network_type" content="cableproxy" content="anonymous" />
<meta name="DCSext.wt_aka_location_id" content="0" />
```

**Figure 23. WebTrends statistics**

The above response shows very detailed information about the location of the client being collected for WebTrends. WebTrends allows the site owner to collect statistics and query trends about clients visiting the website. Shown below in Figure 24, the server was even able to identify the difference between the

Comcast connection used in the original request and the anonymous proxy used for the request sent through a proxy in China.

<meta name="DCSext.wt_aka_network" content="comcast" />
<meta name="DCSext.wt_aka_asnum" content="336517621" />
<meta name="DCSext.wt_aka_network_type" content="cableproxy" content="anonymous" />
<meta name="DCSext.wt_aka_location_id" content="0" />

**Figure 24. Recognizing the switch from Comcast to an Anonymous Proxy**

This shows not only the collection of information about the client, but also shows how the information is augmented with previously collected information about IP addresses, providers, and anonymous proxies.

**Injected IP Address**

*www.opendns.com*

http://www.opendns.com/ *Location Anomaly!* *Variations: 4* *Stripped Variations: 1* *Structure Variations: 1* *Length Anomalies: 3*

http://www.opendns.com/

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=UTF-8 | 3e9e8a3a7eb5190e856cf634cab3eead | 11999 | view | text | cleaned | stripped | | |
| Duplicate | duplicate | text/html; charset=UTF-8 | 3e9e8a3a7eb5190e856cf634cab3eead | 11999 | view | text | cleaned | stripped | | |
| Duplicate | duplicate | text/html; charset=UTF-8 | 3e9e8a3a7eb5190e856cf634cab3eead | 11999 | view | text | cleaned | stripped | | |
| Duplicate | duplicate | text/html; charset=UTF-8 | 3e9e8a3a7eb5190e856cf634cab3eead | 11999 | view | text | cleaned | stripped | | |
| Location | China | text/html; charset=UTF-8 | ae84688f4172f12e601c5395c30430b5 | 12000 | view | text | cleaned | stripped | diff | inline diff |
| Location | Great Britain | text/html; charset=UTF-8 | a8cae470466767b25f6252a4d3def0da | 12002 | view | text | cleaned | stripped | diff | inline diff |
| Location | Russia | text/html; charset=UTF-8 | 32ebf9d87aa6d170cc695ae42ee7a802 | 12000 | view | text | cleaned | stripped | diff | inline diff |

Your current IP is 24.4.25.115117.34.73.50.

**Figure 25. IP address injected into the response content**

Because OpenDNS is heavily invested in its ability to offer secure Internet navigation, it is particularly interested in the IP addresses of clients connecting to their web service. In figure 25, above, the inline difference between the original response and a response sent through a remote proxy clearly shows how the IP address of the connecting client is injected into the response in a desperate move to impress the

visitor. This caused all of the duplicate requests to match, but all of the requests sent through a proxy to correctly appear as an anomaly when compared to the original response.

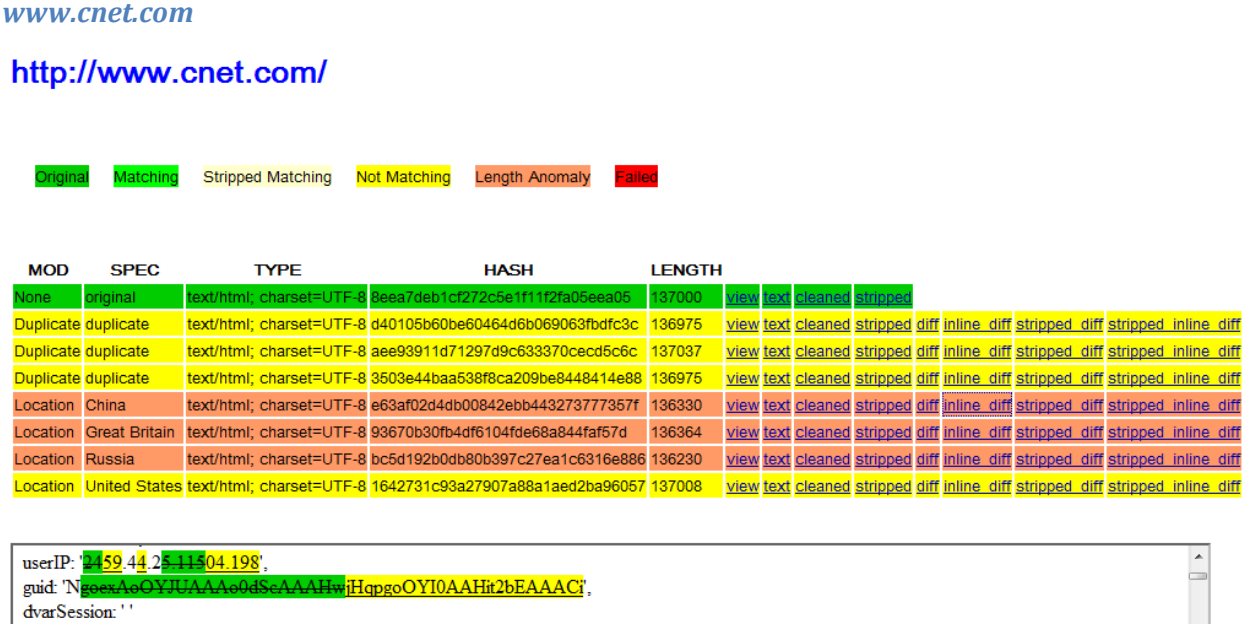



**Figure 26. Inserting the client's IP address into a JavaScript object**

Figure 26 clearly shows that the IP address of the client is being inserted into a JavaScript object in the responses. However, this alone does not account for the length anomalies seen for the requests sent through remote proxies. Unfortunately, as figure 27 shows, the differences were too significant to be useful. However, the consistency of length anomalies makes a strong case toward additional location-based tailoring in the response.


**Figure 27. Response "Diff" failing to find patterns**

**Tailored Marketing**

### *www.adobe.com / www.comcast.com / omtrdc.com*

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/javascript | f817896aa1404abe3fa4a0d46338c8ac | 122 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | text/javascript | f817896aa1404abe3fa4a0d46338c8ac | 122 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | text/javascript | f817896aa1404abe3fa4a0d46338c8ac | 122 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | text/javascript | f817896aa1404abe3fa4a0d46338c8ac | 122 | view | text | cleaned | stripped | | | | |
| Location | China | text/javascript | d42c80c7b0c9bdb9b56ad6fe4db857e8 | 196 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | Great Britain | text/javascript | ee84b8f1f499f43e4fbf527fae3765ec | 196 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | Russia | text/javascript | ee84b8f1f499f43e4fbf527fae3765ec | 196 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |

```
mboxFactories.get('default').get('adobecom_enus_Homepage_Homepage_priority1',0).setOffer(new
mboxOfferDefault()).loaded();mboxFactories.get('default').getPCId().forceId("1305305153076-634464.18");
```

**Figure 28. Adobe Marketing Suite Forcing Identification**

When visiting *www.adobe.com*, a request was made for an 3rd party resource located at omtrdc.com. The responses received by remote proxies all had an additional line of JavaScript that appeared to assign a PC identifier to the system. RMTODC is an Adobe marketing suite, making it likely that this is a way they are keeping track of the same user despite the use of proxies. This may be due to a cookie on the machine that allows the session to be kept alive despite the re-routed traffic. *www.comcast.com*, shown in Figure 29, showed a similar behavior.

### *www.comcast.com / omtrdc.com*

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/javascript | ec5fb05b1350102b7d568b07cae1e6a4 | 102 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | text/javascript | ec5fb05b1350102b7d568b07cae1e6a4 | 102 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | text/javascript | ec5fb05b1350102b7d568b07cae1e6a4 | 102 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | text/javascript | ec5fb05b1350102b7d568b07cae1e6a4 | 102 | view | text | cleaned | stripped | | | | |
| Location | China | text/javascript | b516a45e92fd9fa192b523838c304aac | 176 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | Great Britain | text/javascript | caaef26323353bfaa122341fd40c3f70 | 176 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | Russia | text/javascript | caaef26323353bfaa122341fd40c3f70 | 176 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |

```
mboxFactories.get('default').get('Homepage_Flash_Banner',0).setOffer(new
mboxOfferDefault()).loaded();mboxFactories.get('default').getPCId().forceId("1305309391096-109250.18");
```

**Figure 29. Adobe Marketing Suite Forcing Identification**

Once again, the additional line of JavaScript is only seen in the responses to requests sent through a remote proxy.

### Misclassification

*news.yahoo.com*

http://news.yahoo.com/

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html;charset=utf-8 | 29aed19b4c197b219abc2338c5a2ffec | 229493 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | text/html;charset=utf-8 | bcc8359978e7db6cdc7ffbc7de265c10 | 228464 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html;charset=utf-8 | 698cc560b9328b6bc9eb666a70da41c9 | 229557 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html;charset=utf-8 | 9835235539820bb06dae411d0cf816a5 | 229226 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | China | text/html;charset=utf-8 | f08418d02b659fb32d61be0244dcfab | 226674 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | Great Britain | text/html;charset=utf-8 | ad4e4f3b49c301476a298b4a10654404 | 228125 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Location | Russia | text/html;charset=utf-8 | 953dd4b2836a234326dd8a424d27dde | 228112 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |

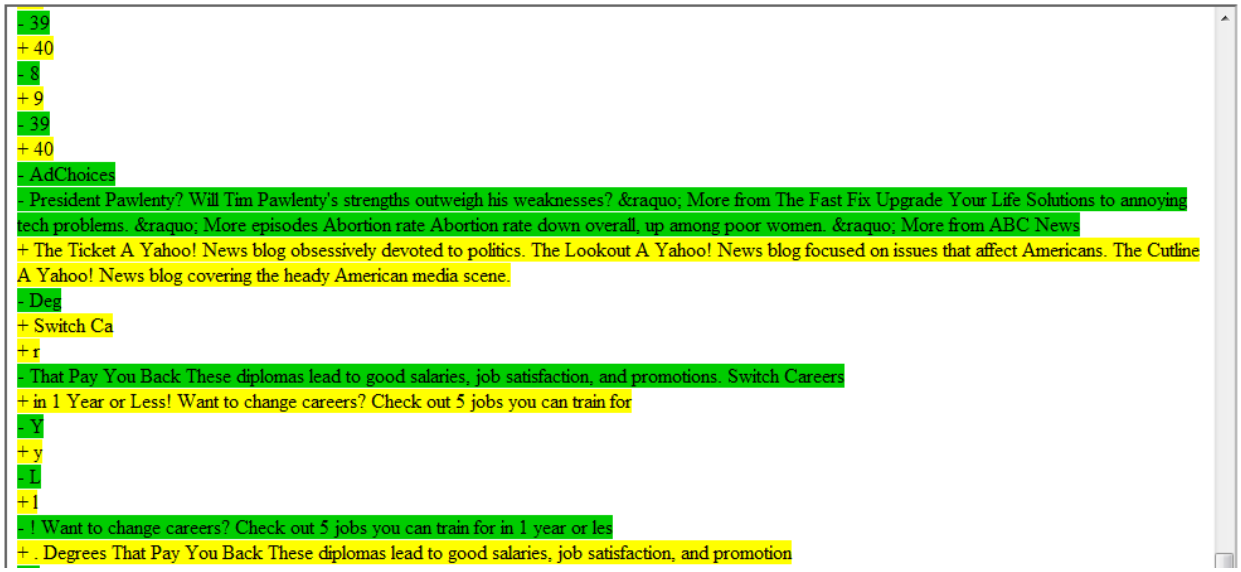



**Figure 30. Misclassification due to a length anomaly**

Articles on news websites can often range drastically in the size of their summaries and content. Because many news websites have instantly dynamic articles and ads, this increases the probability that there will be a response that varies heavily in length when compared to other responses. Figure 30 shows a response for a Chinese-based client coming back as significantly shorter than the other responses. However, when

looking over the differences, none of them appeared to be due to location-based tailoring. While it is possible that the response returned to the proxy in China was tailored, there is not sufficient evidence to make a strong case for it.

### A Case Against Geolocation

***www.fedex.com / www.ups.com***

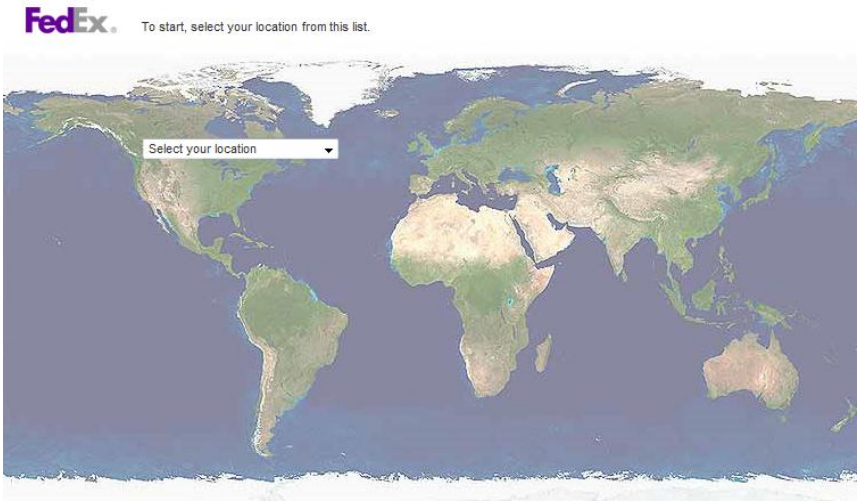



**Figure 31. FedEx requiring the user to input a location**


**Figure 32. UPS requiring the user to input a location**

The two worldwide shipping companies shown in Figures 31 and 32, with business models based entirely around client location, are not using geolocation to initially tailor content. Both of these companies are in an ideal position to both collect and use geolocation data to offer better service to their customers. One has to wonder whether or not this was tried in the past, but did not provide sufficient accuracy.

**Summary**

Exploring the utility of the multiplexing proxy for an open source analyst provided many useful discoveries. Tailored news, prices, and resources are just a few of the types of detected content tailoring. In addition to the tool successfully detecting tailored content, its analytics proved ideal for detecting both censorship and redirection.

A major weakness detected in the multiplexing proxy is its inability to cleanly handle websites with multiple, large changes in content. When the responses differ significantly, the "diff" algorithms can have difficulty appropriately matching the content. Unfortunately, this can often be the case in large, dynamic news websites. Misclassification in this context was seen when analyzing news.yahoo.com.

Another unexpected weakness from the perspective of an open source analyst is the amount of tailored JavaScript and metadata in the response. While this information is useful for individuals such as reverse engineers, it does not benefit someone interested in the visible content on the website. Despite this, there is still a large amount of useful discoveries that would greatly benefit an individual in the open source analyst role.

## CYBER ANALYST

One method of tailoring an exploit toward a specific client fingerprint is to include multiple exploits in the response and use JavaScript browser detection strategies to direct the flow of the script to the correct exploit for the particular browser being used. While this is an effective approach, it has the downside that every exploit (or malicious domain) must be provided to the client. This provides a reverse engineer with more information than was necessary to accomplish the task. Because of the value of zero-day exploits, this typically works against an adversary. A reverse engineer, with visibility to all the exploits, can develop signatures for the exploits and black lists for the domains and IP addresses.

Alternatively, a server can fingerprint a client and send only the exploit necessary for that particular system. If the client happens to be a cyber analyst or honeypot, there is less loss of valuable exploit information. This strategy is possible through both passive and active client fingerprinting. Using active fingerprinting, JavaScript must be sent to the client that will identify the system and send the information back to the server. This information can then be used in responses to subsequent requests for new or updated content. The second approach, passive fingerprinting, removes the ability of the end client to see the code that fingerprints their system. Because fingerprinting code is not sent to the client, it is not easily detectable like with active fingerprinting. Once again, the client is provided with even less information, only receiving an exploit when the server can passively deduce that the client is vulnerable to a specific exploit in the adversary's collection. While the multiplexing proxy does not have the ability to detect tailored exploit delivery through the use of complex, conditional fingerprinting logic in JavaScript that is delivered to every client, it ironically has great potential at detecting the more stealthy method.

The following case studies show how the multiplexing proxy can detect when a malicious server is only delivering tailored exploits to a vulnerable client. In these scenarios, a website has been modified in a lab environment to mimic the behavior of malicious websites using popular attacks against clients. In the following case studies, the server passively fingerprints client requests and returns content similar to a tailored exploit whenever the Internet Explorer browser is identified. Each scenario involves visiting the modified website with the multiplexing proxy configured to use duplicate requests, multiple browser fingerprints, and multiple operating system fingerprints. While the first four scenarios are based on a website with static content, the fifth scenario provides a more complex demonstration with multiple exploits delivered over a website already returning dynamic content.

Additionally, a sixth scenario will look at how the multiplexing proxy was able to detect an illegitimate remote proxy before it was used for live testing. This example will show that the multiplexing proxy is able to find strange activity not just inside a test environment, but also is able to find anomalous activity on the Internet.

*The next section requires an understanding of the tool's output and analysis.*
*Please see "Appendix A" for this information.*

**JavaScript Injection**

Malicious attacks on the web commonly take the form of obfuscated JavaScript. While intrusion detection systems and anti-virus software look for signatures developed from previously seen malicious code, JavaScript's robust obfuscation capabilities typically render them useless. When new exploits are discovered for a particular browser, a server is able to inject malicious, obfuscated JavaScript into responses destined for the targeted browser. While the exploit may be contained in the JavaScript itself, it is most common to see a script that redirects the user to a malicious cross-domain website.

This scenario assumes that a new Internet Explorer exploit has been discovered and is actively being used to exploit visitors at a cross-domain website. In order to redirect clients to this website, a malicious, seemingly benign web server has been configured to deliver JavaScript in the head section of the initial web page whenever a client using Internet Explorer made the request. The JavaScript is responsible for redirecting the user to the web page hosting the malicious exploit.

Below, figure 33 shows the analysis provided by the multiplexing proxy when visiting the website.

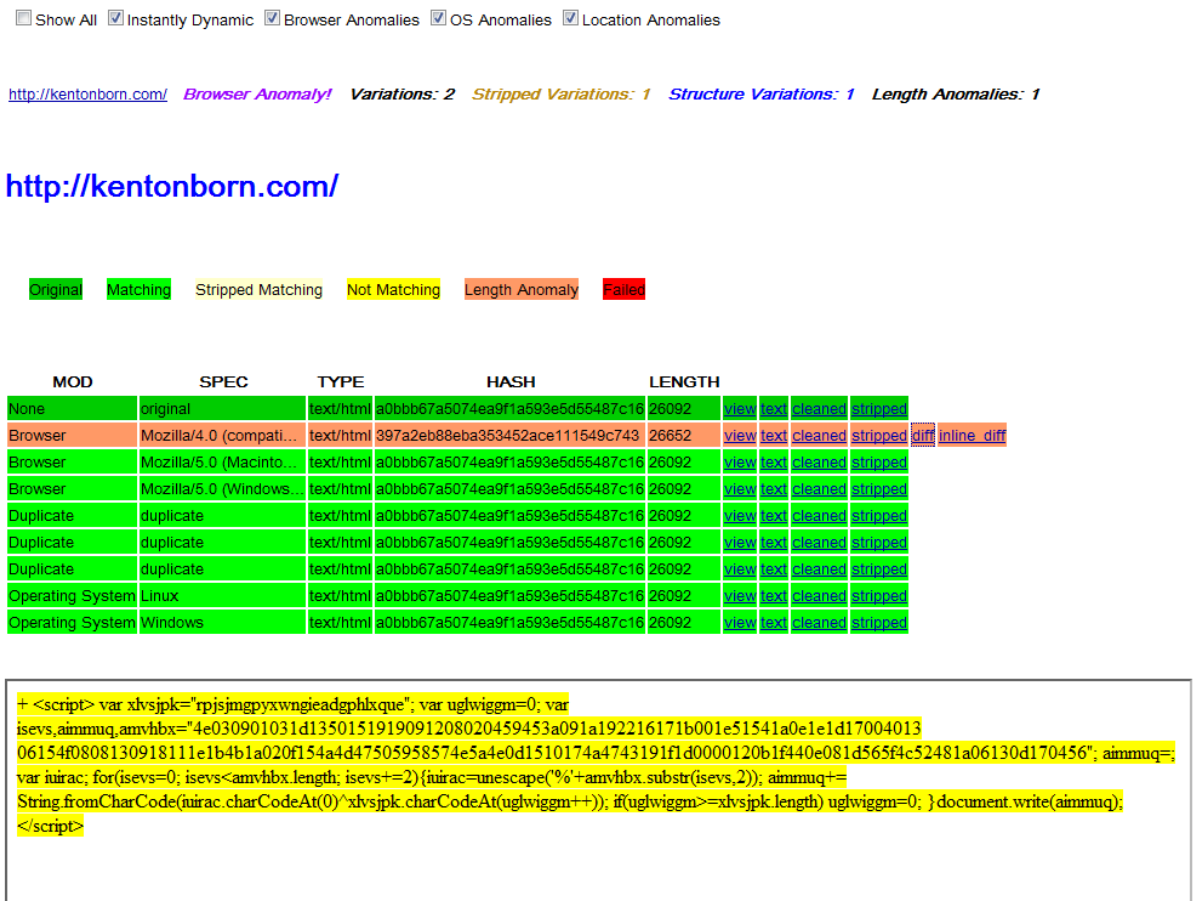



**Figure 33. Obfuscated JavaScript injected for Internet Explorer**

Analysis of the responses instantly triggered a "browser anomaly". The response to the Internet Explorer-based request was the only one that did not match the original response. When the user clicks on the "diff" option, the injected JavaScript is seen in yellow, showing that it did not exist in the original response.

Examining the injected JavaScript provides several hints of its malicious intent. While it is popular to compress JavaScript to an obfuscated form that lowers bandwidth requirements, it is apparent that it is not the case in this scenario. Suspicious JavaScript calls such as "unescape" can be used to mitigate signature detection by masking the malicious content in seemingly non-malicious forms. However, this call itself is often a telling signature itself.

In this scenario, the combination of the multiplexing proxy's automated analysis with manual JavaScript analysis provides the user with all the information necessary to understand the server's behavior. In this case, we know that malicious JavaScript is being sent to clients using an Internet Explorer browser.

### I-frame Injection

Another common form of exploitation on the internet comes from malicious i-frames that are injected into legitimate websites (Patil). These i-frames, often not visible to the user, exploit clients through a resource typically hosted at a cross-domain website.

In this scenario, a hidden, malicious i-frame element has been injected at the bottom of a large html web page. The i-frame loads a resource from a third party domain capable of exploiting Internet Explorer browsers. Once again, the web server is configured to only inject the malicious i-frame when it detects a client vulnerable to the exploit hosted at the third party website.

Below, Figure 34 shows the analysis provided by the multiplexing proxy.

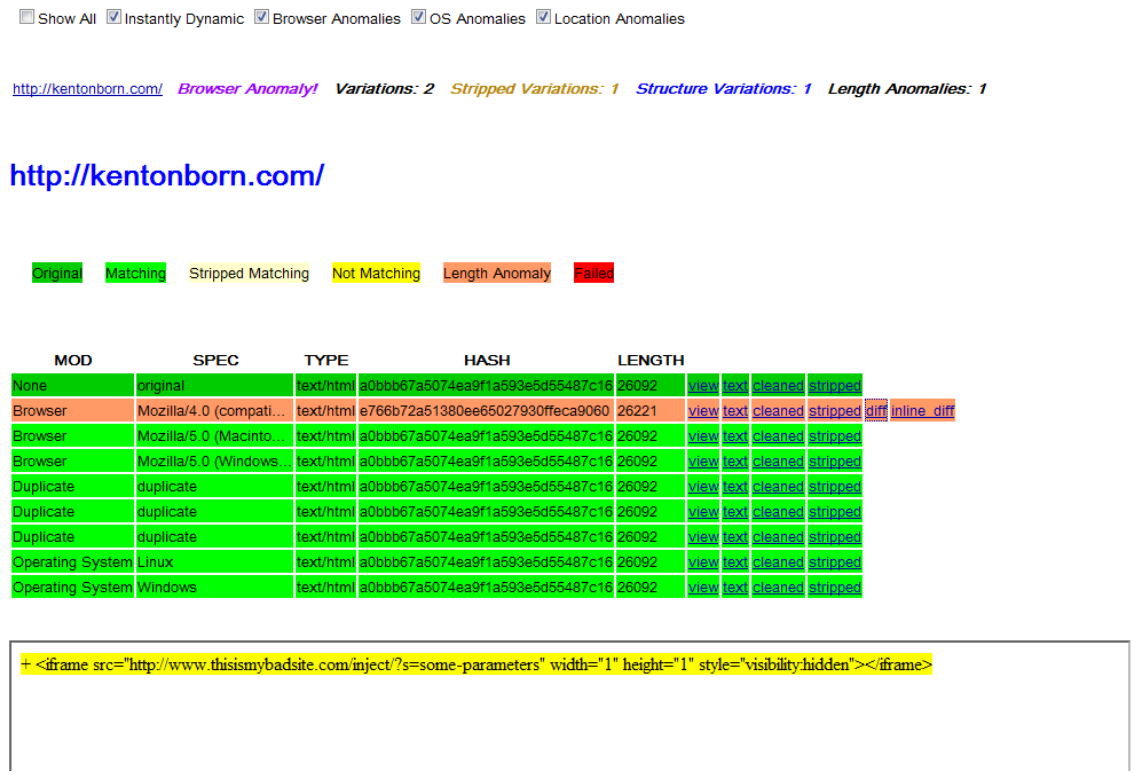



**Figure 34. Output for a maliciously injected i-frame element**

Multiple strategies were used to help conceal the i-frame. Firstly, the visibility of the i-frame was set to hidden. Secondly, the width and length of the i-frame were set to ‘1’, making it effectively the size of a pixel on the screen. However, since the i-frame code was only in the response to the Internet Explorer client fingerprint, a “diff” of the responses immediately identified the malicious i-frame. Manually examining the properties of the injected i-frame element reveals that it is not there for a benign purpose.

Like the malicious JavaScript case study above, malicious i-frames are sometimes obfuscated in order to avoid simple methods of detection. This can either be done through using an alternative encoding, or by including JavaScript that dynamically creates the i-frame element.

**URL Modification**

Many websites load over a hundred resources into the browser when visited. This can make it a daunting task when analyzing the multitude of HTML pages, images, scripts, and stylesheets. Additionally, many content management systems and obfuscation engines will use resource names that are not conducive to easy analysis.

This scenario depicts how the multiplexing proxy’s analysis would look when a webpage makes the most minimal change possible: modifying one character. By making this small change to a URL, a benign resource can be replaced with a malicious resource without having a significant visual impact on the markup or code. Additionally, the overall length of the response remains unchanged. This scenario applies that strategy to a seemingly harmless Cascading Style Sheet (CSS) in order to emulate vulnerabilities similar to CVE-2010-3971 (NVD). According to the NVD database, versions six through eight of Internet Explorer had vulnerable CSS parser that allowed remote attackers to execute arbitrary code on the system.

Figure 35, below, shows the results of analyzing the modified website using the multiplexing proxy.

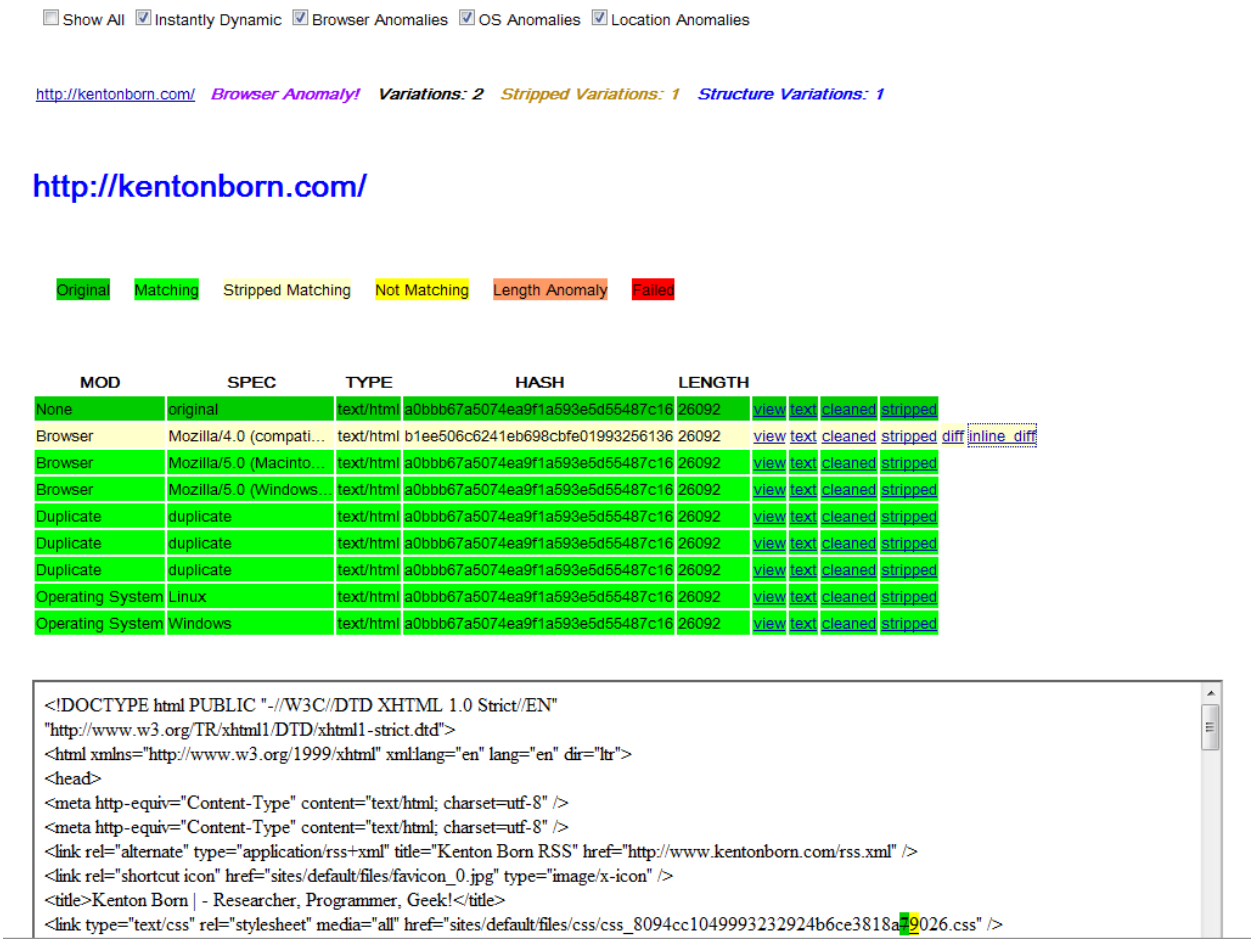



**Figure 35. Output for a maliciously modified URL**

The most noticeable difference between the results in this scenario with previous scenarios is that the color of the modified response is different from previous examples. Since the only change in the content was transforming a ‘7’ into a ‘9’ in the URL, there was not a length anomaly in the malicious response. Additionally, it is not shown in yellow like visible modifications because the change was only to an attribute of an element instead of a change that can be readily seen by the end user. Because of this, the multiplexing proxy classified it as “stripped matching”, meaning that the content matches when you simply look at the content visible to the end user. By analyzing it further with an “inline diff”, the user can see the modification to the URL. While this is not evidence enough to prove malicious intent, it would then be possible to investigate the two versions of the cascading style sheet and find an anomalous modification.

One issue with detecting this type of modification, however, is when the website also has a significant amount of instantly dynamic content. With a large amount of varying content, the anomalous link will be buried among several benign differences. This will result in the resource being classified as instantly dynamic instead of as a browser-based anomaly.

**Maliciously Crafted Image**

Vulnerabilities are occasionally found in the way browsers implement the parsers that handle specific file types. Historically, this has been the cause of many different types of attack, including denial of service attacks (DOS attacks) and arbitrary code execution.

This scenario depicts what the multiplexing proxy would display if a malicious server were to attempt to exploit a vulnerability similar to CVE-2005-2308. According to the National Vulnerability Database, Microsoft Internet Explorer had a vulnerable JPEG decoder that allowed a crafted JPEG image to cause a denial of service and possibly execute arbitrary code (NVD). In order to emulate a crafted image that can execute arbitrary code, shellcode has been injected into a previously legitimate image on the website. Depending on whether or not the client appears vulnerable, the web server will choose to either send the legitimate or crafted image.

Figure 36, below, depicts the output of the multiplexing proxy after visiting the website.

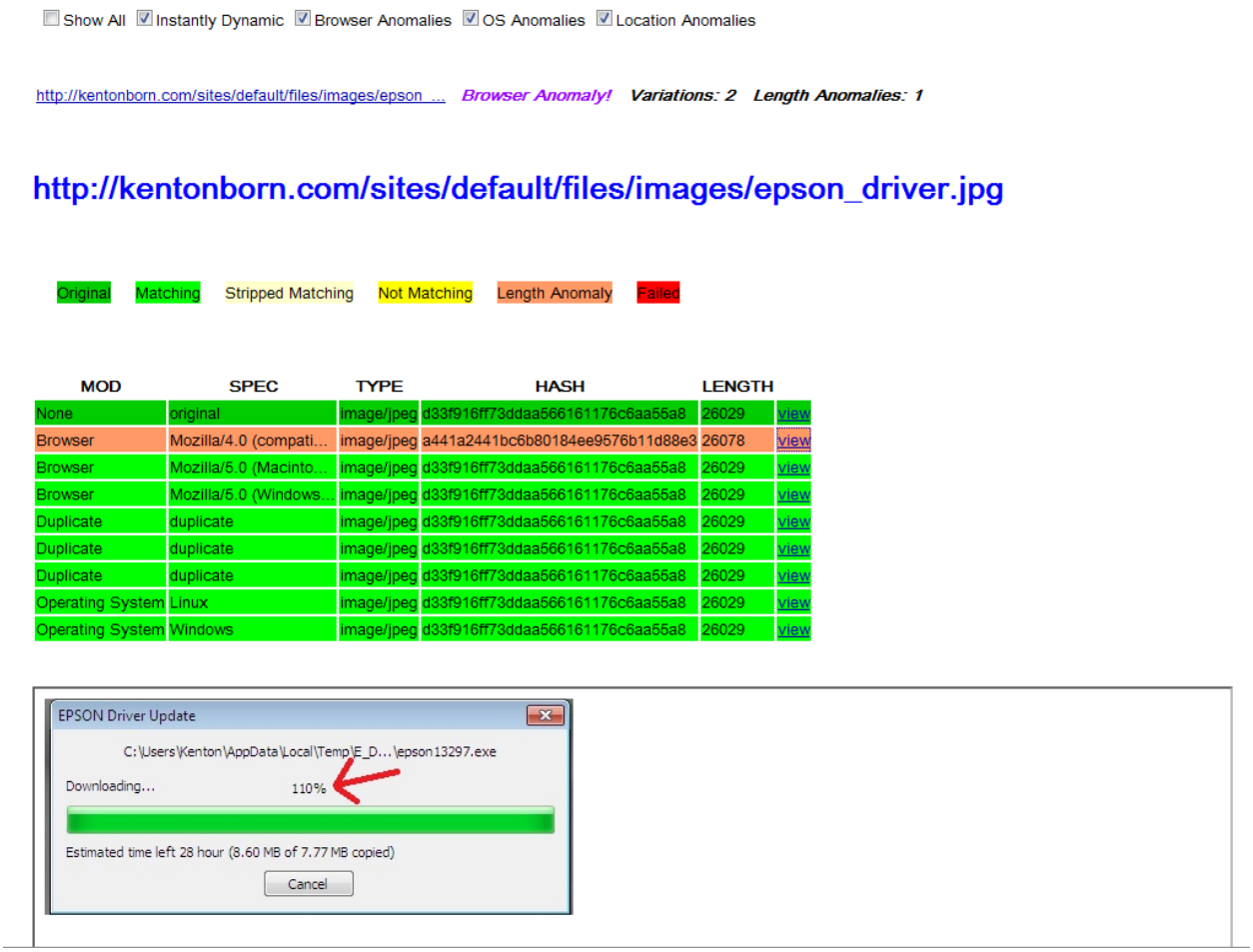



**Figure 36. Output for a maliciously crafted image**

Similar to the previous scenario, we can see that the proxy had no issue detecting a difference in content between the vulnerable and non-vulnerable client fingerprints. In this case, since the shellcode was injected at the end of the JPEG, it actually did not affect how the image displayed to the client. However, most maliciously crafted images will not render correctly, and will be visually identifiable as well.

The multiplexing system would also succeed in detecting steganographic material only sent to specific client fingerprints. Because the byte-to-byte comparison between the image with embedded content and the original response would not match, the multiplexing proxy would easily be able to flag the anomalous response. However, if every image contained the same steganographic material, the multiplexing proxy would have no advantage over other methods of detection.

**Multiple Vulnerabilities on a Dynamic Website**

Every scenario up to this point has focused on manipulating a static resource to exploit a particular vulnerable client. While this is an entirely plausible scenario, exploitation also occurs over significantly more dynamic, complex sites and may have methods of exploiting more than one client fingerprint.

In this case study, the web server delivers a tailored JavaScript exploit to clients using a Macintosh, and it delivers a malicious image to clients using Internet Explorer. Additionally, these vulnerabilities are delivered over a website with dynamic content, often changing from one request to another.

Below, figures 37 and 38 show the output of the multiplexing proxy for the scenario just described.

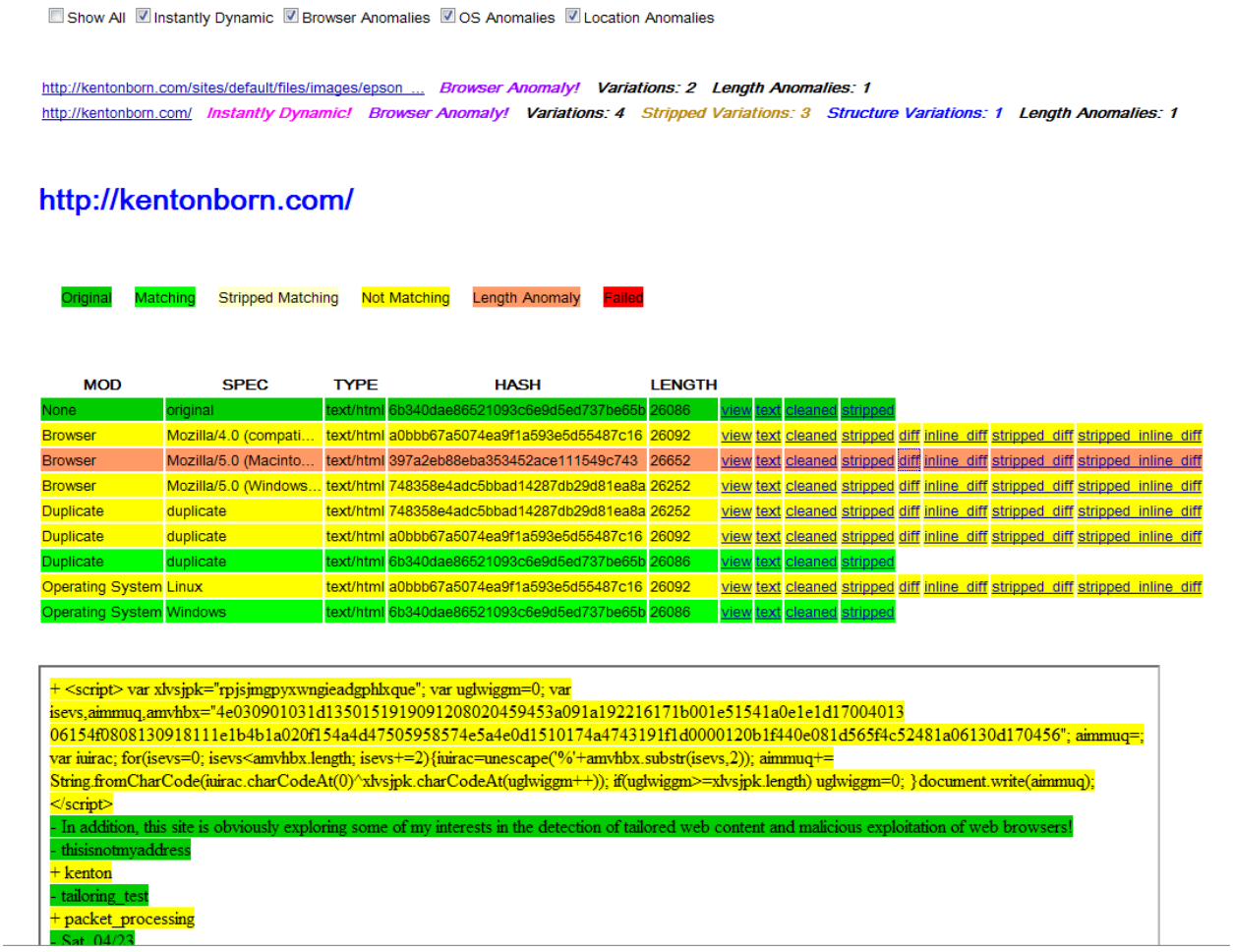



**Figure 37. JavaScript analysis with multiple vulnerabilities in a dynamic website**

A glance at the results is enough to realize that the website is significantly more complex than before. Instead of simply reporting one resource as having a browser anomaly, there are two resources flagged as anomalous. While the first anomaly looks similar to the image scenario above, the second resource is not only flagged as a browser anomaly, but is also flagged as instantly dynamic and has four variations.

Figure 37 shows the multiplexing proxy's analysis of the resource with dynamic content. The first thing to notice is that while two of the multiplexed requests had a response that matched the original, two of the duplicate requests had a varying response. Therefore, while not every response had unique content, it was straightforward to see that there were multiple variations of the website that could be returned for any particular client fingerprint. Because of this, the resource was also classified as instantly dynamic instead of simply as tailored toward a specific client fingerprint.

Despite being instantly dynamic, an anomaly was detected in the length of the response to the client with a Macintosh fingerprint. While various lengths were seen for each of the instantly dynamic responses, the response to the Macintosh-based client was slightly larger than the rest. Figure 38, below, shows how the "diff" tool highlights the obfuscated JavaScript once again. However, unlike the previous case study, there is also a significant number of benign modifications shown below the JavaScript attributed to the dynamic content that originally made the website classify as instantly dynamic. While the malicious content of the website was not as easily seen as before, the multiplexing proxy still provided a simple, straight-forward approach to discovering and analyzing the anomaly. Due to the instantly dynamic content, however, a less significant malicious change, such as a URL modification, would likely have slipped through without triggering the length anomaly.

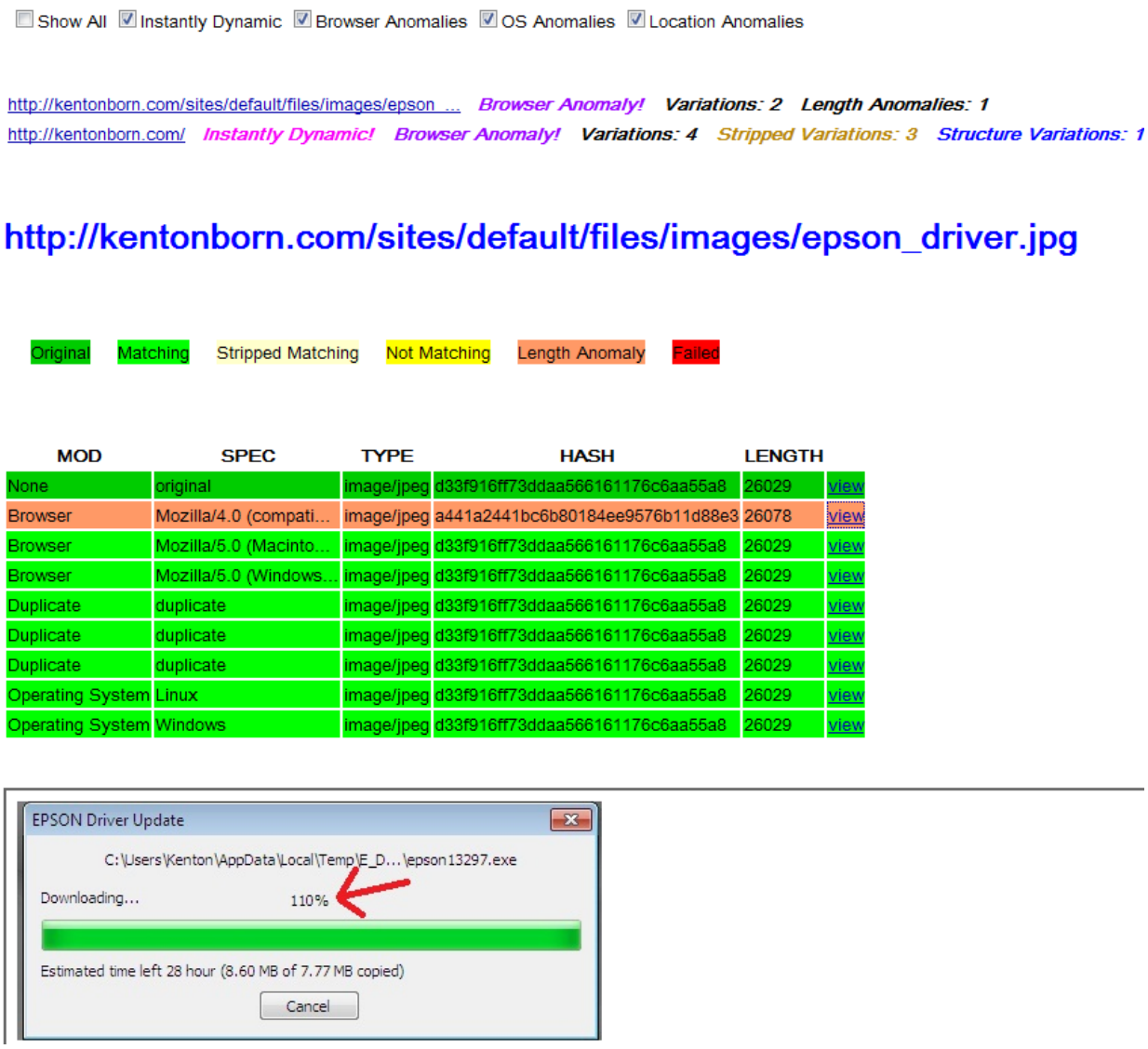



**Figure 38. Image analysis with multiple vulnerabilities in a dynamic website**

Clicking on the image-based browser anomaly revealed similar results to the maliciously crafted image case study described earlier. Since it is the original web page that was instantly dynamic as opposed to the image itself, there were no significant differences in the results from the more simplistic scenario. Immediately, it can be seen that the malicious image was once again delivered to the client with an Internet Explorer fingerprint.

**Malicious Proxy**

The multiplexing proxy has a unique ability to detect malfunctioning and malicious proxies. With the capability to route requests through proxies in various locations and analyze the results, any anomalous responses that were modified at an intermediary proxy will immediately be noticed when the response is compared to the legitimate responses that were not routed through the malfunctioning device.

Before using new and possibly unreliable proxies located around the world, they were tested against a very simple website to ensure that they were returning the correct response. This task was typically performed using *www.iamawesome.com*, a static website that always returns a 10-character response matching "it's true." In one particular case, a Chinese proxy was used that, instead of routing the request to the correct web server, returned a response often seen from newly configured Apache web servers.

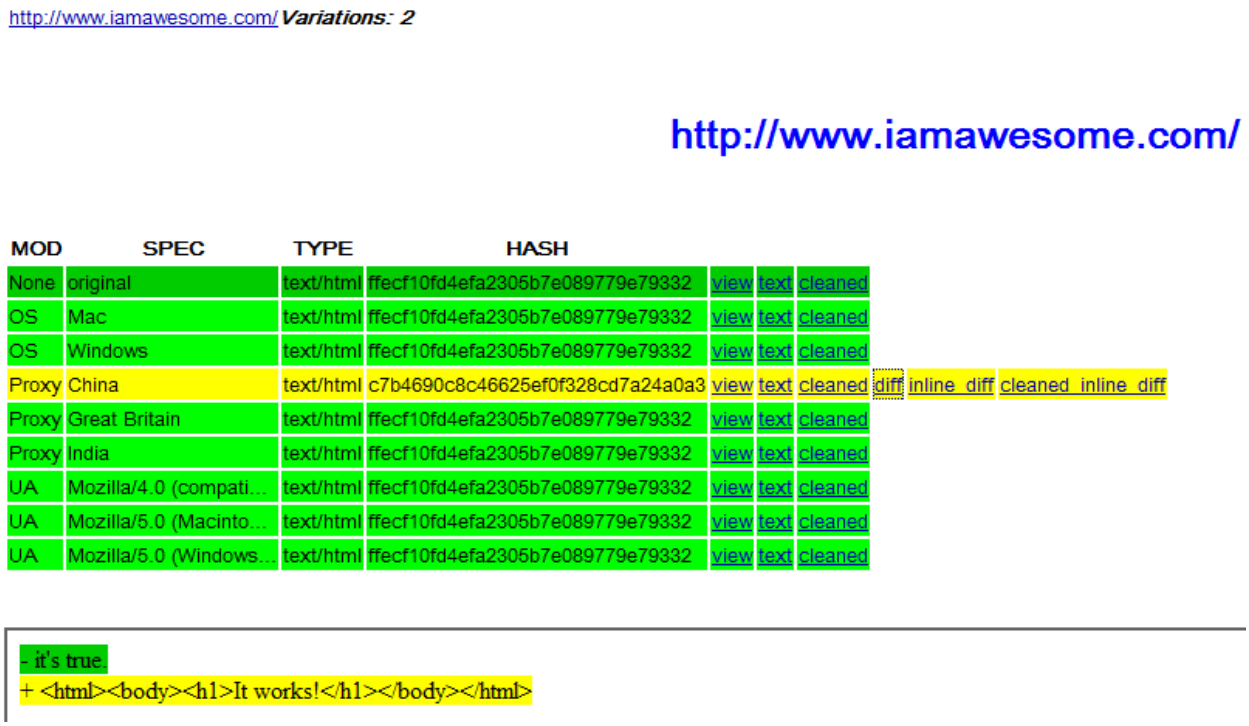



**Figure 39. Chinese proxy returning illegitimate results**

While misconfigured and malicious proxies were not initially planned as part of the cyber analyst study, it is impossible to overlook the power of the tool in this case. With no intention of finding an illegitimate proxy, the tool successfully identified an intermediary device that was not returning the appropriate response. While nothing malicious was done in this case, it shows how easy it would be for an adversary

to modify a legitimate response for their own means, and how simple it would be for the multiplexing proxy to identify it.

**Summary**

While the first four case studies used a relatively simple website, they provide a solid understanding of the type of malicious content that can be discovered and investigated by a cyber analyst using the multiplexing proxy. In the fifth scenario, similar strategies are shown to work for significantly more complex and dynamic websites that may attempt to exploit multiple vulnerabilities. Lastly, the illegitimate proxy detection shows the true power of using the tool both inside and outside the testing environment.

The biggest weakness in using the multiplexing proxy to investigate malicious websites is the inability to easily detect malicious content that is delivered to every client fingerprint. When every client receives the same response, the multiplexing proxy does not have an advantage over previously developed methods of detecting malicious content such as script and obfuscation analysis.

Despite this weakness, it is apparent that the multiplexing proxy is a valuable tool that should be added to a cyber analyst's arsenal. The ability to detect this type of stealthy exploitation is a new capability that is not currently available. Combining this approach with past efforts for detecting static exploitation will yield a significantly greater capability than either approach can achieve separately. Additionally, these detection capabilities can be combined with preventative measures such as the Google Safe Browsing API to provide a well-rounded defense that can both prevent old attacks and discover new ones (SFB).

**REVERSE ENGINEER**

There are many different scenarios that require a reverse engineer to better understand the behavior of a web server. However, access to the server's source code is not always available. Due to the rise in instantly dynamic content, static analysis of the response to a simple HTTP request is not always sufficient to obtain the entire picture.

While instantly dynamic data was mostly ignored in previous studies, a reverse engineer is still interested in this behavior. Therefore, this study will investigate what additional information can be gleamed by examining both instantly dynamic and tailored data. In this study, the multiplexing proxy will be used to revisit many of the complex websites from the empirical classification study. Both useful and benign findings will be discussed in an attempt to determine whether or not the multiplexing proxy can provide additional, useful information that is not easily obtainable without using the multiplexing strategy. Additionally, since location-based tailoring was already heavily analyzed, it will be left out of this study in order to focus more on browser and operating system anomalies.

For each of the following case studies, the multiplexing proxy configuration consists of three duplicate requests, a modified browser fingerprint of Internet Explorer 6, Internet Explorer 8, Firefox 3.0.11, Safari 4.0.2, and a modified operating system fingerprint of Linux and OpenBSD.

*The next section requires an understanding of the tool's output and analysis.*
*Please see "Appendix A" for this information.*

## Content and Advertisement Tailoring

### *www.microsoft.com*

**http://www.microsoft.com/en-us/default.aspx**

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=utf-8 | 8eba329a044e8cdd2d543102a8d11903 | 212875 | view | text | cleaned | stripped | | | | |
| Browser | Mozilla/4.0 (compati... | text/html; charset=UTF-8 | 2ebb3458049689d2c1a9507ba80d9d2f | 166 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/4.0 (compati... | text/html; charset=utf-8 | 38dd30c16b5b70996f82f33f4e67db95 | 212135 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/5.0 (Macinto... | text/html; charset=utf-8 | 451aa053995ef9938d3603b86cc2d13b | 212611 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/5.0 (Windows... | text/html; charset=utf-8 | 9ad018d501ef26ad569abb5c96059c1b | 212733 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8 | 8eba329a044e8cdd2d543102a8d11903 | 212875 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | text/html; charset=utf-8 | 8eba329a044e8cdd2d543102a8d11903 | 212875 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | text/html; charset=utf-8 | 8eba329a044e8cdd2d543102a8d11903 | 212875 | view | text | cleaned | stripped | | | | |
| Operating System | Linux | text/html; charset=utf-8 | 8eba329a044e8cdd2d543102a8d11903 | 212875 | view | text | cleaned | stripped | | | | |
| Operating System | OpenBSD | text/html; charset=utf-8 | 8eba329a044e8cdd2d543102a8d11903 | 212875 | view | text | cleaned | stripped | | | | |


**Figure 40. Browser-based tailoring from www.microsoft.com**

Figure 40 shows an obvious case of browser-based tailoring. The only responses that did not match the original are from requests with a modified User-Agent string. Analyzing the differences more closely shows that the responses are not only tailored toward the browser, but also toward the operating system specified in the User-Agent string!

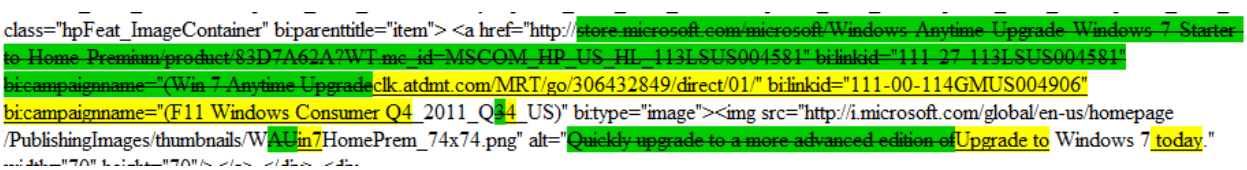



**Figure 41. Tailoring based on the operating system specified in the User-Agent string**

bi:linkid="111-27_113LSUS00519500-112SUS03899" bi:campaignname="(Windows 7 SP1 Download_2011_Q3_USPC Scout(Windows Consumer))" bi:index="0" bi:type="secondarycta" href="http://windowsww.microsoft.com/en-US/windows7/learn-how-to-install-windows-7-service-pack-1-sp1">Install Windows 7 Service Pack 1 today/pc-scout/"> Looking for a new PC? Start here<span

**Figure 42. Additional tailoring based on the operating system in the User-Agent string**

Figures 41 and 42 show how content in the response was tailored for the client based on the operating system specified in the User-Agent string. In the original request, Windows 7 was specified in the User-Agent string, and the response suggests upgrading to service pack 1. However, the response based on the Safari/Macintos- based client fingerprint instead asks the user to upgrade to Windows 7 (Figure 41) and to get a new PC (Figure 42).

Looking back at Figure 40, one also sees a drastically reduced size in the response to the Internet Explorer 6 client fingerprint. Figure 43, below, shows the results of a "diff" operation on this response.

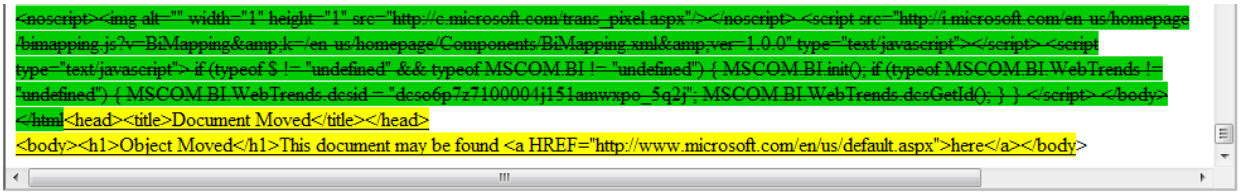
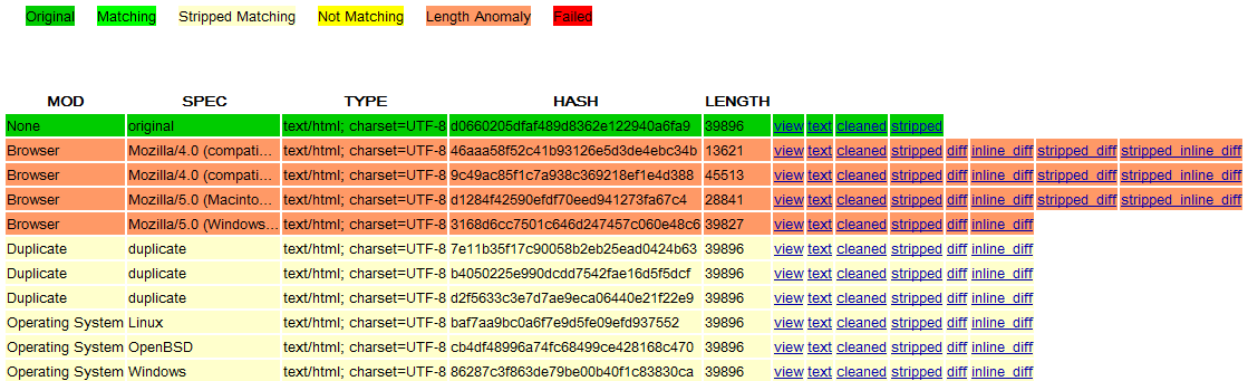

<noscript><img alt="" width="1" height="1" src="http://c.microsoft.com/trans_pixel.aspx"/></noscript><script src="http://i.microsoft.com/en-us/homepage/bimapping.js?v=BiMapping&lc=/en-us/homepage/Components/BiMapping.xml&ver=1.0.0" type="text/javascript"></script><script type="text/javascript">if (typeof $ != "undefined" && typeof MSCOM.BI != "undefined") { MSCOM.BI.init(); if (typeof MSCOM.BI.WebTrends != "undefined") { MSCOM.BI.WebTrends.dcsid = "dcso6p7z7100004j151amwxpo_5q2j"; MSCOM.BI.WebTrends.dcsGetId(); } }</script></body></html><head><title>Document Moved</title></head>
<body><h1>Object Moved</h1>This document may be found <a HREF="http://www.microsoft.com/en/us/default.aspx">here</a></body>

**Figure 43. Redirect for Internet Explorer 6 clients**

Clients using the Internet Explorer 6 were redirected to a website that has a slightly different URL. More specifically, clients using Internet Explorer 6 are forwarded from *www.microsoft.com/**en-us**/default.aspx* to *www.microsoft.com/**en/us**/default.aspx*. It is likely that they created a custom page for Internet Explorer 6 because of the many DOM/JavaScript quirks that are found in that particular version that are not present in most other browsers.

*www.google.com*

http://www.google.com/



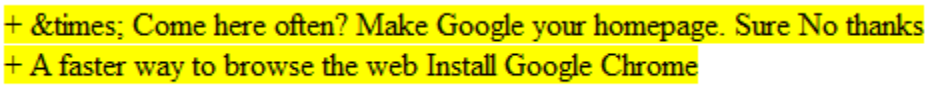
Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=UTF-8 | d0660205dfaf489d8362e122940a6fa9 | 39896 | view | text | cleaned | stripped | | | | |
| Browser | Mozilla/4.0 (compati... | text/html; charset=UTF-8 | 46aaa58f52c41b93126e5d3de4ebc34b | 13621 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/4.0 (compati... | text/html; charset=UTF-8 | 9c49ac85f1c7a938c369218ef1e4d388 | 45513 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/5.0 (Macinto... | text/html; charset=UTF-8 | d1284f42590efdf70eed941273fa67c4 | 28841 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/5.0 (Windows... | text/html; charset=UTF-8 | 3168d6cc7501c646d247457c060e48c6 | 39827 | view | text | cleaned | stripped | diff | inline_diff | | |
| Duplicate | duplicate | text/html; charset=UTF-8 | 7e11b35f17c90058b2eb25ead0424b63 | 39896 | view | text | cleaned | stripped | diff | inline_diff | | |
| Duplicate | duplicate | text/html; charset=UTF-8 | b4050225e990dcdd7542fae16d5f5dcf | 39896 | view | text | cleaned | stripped | diff | inline_diff | | |
| Duplicate | duplicate | text/html; charset=UTF-8 | d2f5633c3e7d7ae9eca06440e21f22e9 | 39896 | view | text | cleaned | stripped | diff | inline_diff | | |
| Operating System | Linux | text/html; charset=UTF-8 | baf7aa9bc0a6f7e9d5fe09efd937552 | 39896 | view | text | cleaned | stripped | diff | inline_diff | | |
| Operating System | OpenBSD | text/html; charset=UTF-8 | cb4df48996a74fc68499ce428168c470 | 39896 | view | text | cleaned | stripped | diff | inline_diff | | |
| Operating System | Windows | text/html; charset=UTF-8 | 86287c3f863de79be00b40f1c83830ca | 39896 | view | text | cleaned | stripped | diff | inline_diff | | |

**Figure 44. Results for www.google.com**

Similar to the open source analyst study, Google once again provided very interesting results. Figure 44 shows significant tailoring based on the User-Agent string in the request. Every modification to the browser fingerprint, even from one version of Firefox to another, resulted in several modifications to the response.

Without question, the most interesting result came from using the "stripped diff" tool on the Internet Explorer 8 response.



+ × Come here often? Make Google your homepage. Sure No thanks
+ A faster way to browse the web Install Google Chrome

**Figure 45. Stripped diff for Internet Explorer 8 on www.google.com**

The "stripped diff" tool shows an advertisement for Google Chrome being included in the response to the request with an Internet Explorer 8 browser fingerprint. While the user may suspect that everyone sees this advertisement, the multiplexing proxy shows that this is not the case!

Looking at the inline differences between the original response and a "stripped matching" response is less interesting. Shown in figure 46, below, Google appears to be dynamically creating a unique URL inside a script element.

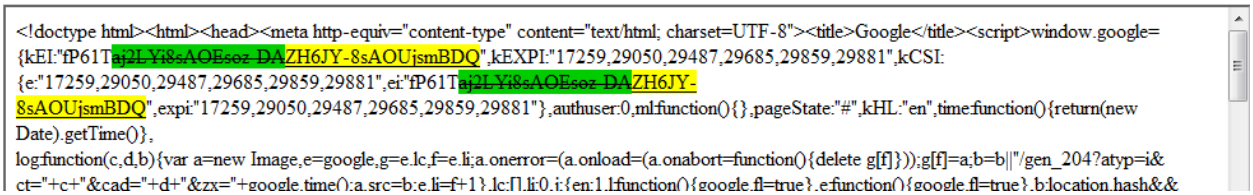



**Figure 46. Inline diff for a stripped matching response for www.google.com**

Investigating the requests with a modified browser fingerprint shows additional differences not seen in the "stripped matching" responses. Even for User-Agent string differences as small as one version of Firefox to the next, several differences were identified.

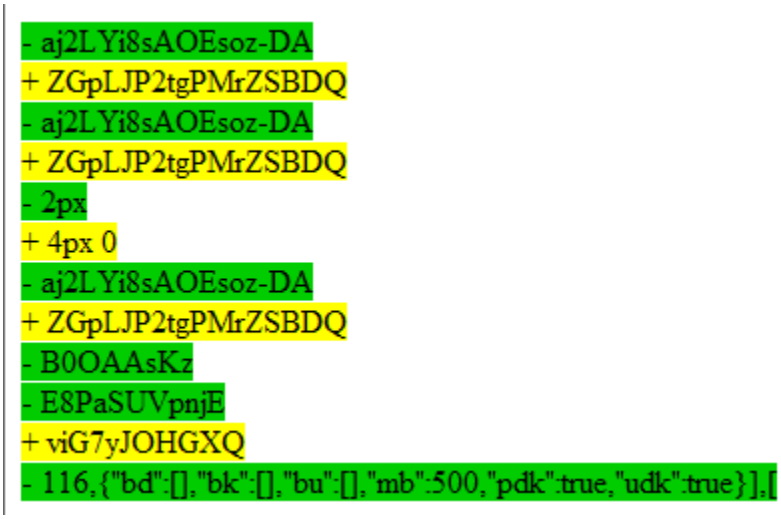



**Figure 47. Unique tailoring between Firefox fingerprints**

While the first few modifications are the script-based dynamic URL discussed previously, several new differences are present in figure 47. One difference that sticks out is a size modification, changing 2px to 4px. Further investigation with an "inline diff" showed this to be a padding measurement. Additionally, the content shown in the last line of the "diff" output was only present in the original response. This was present at the end of some form of pre-computed error dialogue. Lastly, a few smaller sequences of seemingly random characters are seen that required an "inline diff" to investigate further.

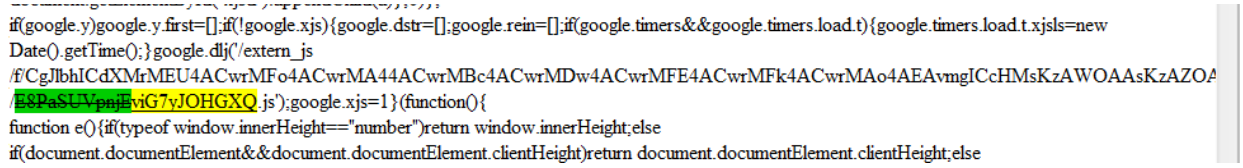



**Figure 48. Tailored JavaScript retrieval for specific browsers**

The “inline diff” showed that there was JavaScript being retrieved that was unique to that particular browser fingerprint. Unlike the “diff” results for “stripped matching” responses that showed a dynamic URL being built for every response, this particular URL only changed when the User-Agent string was modified from the one used in the original request.

Lastly, browser-based tailoring can also be seen in a script resource required by the initial web page.

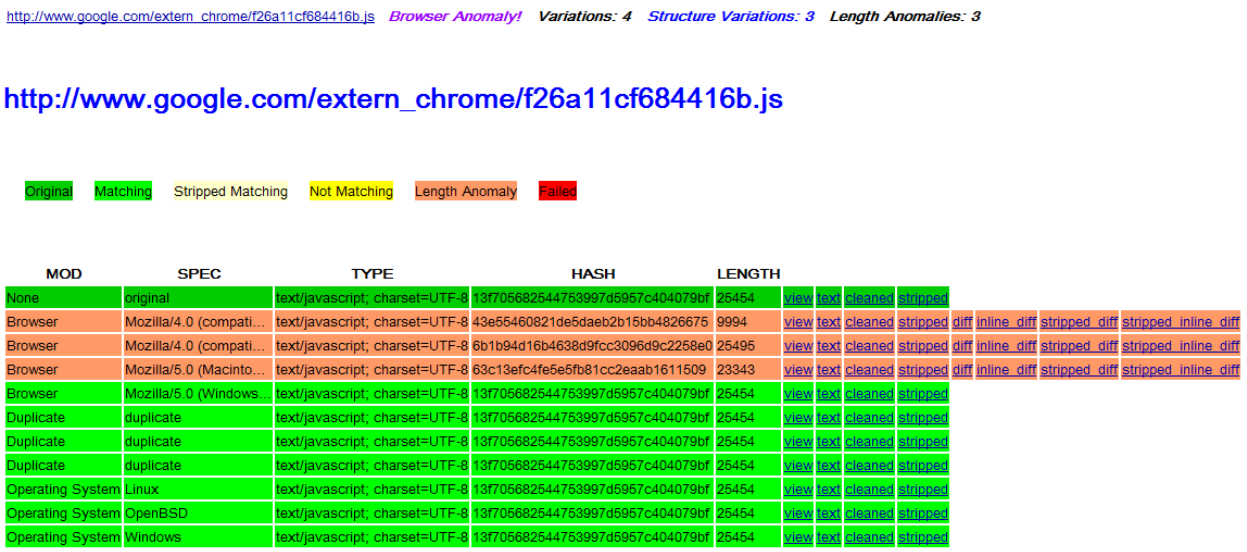



**Figure 49. Tailored JavaScript from an identical URL**

Figure 49 clearly shows that the JavaScript is tailored based on the browser of the connecting client. Unlike the initial web page, however, clients with varying versions of Firefox received the same response. Safari and both versions of Internet Explorer, on the other hand, all received different responses.

The multiplexing proxy proved to be an invaluable resource for identifying server-side behavior that could not have been discovered, otherwise. Without its unique investigative approach, it is likely that Google’s server-side behavior could not have been fully analyzed.

*www.skype.com*

**http://www.skype.com/intl/en-us/home**

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=utf-8 | cb69f7d33e09bca64b67e45f31f6d064 | 72093 | view | text | cleaned | stripped | |
| Browser | Mozilla/4.0 (compati... | text/html; charset=utf-8 | cb69f7d33e09bca64b67e45f31f6d064 | 72093 | view | text | cleaned | stripped | |
| Browser | Mozilla/4.0 (compati... | text/html; charset=utf-8 | cb69f7d33e09bca64b67e45f31f6d064 | 72093 | view | text | cleaned | stripped | |
| Browser | Mozilla/5.0 (Macinto... | text/html; charset=utf-8 | c16f68ad1a664900414a788033ffc151 | 71902 | view | text | cleaned | stripped | diff inline_diff |
| Browser | Mozilla/5.0 (Windows... | text/html; charset=utf-8 | cb69f7d33e09bca64b67e45f31f6d064 | 72093 | view | text | cleaned | stripped | |
| Duplicate | duplicate | text/html; charset=utf-8 | cb69f7d33e09bca64b67e45f31f6d064 | 72093 | view | text | cleaned | stripped | |
| Duplicate | duplicate | text/html; charset=utf-8 | cb69f7d33e09bca64b67e45f31f6d064 | 72093 | view | text | cleaned | stripped | |
| Duplicate | duplicate | text/html; charset=utf-8 | cb69f7d33e09bca64b67e45f31f6d064 | 72093 | view | text | cleaned | stripped | |
| Operating System | Linux | text/html; charset=utf-8 | cb69f7d33e09bca64b67e45f31f6d064 | 72093 | view | text | cleaned | stripped | |
| Operating System | OpenBSD | text/html; charset=utf-8 | cb69f7d33e09bca64b67e45f31f6d064 | 72093 | view | text | cleaned | stripped | |

{ index : "2",

"imageUri": "http://www.skypeassets.com/content/dam/skype/images/skinny/group-video-calls-windowsvc-hero-mac-skinny2.jpg",

**Figure 50. Tailored image for the Mac browser**

Figure 50 shows Skype delivering a tailored image about video calls to clients using a Mac. Figure 51, below, similarly shows custom videos for Mac clients that are not seen in the responses for other client fingerprints.

'videoH264Url': 'http://download.skype.com/share/videos/windows-mac/video-call-download-skype-videocall-enen-mac.mp4',
'videoUrl': 'http://download.skype.com/share/videos/windows-mac/video-call-download-skype-videocall-enen-mac.flv',
'autoPlay': true

**Figure 51. Tailored videos for the Mac browser**

While some websites tailor images in order to handle Internet Explorer 6 transparency issues, the images and videos on this site were changes in content as opposed to compatibility fixes. Due to the largely static content of the website, the Mac-based anomaly was easily identifiable.

**Incompatibility**

***www.ign.com***

**http://rpm.newrelic.com/eum/eum.js1**

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | application/x-javascript | 4ad0a2fbd56efdcf82dde55f8b5f9a7b | 8062 | view | text | cleaned | stripped | | | | |
| Browser | Mozilla/4.0 (compati... | text/html | 788deb2bca865533ad8c556689b8022 | 3776 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/4.0 (compati... | application/x-javascript | 4ad0a2fbd56efdcf82dde55f8b5f9a7b | 8062 | view | text | cleaned | stripped | | | | |
| Browser | Mozilla/5.0 (Macinto... | application/x-javascript | 4ad0a2fbd56efdcf82dde55f8b5f9a7b | 8062 | view | text | cleaned | stripped | | | | |
| Browser | Mozilla/5.0 (Windows... | application/x-javascript | 4ad0a2fbd56efdcf82dde55f8b5f9a7b | 8062 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | application/x-javascript | 4ad0a2fbd56efdcf82dde55f8b5f9a7b | 8062 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | application/x-javascript | 4ad0a2fbd56efdcf82dde55f8b5f9a7b | 8062 | view | text | cleaned | stripped | | | | |
| Duplicate | duplicate | application/x-javascript | 4ad0a2fbd56efdcf82dde55f8b5f9a7b | 8062 | view | text | cleaned | stripped | | | | |
| Operating System | Linux | application/x-javascript | 4ad0a2fbd56efdcf82dde55f8b5f9a7b | 8062 | view | text | cleaned | stripped | | | | |
| Operating System | OpenBSD | application/x-javascript | 4ad0a2fbd56efdcf82dde55f8b5f9a7b | 8062 | view | text | cleaned | stripped | | | | |

```
<div class="container clearfix">
<h1>Unsupported Browser</h1>
<div class="span-10">
<p>We're sorry, New Relic requires Internet Explorer 7 or higher, Chrome 7 or higher, Firefox 3 or higher, or Safari 4 or higher.</p>

<h2>Supported Browser Downloads</h2>

<p>The latest versions of supported browsers can be downloaded below.</p>
<p><a href="http://www.google.com/chrome/">Chrome</a></p>
<p><a href="http://www.mozilla.com/firefox/">Firefox</a></p>
<p><a href="http://www.microsoft.com/windows/internet-explorer/default.aspx">Internet Explorer</a></p>
<p><a href="http://www.apple.com/safari/">Safari</a></p>
</div>
</div>
```

**Figure 52. Tailored request for upgrading the browser**

Internet Explorer 6 is one of the most difficult browsers to account for when developing a website for cross-browser support. Whenever a particular browser is not supported by a website, there are a few clean methods of handling this situation.

The first, seen above in Figure 52, consists of providing the user with a response that requests the user to upgrade their browser or plugins. The biggest problem with this approach is that it requires the user to take additional steps before viewing the website. Inhibiting the user's ability to immediately receive the content may be enough to drive them away. Another popular approach, seen in the following example, is to redirect the user to a website specifically designed for the incompatible browser.

*www.myspace.com*

http://www.myspace.com/

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=utf-8 | c9d71140ca67102228f39202e143c405 | 47860 | view | text | cleaned | stripped | | | | |
| Browser | Mozilla/4.0 (compati... | text/html; charset=utf-8 | 11a1169665b83e03cc3b1cb61713dc7 | 141 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/4.0 (compati... | text/html; charset=utf-8 | ab43baace3411167f393b1d0ceb4567d | 48374 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/5.0 (Macinto... | text/html; charset=utf-8 | a5024b781453339f41ef8f8933e619f8 | 47862 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/5.0 (Windows... | text/html; charset=utf-8 | f8e73c17ee6caf521030d000a0b4bb05 | 47896 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8 | ff9324cce63e8a34154c2158d12b4ae5 | 47880 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8 | 2229a393ce1d69ea671c9d751fd5582e | 47870 | view | text | cleaned | stripped | diff | inline_diff | | |
| Duplicate | duplicate | text/html; charset=utf-8 | ae39a363cfcb95cb221fe96fe54ce520 | 47858 | view | text | cleaned | stripped | diff | inline_diff | | |
| Operating System | Linux | text/html; charset=utf-8 | ef02a052d4def8053dfd63e9d0962e0a | 47853 | view | text | cleaned | stripped | diff | inline_diff | | |
| Operating System | OpenBSD | text/html; charset=utf-8 | 380ca62e6ab1c4ac76a3e5092c04042 | 47913 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |

**Figure 53. Anomalous lengths for Internet Explorer browsers**

Figure 53 shows length anomalies in the responses to client fingerprints mimicking versions 6 and 8 of the Internet Explorer browser. While both versions triggered an anomaly, each case was unique. The response for the request with an Internet Explorer 6 fingerprint was only 141 bytes, making it drastically shorter than the rest. Investigating this length anomaly showed a response containing links to a help page explaining how the website does not support the browser being used.

```
<script>jQuery.ready()</script>
</body>
html><head><title>Object moved</title></head><body>
<h2>Object moved to <a href="/help/browserunsupported">here</a>.</h2>
</body></html>
```

**Figure 54. Unsupported browser response for Internet Explorer 6**

In contrast, the response for the request with an Internet Explorer 8 fingerprint had more content than the rest. A snapshot of several inline differences is shown in figure 55, below.

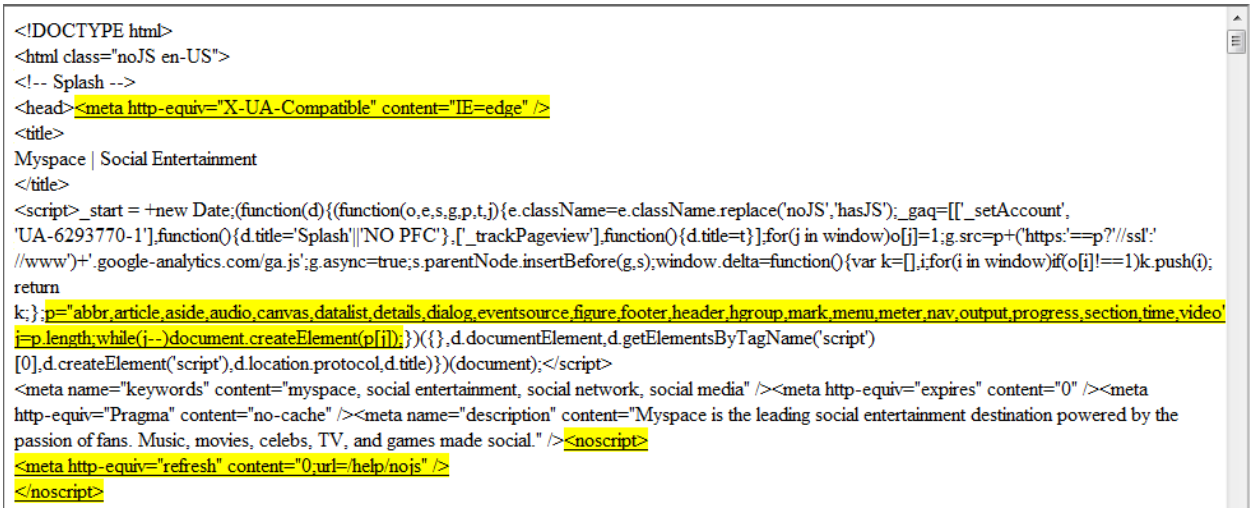



**Figure 55. Injected content due to an Internet Explorer 8 fingerprint**

Further investigation of this response showed an interesting beaconing behavior. Shown in figure 56, below, client fingerprint information was stored in a beacon object that was sent back to the server along with other information including location and session data. While this may have been identified by closely scrutinizing the response, the highlighted differences make it significantly easier to identify the behavior.

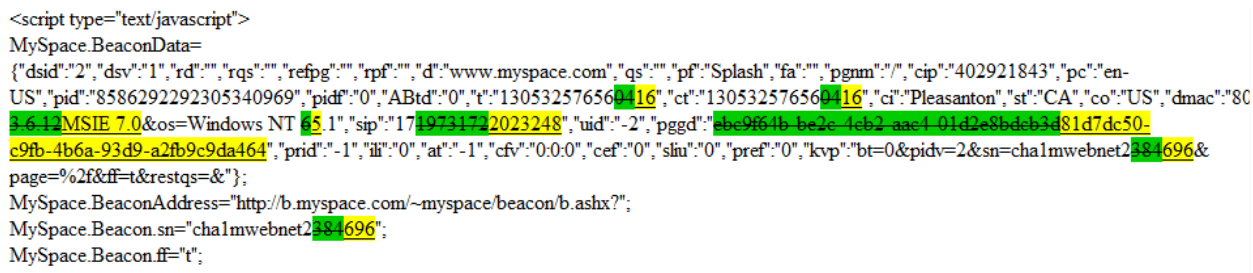



**Figure 56. Client fingerprint information beaconing to the server**

**Format Tailoring**

***www.multiply.com / www.aol.com***

**http://multiply.com/**

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=utf-8 | b81fbc5243d3fde0dd946a7123cd03cd | 35527 | view | text | cleaned | stripped | | |
| Browser | Mozilla/4.0 (compati... | text/html; charset=utf-8 | c6e7d97e96ebeb1f904aeee3b8b22d88 | 35958 | view | text | cleaned | stripped | diff | inline_diff |
| Browser | Mozilla/4.0 (compati... | text/html; charset=utf-8 | a9585eb15a91dde306d51354df4dc808 | 35527 | view | text | cleaned | stripped | diff | inline_diff |
| Browser | Mozilla/5.0 (Macinto... | text/html; charset=utf-8 | 96867e6d8de59f281aefec47fc5a2723 | 35548 | view | text | cleaned | stripped | diff | inline_diff |
| Browser | Mozilla/5.0 (Windows... | text/html; charset=utf-8 | f61e8192cb72d26cdc9e8f6ca75fcd5a | 35384 | view | text | cleaned | stripped | diff | inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8 | 7bdd38af95c358b8abe829e79d39d086 | 35531 | view | text | cleaned | stripped | diff | inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8 | c03162fcbeab83ba5f03c1525be681c5 | 35384 | view | text | cleaned | stripped | diff | inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8 | 32e877c2c242636cf99f913ffe78e52f | 35384 | view | text | cleaned | stripped | diff | inline_diff |
| Operating System | Linux | text/html; charset=utf-8 | 4c51d82fb5243543b599bf8af01846a3 | 35383 | view | text | cleaned | stripped | diff | inline_diff |
| Operating System | OpenBSD | text/html; charset=utf-8 | 6a0e5f82841192fb4b5c659db18eaaf4 | 35384 | view | text | cleaned | stripped | diff | inline_diff |

```
<div class=splash_press>
<div class=splash_pressbox style='width: 250px;'>
<img src='http://images.multiply.com/common/dot_clear.gif'
style="filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='http://images.multiply.com/multiply/logo/external/self-
white.png',sizingMethod='scale')" border=0 align=left width=63 height=37 />
<div class=splash_presstext>Multiply makes sharing your<br>art and stories easy</div>
</div>
<div class=splash_pressbox style='width: 220px;'>
<img src='http://images.multiply.com/common/dot_clear.gif'
style="filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='http://images.multiply.com/multiply/logo/external/usatoday-
white.png',sizingMethod='scale')" border=0 align=left width=63 height=37 />
<div class=splash_presstext>Multiply: The next<br>social-networking wave</div>
</div>
<div class=splash_pressbox style='width: 240px;'>
<img src='http://images.multiply.com/common/dot_clear.gif'
style="filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='http://images.multiply.com/multiply/logo/external/cnet-
white.png',sizingMethod='scale')" border=0 align=left width=63 height=37 />
```

**Figure 57. Image tailoring for Internet Explorer 6**

Some versions of Internet Explorer handle images in the PNG format differently than most browsers, often rendering transparent backgrounds as grey. This issue requires developers to use an AlphaImageLoader filter (.NET framework) to handle PNG images for affected versions of Internet Explorer. Figure 57, above, shows an example of a website that only includes these filters in the responses sent to the client with an Internet Explorer 6 fingerprint. This was not only seen in the initial HTML page, but also in an included CSS and JavaScript resource.

A similar strategy was used by AOL for Internet Explorer 6 and Internet Explorer 8. This can be seen in Figure 58, below.

http://www.aol.com/

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html;;charset=utf-8 | 4bfee8d5b2159583ef04bff0fc41498f | 70927 | view | text | cleaned | stripped | | | | |
| Browser | Mozilla/4.0 (compati... | text/html;;charset=utf-8 | e6a8be44dc299f03f206f56a1126cfb4 | 73253 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/4.0 (compati... | text/html;;charset=utf-8 | 29449c86460c8d830e79eb136203ebcb | 72341 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/5.0 (Macinto... | text/html;;charset=utf-8 | ed3a4f3d74954831f4aad616d9cb862c | 70895 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/5.0 (Windows... | text/html;;charset=utf-8 | d3297077807a25a0e1fc184e49e2fd28 | 70917 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html;;charset=utf-8 | 88891bd21479b243a236e5c95dd8e44f | 70942 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html;;charset=utf-8 | 3c2822cc53c3026896dd77479c152abe | 70918 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html;;charset=utf-8 | c80c40dbf288aa474428845876c14369 | 70951 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Operating System | Linux | text/html;;charset=utf-8 | 6665e384ddde9dc80ac7aa6683051fc1 | 70906 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Operating System | OpenBSD | text/html;;charset=utf-8 | af9dbf77b6bf13be0a19bea430dcff19 | 70894 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |

```
/* daily vj headline swap */
#dailyvj .header, .dailyvj .header {border-bottom:none; top:160px; padding-bottom: 0px;}
.vjvideo {position:absolute; top:0;}
.vj-playlist {margin-top:160px;}
/* end: daily vj headline swap */

</style><head>
<body class="pinkink">style>#header_logo{filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='http://portal.aolcdn.com/p5/_v42.5/css/ie
/logo_IE.png',sizingMethod='crop')}
.vbtn-middle1 .dlvideo{filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='http://portal.aolcdn.com/p5/_v42.5/css/ie
/pinkPlayButtonHover.png'),sizingMethod='crop')}
.vbtn-middle2 .dlvideo{filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='http://portal.aolcdn.com/p5/_v42.5/css/ie
/bluePlayButtonHover.png'),sizingMethod='crop')}
.vbtn-middle3 .dlvideo{filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='http://portal.aolcdn.com/p5/_v42.5/css/ie
/grayPlayButtonHover.png'),sizingMethod='crop')}</style></head>
<body class="space_chase">
```

**Figure 58. Injected code for handling Internet Explorer PNG inconsistencies**

Less obvious tailoring can be seen in other portions of the web page. Figure 59, below, shows the class of an element being modified to account for the client's browser.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns:og="http://opengraphprotocol.org/schema/" xmlns:fb="http://www.facebook.com/2008/fbml" xmlns="http://www.w3.org/1999/xhtml"
xml:lang="en" lang="en" class="Gecko ff3 ff35 ff36IE8">
```

**Figure 59. Element class modification based on browser**

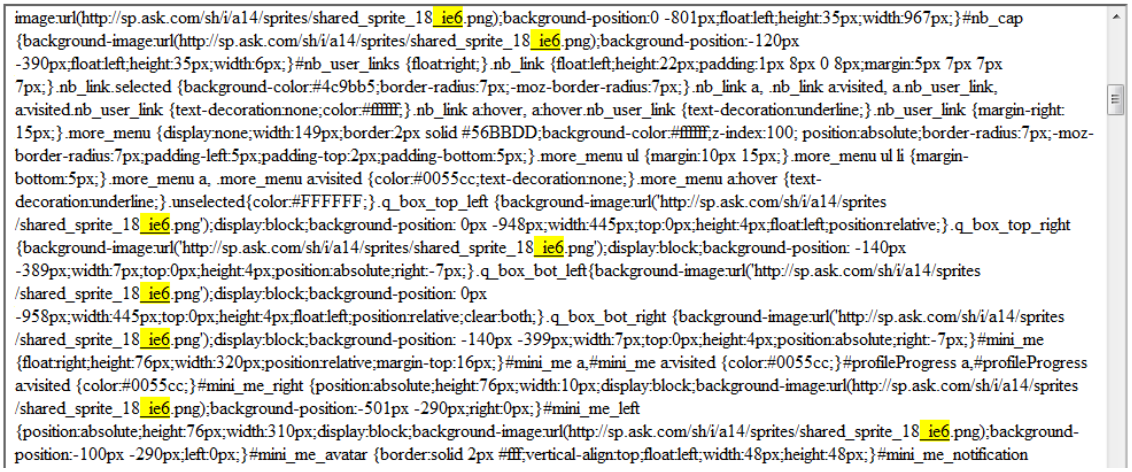
It is likely that there were CSS statements specifically defined for each browser type that would or would not be invoked depending on the browser specified in the class variable. Therefore, while the CSS document itself would not be tailored, it would contain everything necessary to allow the dynamic modifications to the original web page to affect the CSS statements that affect the output.

*www.ask.com*

**http://www.ask.com/o=0&l=dir**

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html;charset=UTF-8 | c08c796713861fac9d6b0484e43a4a01 | 85996 | view | text | cleaned | stripped | | | |
| Browser | Mozilla/4.0 (compati... | text/html;charset=UTF-8 | bda7cbdd52814dc74888ee44b86bfcd4 | 86456 | view | text | cleaned | stripped | diff | inline_diff | stripped_d |
| Browser | Mozilla/4.0 (compati... | text/html;charset=UTF-8 | 6ccf06e6de7ecf0eecc3c45ddaac9600 | 86145 | view | text | cleaned | stripped | diff | inline_diff | stripped_d |
| Browser | Mozilla/5.0 (Macinto... | text/html;charset=UTF-8 | 5205f0d2a98db0db8b3cfaa6bb757910 | 85900 | view | text | cleaned | stripped | diff | inline_diff | |
| Browser | Mozilla/5.0 (Windows... | text/html;charset=UTF-8 | 61f90dd44e40552a61b2ce7f75a87995 | 85880 | view | text | cleaned | stripped | diff | inline_diff | |
| Duplicate | duplicate | text/html;charset=UTF-8 | 12f86ffaa69fc21c320f0e897f4f235b | 85860 | view | text | cleaned | stripped | diff | inline_diff | |
| Duplicate | duplicate | text/html;charset=UTF-8 | 5599ad9584f3f2acf2018f7c52ff4cfe | 85870 | view | text | cleaned | stripped | diff | inline_diff | |
| Duplicate | duplicate | text/html;charset=UTF-8 | ecb57f1f1e088cbe49976a17fc49a711 | 85984 | view | text | cleaned | stripped | diff | inline_diff | |
| Operating System | Linux | text/html;charset=UTF-8 | cf7910e07228d5ab9fcec33dbfe54dff | 85870 | view | text | cleaned | stripped | diff | inline_diff | |
| Operating System | OpenBSD | text/html;charset=UTF-8 | 213d5fb8415cf8d0448a6f51684cd094 | 85871 | view | text | cleaned | stripped | diff | inline_diff | |



```
image:url(http://sp.ask.com/sh/i/a14/sprites/shared_sprite_18_ie6.png);background-position:0 -801px;float:left;height:35px;width:967px;}#nb_cap
{background-image:url(http://sp.ask.com/sh/i/a14/sprites/shared_sprite_18_ie6.png);background-position:-120px
-390px;float:left;height:35px;width:6px;}#nb_user_links {float:right;}.nb_link {float:left;height:22px;padding:1px 8px 0 8px;margin:5px 7px 7px
7px;}.nb_link.selected {background-color:#4c9bb5;border-radius:7px;-moz-border-radius:7px;}.nb_link a, .nb_link a:visited, a.nb_user_link,
a:visited.nb_user_link {text-decoration:none;color:#ffffff;}.nb_link a:hover, a:hover.nb_user_link {text-decoration:underline;}.nb_user_link {margin-right:
15px;}.more_menu {display:none;width:149px;border:2px solid #56BBDD;background-color:#ffffff;z-index:100; position:absolute;border-radius:7px;-moz-
border-radius:7px;padding-left:5px;padding-top:2px;padding-bottom:5px;}.more_menu ul {margin:10px 15px;}.more_menu ul li {margin-
bottom:5px;}.more_menu a, .more_menu a:visited {color:#0055cc;text-decoration:none;}.more_menu a:hover {text-
decoration:underline;}.unselected{color:#FFFFFF;}.q_box_top_left {background-image:url('http://sp.ask.com/sh/i/a14/sprites
/shared_sprite_18_ie6.png');display:block;background-position: 0px -948px;width:445px;top:0px;height:4px;float:left;position:relative;}.q_box_top_right
{background-image:url('http://sp.ask.com/sh/i/a14/sprites/shared_sprite_18_ie6.png');display:block;background-position: -140px
-389px;width:7px;top:0px;height:4px;position:absolute;right:-7px;}.q_box_bot_left{background-image:url('http://sp.ask.com/sh/i/a14/sprites
/shared_sprite_18_ie6.png');display:block;background-position: 0px
-958px;width:445px;top:0px;height:4px;float:left;position:relative;clear:both;}.q_box_bot_right {background-image:url('http://sp.ask.com/sh/i/a14/sprites
/shared_sprite_18_ie6.png');display:block;background-position: -140px -399px;width:7px;top:0px;height:4px;position:absolute;right:-7px;}#mini_me
{float:right;height:76px;width:320px;position:relative;margin-top:16px;}#mini_me a,#mini_me a:visited {color:#0055cc;}#profileProgress a,#profileProgress
a:visited {color:#0055cc;}#mini_me_right {position:absolute;height:76px;width:10px;display:block;background-image:url(http://sp.ask.com/sh/i/a14/sprites
/shared_sprite_18_ie6.png);background-position:-501px -290px;right:0px;}#mini_me_left
{position:absolute;height:76px;width:310px;display:block;background-image:url(http://sp.ask.com/sh/i/a14/sprites/shared_sprite_18_ie6.png);background-
position:-100px -290px;left:0px;}#mini_me_avatar {border:solid 2px #fff;vertical-align:top;float:left;width:48px;height:48px;}#mini_me_notification
```

**Figure 60. Image tailoring for Internet Explorer 6**

Analyzing *www.ask.com* identified browser-based tailoring for both versions of Internet Explorer. Figure 60 shows that every image was modified specifically for Internet Explorer 6. This is likely due to rendering or transparency issues with that particular browser.

Clients using Internet Explorer 8 also receive a tailored response. Figure 61, below, shows several formatting modifications that were specific to that particular response.

```
+ margin-top:29px;margin-bottom:14px;
+ line-height:1.6;
- relativ
+ absolut
- block;white-space:nowrap;float:left
+ inline
- 2
+ height:270px;
+ width:302px;
+ padding-top:0px;margin-top:0px;
- 8
+ 11
- body {overflow-y:scroll;}
```

**Figure 61. Tailored formatting for Internet Explorer 8**

### *www.match.com / www.expedia.com*

**http://www.match.com/**

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=utf-8 | 52c60953bf93de3ebb7b62729848b17d | 34793 | view | text | cleaned | stripped | | |
| Browser | Mozilla/4.0 (compati... | text/html; charset=utf-8 | b1a2e6b43691e3727ca3743dda552556 | 34797 | view | text | cleaned | stripped | diff | inline diff |
| Browser | Mozilla/4.0 (compati... | text/html; charset=utf-8 | c67fe71bf0e95a4f6ce3cc15e8334d81 | 34796 | view | text | cleaned | stripped | diff | inline diff |
| Browser | Mozilla/5.0 (Macinto... | text/html; charset=utf-8 | 323e99e065a0cb52121650d65c777f8d | 34792 | view | text | cleaned | stripped | diff | inline diff |
| Browser | Mozilla/5.0 (Windows... | text/html; charset=utf-8 | 6769beff76d582dfeddfb16a344f4ed4 | 34793 | view | text | cleaned | stripped | diff | inline diff |
| Duplicate | duplicate | text/html; charset=utf-8 | 1867baf1724fe647107a7387dcbdfd20 | 34792 | view | text | cleaned | stripped | diff | inline diff |
| Duplicate | duplicate | text/html; charset=utf-8 | fb9b1710b280f678005d6510e950a185 | 34793 | view | text | cleaned | stripped | diff | inline diff |
| Duplicate | duplicate | text/html; charset=utf-8 | 43bd26df646bf22cdf2e19c8fe34a6fa | 34793 | view | text | cleaned | stripped | diff | inline diff |
| Operating System | Linux | text/html; charset=utf-8 | b0106e0afded4636c96179d1303aabff | 34793 | view | text | cleaned | stripped | diff | inline diff |
| Operating System | OpenBSD | text/html; charset=utf-8 | 70ba0812b2fa592772368f3c8a609999 | 34793 | view | text | cleaned | stripped | diff | inline diff |

```
<body id="ctl00_pageBody" class="firefox w3c ie6 traditionalBoxModel">
```

**Figure 62. HTML element with a tailored class based on the browser fingerprint**

Above, figure 62 shows how the web page is providing a tailored class to elements based on the User-Agent string presented by the client. This is obvious both by the names in the highlighted text as well as the color visualization provided by the tool.

Another website, Expedia, showed a more extreme case of tailoring class elements based on the browser fingerprint.

```
<div id="divHBWSavingsRad" >
<div id="divWr1" class="divWr" onclick="hw('1');"><input id="r1" name="srch" type="radio" value="flthot" class="Wr chkFF" /><label for="r1"
id="l1">Flight + Hotel</label></div>
<div id="divWr7" class="divWr" onclick="hw('7');"><input id="r7" name="srch" type="radio" value="fltcar" class="Wr chkFF" /><label for="r7"
id="l7">Flight + Car</label></div>
<div id="divWr5" class="divWr" onclick="hw('5');"><input id="r5" name="srch" type="radio" value="flthotcar" class="Wr chkFF" /><label for="r5"
id="l5">Flight + Hotel + Car</label></div>
<div id="divWr6" class="divWr" onclick="hw('6');"><input id="r6" name="srch" type="radio" value="hotcar" class="Wr chkFF" /><label for="r6"
id="l6">Hotel + Car</label></div>
</div>

<div class="dsText" style="background:transparent url(http://media.expedia.com/media/content/expus/graphics/home/wiz/wizard_booking_image.gif)
no-repeat;text-align:right;margin-top:0px;width:196px;height:107px;margin-right:26px;">
<a rel="nofollow" class="WizOfferLink" style="color:#ffffff;font-size:10px;margin-top:92px;margin-right:66px;background:transparent
url(http://media.expedia.com/media/content/expus/graphics/home/wiz/i_icon_10x10.gif) no-repeat right;padding-right:14px;" href="/" onclick="doDS_SA('-
57571'); expDS.showDS(this, event);return false;"><span>Learn more</span></a>IE" checked="checked" /><label for="r1" id="l1">Flight</label></div>
<div id="divWr2" class="divWr" onclick="hw('2');"><input id="r2" name="srch" type="radio" value="hot" class="Wr chkIE" /><label for="r2"
id="l2">Hotel</label></div>
<div id="divWr3" class="divWr" onclick="hw('3');"><input id="r3" name="srch" type="radio" value="car" class="Wr chkIE" /><label for="r3"
id="l3">Car</label></div>
<div id="divWr8" class="divWr" onclick="hw('8');"><input id="r8" name="srch" type="radio" value="cru" class="Wr chkIE" /><label for="r8"
id="l8">Cruise</label></div>
<div id="divWr9" class="divWr" onclick="hw('9');"><input id="r9" name="srch" type="radio" value="act" class="Wr chkIE" /><label for="r9"
id="l9">Activities</label></div>
</div>
```

**Figure 63. HTML elements with a tailored class attribute for different browsers**

Figure 63 shows the *div* elements in each response receiving a tailored class attribute that based on the browser seen in the User-Agent string. In this particular case, the "WR-chkFF" class was changed to "Wr-chkIE" on all of the elements due to a change in the client fingerprint from a Firefox browser to an Internet Explorer browser.

*www.opendns.com*

**http://www.opendns.com/**

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=UTF-8 | c9c2d3f68fffd9d65605f6919bd6525d | 11919 | view | text | cleaned | stripped | | |
| Browser | Mozilla/4.0 (compati... | text/html; charset=UTF-8 | c9c2d3f68fffd9d65605f6919bd6525d | 11919 | view | text | cleaned | stripped | | |
| Browser | Mozilla/4.0 (compati... | text/html; charset=UTF-8 | c9c2d3f68fffd9d65605f6919bd6525d | 11919 | view | text | cleaned | stripped | | |
| Browser | Mozilla/5.0 (Macinto... | text/html; charset=UTF-8 | cb53f95a2c47e6675c9679c9283c3d91 | 12014 | view | text | cleaned | stripped | diff | inline diff |
| Browser | Mozilla/5.0 (Windows... | text/html; charset=UTF-8 | c9c2d3f68fffd9d65605f6919bd6525d | 11919 | view | text | cleaned | stripped | | |
| Duplicate | duplicate | text/html; charset=UTF-8 | c9c2d3f68fffd9d65605f6919bd6525d | 11919 | view | text | cleaned | stripped | | |
| Duplicate | duplicate | text/html; charset=UTF-8 | c9c2d3f68fffd9d65605f6919bd6525d | 11919 | view | text | cleaned | stripped | | |
| Duplicate | duplicate | text/html; charset=UTF-8 | c9c2d3f68fffd9d65605f6919bd6525d | 11919 | view | text | cleaned | stripped | | |
| Operating System | Linux | text/html; charset=UTF-8 | c9c2d3f68fffd9d65605f6919bd6525d | 11919 | view | text | cleaned | stripped | | |
| Operating System | OpenBSD | text/html; charset=UTF-8 | c9c2d3f68fffd9d65605f6919bd6525d | 11919 | view | text | cleaned | stripped | | |

```
<link rel="stylesheet" type="text/css" href="http://www-files.opendns.com/css/style.v119.css" />
<link rel="stylesheet" type="text/css" href="http://www-files.opendns.com/css/header.v113.css" />
<!--[if IE]>
<link rel="stylesheet" type="text/css" href="http://www-files.opendns.com/css/header_ie.v104.css" />
<link rel="stylesheet" type="text/css" href="http://www-files.opendns.com/css/style_ie.v102.css" />
<![endif]-->
<!--[if IE 6]>
<link rel="stylesheet" type="text/css" href="http://www-files.opendns.com/css/style_ie6.v104.css" />
<link rel="stylesheet" type="text/css" href="http://www-files.opendns.com/css/header_ie6.v101.css" />
<![endif]-->
<link rel="stylesheet" href="http://www-files.opendns.com/css/safari.css" type="text/css" />
</head>
```

**Figure 64. Tailored formatting for Safari**

Unlike most of the examples containing tailored formatting, OpenDNS only identified an anomaly for the Safari browser. Despite all of the responses containing multiple CSS resources by default, responses to the Safari browser received an additional resource, *safari.css*. Strangely, all the responses include CSS files that seem tailored for other browsers. For example, every response contained the *style_ie6v104.css* resource.

*www.costco.com / www.forums.thestranger.com*

http://www.costco.com/

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=utf-8 | 5539dd9626f065c36c9c7bba5715940b | 110195 | view | text | cleaned | stripped | | |
| Browser | Mozilla/4.0 (compati... | text/html; charset=utf-8 | aa6d4bc5bfee63b2e4d42706194cc24a | 108611 | view | text | cleaned | stripped | diff | inline_diff |
| Browser | Mozilla/4.0 (compati... | text/html; charset=utf-8 | 7e0f3cc3ed0abd319cac4758cc15a169 | 108611 | view | text | cleaned | stripped | diff | inline_diff |
| Browser | Mozilla/5.0 (Macinto... | text/html; charset=utf-8 | 8711816058be410d68d510a161d287b4 | 110195 | view | text | cleaned | stripped | diff | inline_diff |
| Browser | Mozilla/5.0 (Windows... | text/html; charset=utf-8 | d3d9c2710b0a0cf7dccffcfcb098cef8 | 110195 | view | text | cleaned | stripped | diff | inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8 | 7c780d8980075f797e815e2090973d8c | 110195 | view | text | cleaned | stripped | diff | inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8 | e85dae0185e8928ef7792cc00445bc01 | 110195 | view | text | cleaned | stripped | diff | inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8 | 97d0daa7753c5a7a9328f4fb835ecfe3 | 110195 | view | text | cleaned | stripped | diff | inline_diff |
| Operating System | Linux | text/html; charset=utf-8 | afa818de13c783a01864500886d72541 | 110195 | view | text | cleaned | stripped | diff | inline_diff |
| Operating System | OpenBSD | text/html; charset=utf-8 | 859b3557f7807d7df5e3c949210b41db | 110195 | view | text | cleaned | stripped | diff | inline_diff |


**Figure 65. Tailored formatting for Internet Explorer 8**

Analyzing www.costco.com showed tailoring specifically for the Internet Explorer browser. In this case, the "topnav" was removed as a URL attribute on anchor elements. This is likely a flag used by the server to deliver a tailored response for that particular resource.

This type of tailored formatting was seen during the analysis of many other websites. Figure 66, below, shows a "diff" identifying the insertion of *floatleft* attributes.

http://forums.thestranger.com/

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=ISO-8859-1 | 6bb5dbdf615e9e655480c779c6fdeaff | 40299 | view | text | cleaned | stripped | | |
| Browser | Mozilla/4.0 (compati... | text/html; charset=ISO-8859-1 | 8223d9c611482adb81264787811e1462 | 40443 | view | text | cleaned | stripped | diff | inline_diff |
| Browser | Mozilla/4.0 (compati... | text/html; charset=ISO-8859-1 | 8223d9c611482adb81264787811e1462 | 40443 | view | text | cleaned | stripped | diff | inline_diff |
| Browser | Mozilla/5.0 (Macinto... | text/html; charset=ISO-8859-1 | 6bb5dbdf615e9e655480c779c6fdeaff | 40299 | view | text | cleaned | stripped | | |
| Browser | Mozilla/5.0 (Windows... | text/html; charset=ISO-8859-1 | 6bb5dbdf615e9e655480c779c6fdeaff | 40299 | view | text | cleaned | stripped | | |
| Duplicate | duplicate | text/html; charset=ISO-8859-1 | 6bb5dbdf615e9e655480c779c6fdeaff | 40299 | view | text | cleaned | stripped | | |
| Duplicate | duplicate | text/html; charset=ISO-8859-1 | 6bb5dbdf615e9e655480c779c6fdeaff | 40299 | view | text | cleaned | stripped | | |
| Duplicate | duplicate | text/html; charset=ISO-8859-1 | 6bb5dbdf615e9e655480c779c6fdeaff | 40299 | view | text | cleaned | stripped | | |
| Operating System | Linux | text/html; charset=ISO-8859-1 | 6bb5dbdf615e9e655480c779c6fdeaff | 40299 | view | text | cleaned | stripped | | |
| Operating System | OpenBSD | text/html; charset=ISO-8859-1 | 6bb5dbdf615e9e655480c779c6fdeaff | 40299 | view | text | cleaned | stripped | | |


**Figure 66. Browser-based format tailoring**

Both *www.costco.com* and *forums.thestranger.com* required tailored formatting to handle inconsistencies between the various browsers. The largely static resources allowed the multiplexing proxy to easily identify the server–side behavior for these websites.

**Element Injection**

*news.google.com*

http://news.google.com/

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=UTF-8 | 2d4f9b39a29304945c1263bbea5085cb | 339663 | view | text | cleaned | stripped | | | |
| Browser | Mozilla/4.0 (compati... | text/html; charset=UTF-8 | a288f7c91d6c639956a34c371ba8c7cf | 314665 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff |
| Browser | Mozilla/4.0 (compati... | text/html; charset=UTF-8 | e3c9a8bdbe1205d339a5c75a526f3ce8 | 326603 | view | text | cleaned | stripped | diff | inline_diff | |
| Browser | Mozilla/5.0 (Macinto... | text/html; charset=UTF-8 | 1c1f6c6ab2297c455e0161f1ea5bbcca | 339587 | view | text | cleaned | stripped | diff | inline_diff | |
| Browser | Mozilla/5.0 (Windows... | text/html; charset=UTF-8 | 6f6beca44ccef2580ccdd217822cecee | 339663 | view | text | cleaned | stripped | diff | inline_diff | |
| Duplicate | duplicate | text/html; charset=UTF-8 | 94f32f91bd651c8470534785829b1f40 | 339663 | view | text | cleaned | stripped | diff | inline_diff | |
| Duplicate | duplicate | text/html; charset=UTF-8 | 15e0c9f3ed19bdb42588291400acf1a8 | 339663 | view | text | cleaned | stripped | diff | inline_diff | |
| Duplicate | duplicate | text/html; charset=UTF-8 | 407910a3387a057d152fb49491c107e | 339663 | view | text | cleaned | stripped | diff | inline_diff | |
| Operating System | Linux | text/html; charset=UTF-8 | be18b87ba274b2c9ad51565a8aad04a0 | 339663 | view | text | cleaned | stripped | diff | inline_diff | |
| Operating System | OpenBSD | text/html; charset=UTF-8 | 2239bbde8cbf37663442725dae0bac65 | 339663 | view | text | cleaned | stripped | diff | inline_diff | |
| Operating System | Windows | text/html; charset=UTF-8 | fa3bc04cf6999920fb72c603eb56c46d | 339663 | view | text | cleaned | stripped | diff | inline_diff | |

```
<td class="video-spacer"></td></tr></table></div></div></div>
</div></div></div><div class="gadget-wrapper gsid-FFH"><div id=":FAST_FLIP_HOMEPAGE" class="gadget t-FAST_FLIP_HOMEPAGE gsid-FFH
yesscript"><div id=":FAST_FLIP_HOMEPAGE_h" class="basic-title"><h2 class="text"><span class="title">Google Fast Flip</span></h2></div>
<div style="height:450px;width:300px"><objectiframe class="gadget-iframe-contents" height="450" style="height:450px;width="300"
type="text/html:300px" frameborder="0" id="ff-hp" datasrc="about:blank"></object></diviframe>
<input id="ff-hp-src" type="hidden" value="http://fastflip.googlelabs.com/embed?source=news&amp;browse=2&amp;fullw=300"></div></div><div
```

**Figure 67. I-frame injected into specific browsers**

Figure 67, above, shows the analysis of news.google.com. Not only was there clearly browser-based tailoring in the responses, but one of the modifications was changing a <*div*> element into an <*iframe*>. Typically, an injected i-frame is a telling sign of malicious behavior. It is often considered to be one of the most popular methods used for infecting legitimate websites (Patil). In this case, however, it was used to handle formatting differences between browsers.

The rest of the differences detected on the website were not as informative or useful. While some of them appeared to be dynamic URLs, several were large differences that the "diff" algorithm could not analyze in a meaningful way. Despite this, the multiplexing proxy was able to identify the i-frame that was injected into a subset of the responses. This is a behavior that would have been very difficult to detect or know about without the aid of a tool that can efficiently and effectively compare responses from varying client fingerprints.

*www.huffingtonpost.com*

http://www.huffingtonpost.com/

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=utf-8 | e8a3de73aec960013eb92d8f941874e6 | 269802 | view | text | cleaned | stripped | | | | |
| Browser | Mozilla/4.0 (compati... | text/html; charset=utf-8 | 2caca1e4a7f9983dfb61496c798c2812 | 270558 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/4.0 (compati... | text/html; charset=utf-8 | 8cb8128a7588157c6507b7a2ac9f8723 | 269994 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/5.0 (Macinto... | text/html; charset=utf-8 | 240945406e1308283bb136e18662db7f | 269876 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/5.0 (Windows... | text/html; charset=utf-8 | e226f2521edc990851fb3e251e5b18b4 | 269875 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/5.0 (Windows... | text/html; charset=utf-8 | a6499451a3bb925ba188c7b565f26929 | 269855 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8 | 36fb457e7bc37be070ccbc6d394ef14 | 269916 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8 | f31c7e5e8fd392a6d8e2a04e85ad3c6c | 269887 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8 | d46e6034890a4c223a144380b6880ed9 | 269879 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Operating System | Linux | text/html; charset=utf-8 | a678d7f6ece55fd5b7033a63d4ddddac | 269846 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Operating System | OpenBSD | text/html; charset=utf-8 | 332cf4c7d827bff6eb459bfa977481b8 | 269908 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Operating System | Windows | text/html; charset=utf-8 | 565c39818f8efc78f795c0ea1683841a | 269835 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |

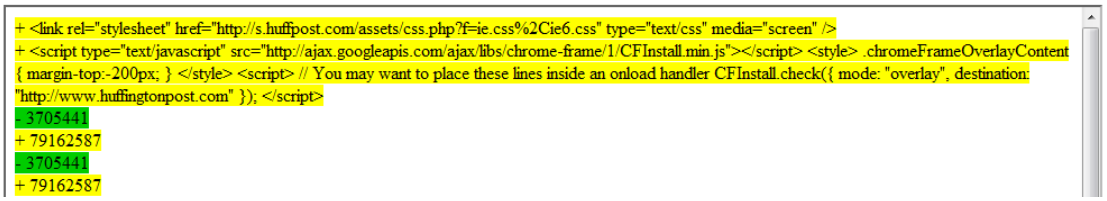



**Figure 68. Injected JavaScript and stylesheet**

Despite a heavy amount of instantly dynamic content, the analysis of *www.huffingtonpost.com* detected a length anomaly in the response for the Internet Explorer 6 browser. Figure 68 shows how both a script and stylesheet were added to the response for that particular request. While all of the responses had varying lengths, the differences between most of the responses was minimal when compared to the response with the injected elements. References to IE6 inside the injected element confirmed this was not simply instantly dynamic content.

In this case, the ability to detect a length anomaly allowed the multiplexing proxy to easily identify a response that had tailored web content despite every response having unique, instantly dynamic data. As more analytics are added to the multiplexing proxy, this type of automated detection should improve even more.

**Instantly Dynamic Behavior**

***www.target.com***

**http://www.target.com/**

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html;charset=UTF-8 | 30b0d2141fd92cca1e5760d43410cf90 | 85810 | view | text | cleaned | stripped | | | |
| Browser | Mozilla/4.0 (compati... | text/html;charset=UTF-8 | 2a17cc2433411f65b84e0a3aa73b6826 | 85810 | view | text | cleaned | stripped | diff | inline_diff | |
| Browser | Mozilla/4.0 (compati... | text/html;charset=UTF-8 | 35074385cca9c1d3354e2756a5798d61 | 87862 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/5.0 (Macinto... | text/html;charset=UTF-8 | 5bde7ec019bbc60ce91d39ece7f27c07 | 85810 | view | text | cleaned | stripped | diff | inline_diff | |
| Browser | Mozilla/5.0 (Windows... | text/html;charset=UTF-8 | 5ba6e668d51e7214ad6aa29c46e88876 | 87862 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html;charset=UTF-8 | 82458e289894835373424d3f1cce9797 | 85810 | view | text | cleaned | stripped | diff | inline_diff | |
| Duplicate | duplicate | text/html;charset=UTF-8 | 27210547a9ed2f699e319dff7115a108 | 85810 | view | text | cleaned | stripped | diff | inline_diff | |
| Duplicate | duplicate | text/html;charset=UTF-8 | 4c203fed26bbd60db44f360717e792d0 | 87862 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Operating System | Linux | text/html;charset=UTF-8 | 204867e85c793db398376111fbff61e0 | 85810 | view | text | cleaned | stripped | diff | inline_diff | |
| Operating System | OpenBSD | text/html;charset=UTF-8 | 585eb53595433580e4eb8cebb44d9db2 | 85810 | view | text | cleaned | stripped | diff | inline_diff | |

```
<script type="text/javascript">
<!--
var t0_date=new Date();
window.ue_t0 = t0_date.getTime();
window.ue = new Object;
ue.id='0HKJ2GC80MX5VZNKTJGF';
ue.url='/mn/uedata/181-9012674-3383728';
//-->
</script>

<script type='text/javascript' src='http://z-ecx.images-amazon.com/images/G/01/media/admin/01/performance/uedata10_core_m._V220300090_.js'>
</script>
<script type='text/javascript'>
<!--
if(window.ue&&ue.t) { uei(); ue.t0 = window.ue_t0; uet('ue'); }
//-->\n
</script>
```

**Figure 69. Randomly injected Performance Analysis Script**

Figure 69 shows the analysis of *www.target.com*. While it initially appeared as though it was inconsistently classifying similar responses as both "stripped matching" and "not matching", analysis of the differences showed that Target was injecting performance analysis scripts into only a subset of the responses!

```
<script type='text/javascript' src='http://z-ecx.images-amazon.com/images/G/01/media/admin/01/performance/uedata10_extra_m._V220300084_.js'>
</script>

<script type='text/javascript' src='http://z-ecx.images-amazon.com/images/G/01/wma/clog/core2._V241266068_.js'></script>

<script type="text/html">
<!--
function ueDebug(A) {}
function ueSession() { return "181-9012674-3383728"; }
function ueRequest() { return "0HKJ2GC80MX5VZNKTJGF"; }
function ueMkplID() { return "A3UN6WX5RRO2AG"; }
function ueBaseURL() { return "http://client-log.amazon.com/clog"; }
function adaptCLOG(A) {
A.addDebugCallback( ueDebug );
A.setSessionIDCallback( ueSession );
A.setRequestIDCallback( ueRequest );
A.setMarketplaceIDCallback( ueMkplID );
A.setBaseURLCallback( ueBaseURL );
}
//-->
</script>

<script type='text/javascript'>
<!--
if( window.CLOG && window.adaptCLOG && !clientLogger ) { var clientLogger = new CLOG(); }
//-->
</script>

<div id='be'><form name='ue_backdetect'><input name='ue_back' value='1' style='visibility:hidden'></form>
<script type='text/javascript'><!--
```

**Figure 70. Randomly injected Performance Analysis Script**

Because this was also seen in a response to one of the duplicate requests, it is apparent that this is not tailored toward specific client fingerprints. As figures 69 and 70 show, the multiplexing proxy was correctly identifying additional instantly dynamic content in several of the responses, and proceeded to correctly classify each response.

*www.wired.com*

http://www.wired.com/

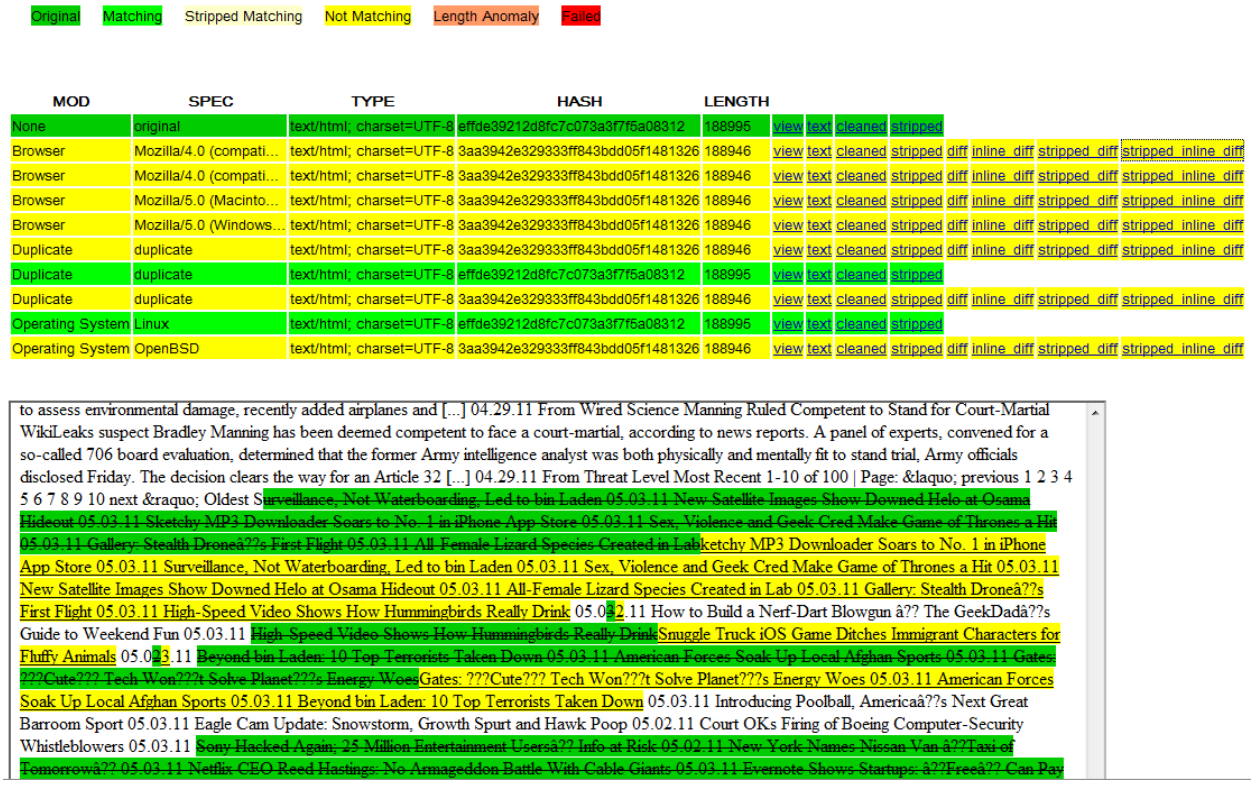



**Figure 71. Instantly dynamic article content**

There was not a significant amount of interesting information gleamed from the multiplexing proxy's analysis of www.wired.com other than the instantly dynamic article content seen in figure 71, above. Despite this, the website was worth including because it provides a great example of the type of tailored content that is not detectable with the multiplexing technique. Figure 72, below, shows how the navigator object may be used in a conditional statement, possibly altering the execution path of the script.

```
/************** DO NOT ALTER ANYTHING BELOW THIS LINE ! ***************/
var s_code=s.t();if(s_code)document.write(s_code)//--></script>
<script language="JavaScript" type="text/javascript"><!--
if(navigator.appVersion.indexOf('MSIE')>=0)document.write(unescape('%3C')+'\!-'+'-')
//--></script><noscript><a href="http://www.omniture.com" title="Web Analytics"><img
src="http://condenast.112.2o7.net/b/ss/condenet-dev/1/H.15.1--NS/0"
height="1" width="1" border="0" alt="" /></a></noscript><!--/DO NOT REMOVE/-->
<!-- End SiteCatalyst code version: H.15.1. -->
```

**Figure 72. Using the navigator object for browser-based tailoring**

In this example, the navigator object was used to test for the presence of "MSIE" in the User-Agent header, a string commonly found in all versions of the Internet Explorer browser. This is done in the context of SiteCatalyst code, a tool that provides website analytics. Because this code was delivered to all client fingerprints regardless of the actual browser used, the technique used by the multiplexing proxy was unable to flag the resource as containing tailored content.

*news.yahoo.com*

http://news.yahoo.com/


**Figure 73. Dynamic URLs for news.yahoo.com**

At first glance, *news.yahoo.com* appeared to simply contain a large amount of dynamic URLs. However, digging a little further into the differences showed instantly dynamic inline ads.

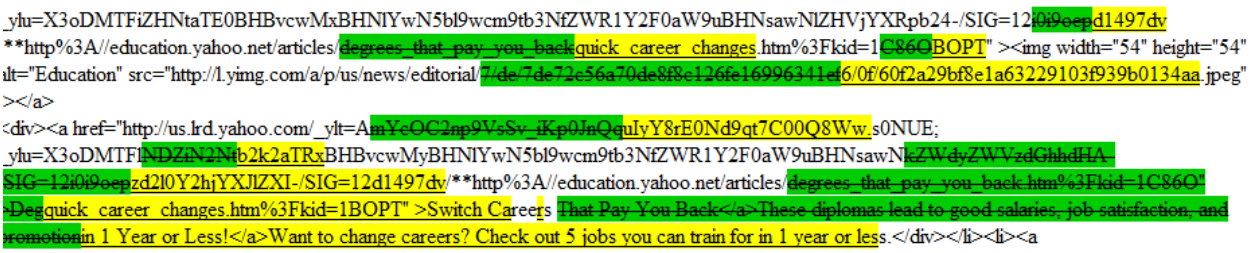



**Figure 74. Instantly dynamic inline ads**

Typically, ads are delivered from a third party domain. In this case, however, the ads are instantly dynamic, yet are still delivered inline with the response.

The "stripped diff" tool was successfully able to cut through the large amount of dynamic URLs, allowing other interesting differences to be analyzed more clearly. This strategy identified browser-based tailoring that was not detected by the multiplexing proxy due to the high volume of instantly dynamic content.

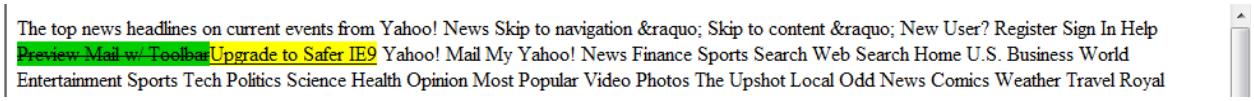



**Figure 75. Browser-based tailoring for toolbar support**

Both Internet Explorer 6 and the earlier version of Firefox received a response asking the user to "upgrade to a Safer IE9". However, the responses to every other client fingerprint provided the option to download a toolbar to preview mail. It is likely that when the request comes from a browser compatible with the toolbar, then the user is given with the option to preview e-mail with the toolbar. However, if the client's browser would not support the toolbar, the user is prodded into upgrading.

The above website, with both tailored and instantly dynamic content, is a good example of the type of response that the multiplexing proxy cannot easily classify well. Unless there is a trigger such as an anomalous length, the multiplexing proxy will not have the ability to detect the tailored content without using improved analytics and deeper content inspection.

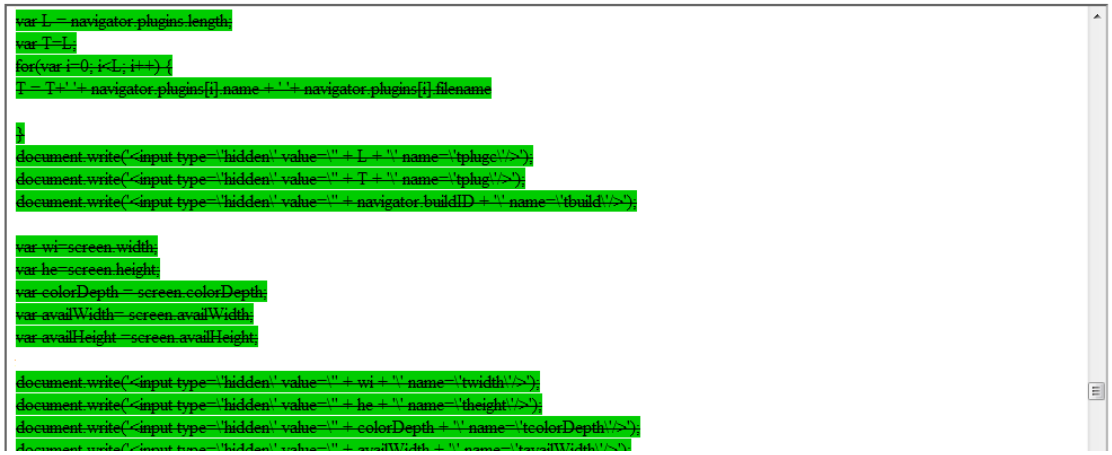
**Other**

***www.plentyoffish.com***

**http://www.plentyoffish.com/**

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=utf-8 | db1a345c63bdf5e4fea67ef555965495 | 49051 | view | text | cleaned | stripped | | | | |
| Browser | Mozilla/4.0 (compati... | text/html; charset=utf-8 | a22d4447ce8e03bd44ff3fb25e5f39e1 | 46837 | view | text | cleaned | stripped | diff | inline_diff | | |
| Browser | Mozilla/4.0 (compati... | text/html; charset=utf-8 | 256eaef6ad403c069fdd94e093dcf301 | 46804 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/5.0 (Macinto... | text/html; charset=utf-8 | 36b2e74792c26300e11e09abcbbe78ad | 48488 | view | text | cleaned | stripped | diff | inline_diff | | |
| Browser | Mozilla/5.0 (Windows... | text/html; charset=utf-8 | d705eda0cbee0c46936b9cbfbc69b2b6 | 49007 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8 | 4303fe13172800510394da3d64e77ac9 | 49011 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8 | a7487b98179f58f03d5e0a99d519c496 | 49055 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8 | fcda2e9077e491f96cc11df9c29c4c02 | 48999 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Operating System | Linux | text/html; charset=utf-8 | 97597951ac7ea93db9dd07196d4a1131 | 48999 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Operating System | OpenBSD | text/html; charset=utf-8 | 9c2004f50b82d5021725d876a5566213 | 49007 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |



```
var L = navigator.plugins.length;
var T=L;
for(var i=0; i<L; i++) {
T = T+''+ navigator.plugins[i].name + ''+ navigator.plugins[i].filename

}
document.write('<input type=\'hidden\' value=\'' + L + '\' name=\'tpluge\'/>');
document.write('<input type=\'hidden\' value=\'' + T + '\' name=\'tplug\'/>');
document.write('<input type=\'hidden\' value=\'' + navigator.buildID + '\' name=\'tbuild\'/>');

var wi=screen.width;
var he=screen.height;
var colorDepth = screen.colorDepth;
var availWidth= screen.availWidth;
var availHeight =screen.availHeight;

document.write('<input type=\'hidden\' value=\'' + wi + '\' name=\'twidth\'/>');
document.write('<input type=\'hidden\' value=\'' + he + '\' name=\'theight\'/>');
document.write('<input type=\'hidden\' value=\'' + colorDepth + '\' name=\'tcolorDepth\'/>');
```

**Figure 76. Collecting client and plugin information using hidden elements**

Figure 76 shows a response from *www.plentyoffish*.com that contains JavaScript collecting client information. In addition to gathering information such as the client's screen resolution, the script iterated through the list of available plugins enabled in the client browser. This information was then included in hidden input elements that could be submitted to the server. However, this information collection was only supported on a subset of the browsers.

*www.gizmodo.com*

http://gizmodo.com/

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | |
|---|---|---|---|---|---|
| None | original | text/html; charset=utf-8; | 730cd11542be9be91d9e41c4f532932f | 53268 | view text cleaned stripped |
| Browser | Mozilla/4.0 (compati... | text/html | a2ff46aa13393af4002caccf252a8f2 | 299 | view text cleaned stripped diff inline_diff stripped_diff |
| Browser | Mozilla/4.0 (compati... | text/html; charset=utf-8; | b6269b2ef624131866a83e4ff4376b54 | 53297 | view text cleaned stripped diff inline_diff |
| Browser | Mozilla/5.0 (Macinto... | text/html; charset=utf-8; | d02c26faee48fa75bc978e671e41e895 | 53303 | view text cleaned stripped diff inline_diff |
| Browser | Mozilla/5.0 (Windows... | text/html; charset=utf-8; | 7c532f3e32d7ad50fefb82d78baa0532 | 53270 | view text cleaned stripped diff inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8; | e867cf423a01ff5964aff21c9f9d9792 | 53268 | view text cleaned stripped diff inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8; | 23702e3ea05731dbdc924e8dbc71ce5d | 53268 | view text cleaned stripped diff inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8; | 286059516ffbf40ddb64003c68a02001 | 53268 | view text cleaned stripped diff inline_diff |
| Operating System | Linux | text/html; charset=utf-8; | e917daec42510353eb2d964824824192 | 53268 | view text cleaned stripped diff inline_diff |
| Operating System | OpenBSD | text/html; charset=utf-8; | e64fbabfeec579a20c436449853e3190 | 53268 | view text cleaned stripped diff inline_diff |

```
<div class="mainContent"></div>
<script type="text/javascript">
var ganjaHost = 'Host: GM6728<br/>Generated on: 2011-05-04 00:07:5808:24<br />INIT: http://gizmodo.com/<br />User agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.12) Gecko/20101026 Firefox/3.6.12Macintosh; U; Intel Mac OS X 10_6_1; en-US) AppleWebKit/530.19.2 (KHTML, like Gecko) Version/4.0.2 Safari/530.19';
```

**Figure 77. Tailoring performed on the server instead of a reverse proxy**

Figure 77 shows a byte-for-byte insertion of the User-Agent string into the response. While course-grained content tailoring based on the client's browser can be performed using a number of strategies, this type of insertion requires the response to be dynamically generated after analyzing the request. When only course-grained tailoring for browsers is necessary, other strategies such as a reverse proxy may be used.

Similar to many of the previous case studies, the response for the Internet Explorer 6 client was dramatically different from the rest, replacing all of the content with just a few lines of code that redirect the user to different URL.

```
</script> <div class="ad_chartbeat_end"><script type="text/javascript"> var _sf_async_config={uid:3012,domain:"gizmodo.com"}; </script></div>
</body> </html>
+ script> var host = location.host.split('.').slice(-2).join('.'); if ( location.pathname.length > 1 ) { location.href = location.protocol + '//m.' + host + '/' +
location.pathname; } else { location.href = location.protocol + '//m.' + host + '/' + location.hash.substr(2); } </script>
```

**Figure 78. Custom redirect for Internet Explorer 6**

While the User-Agent string variable appeared to be benign, discovering the Internet Explorer 6 redirection is exactly the type of analysis the multiplexing proxy excels at. This type of information is critical for reverse engineering, website scraping, and many other related tasks.

***www.apple.com***

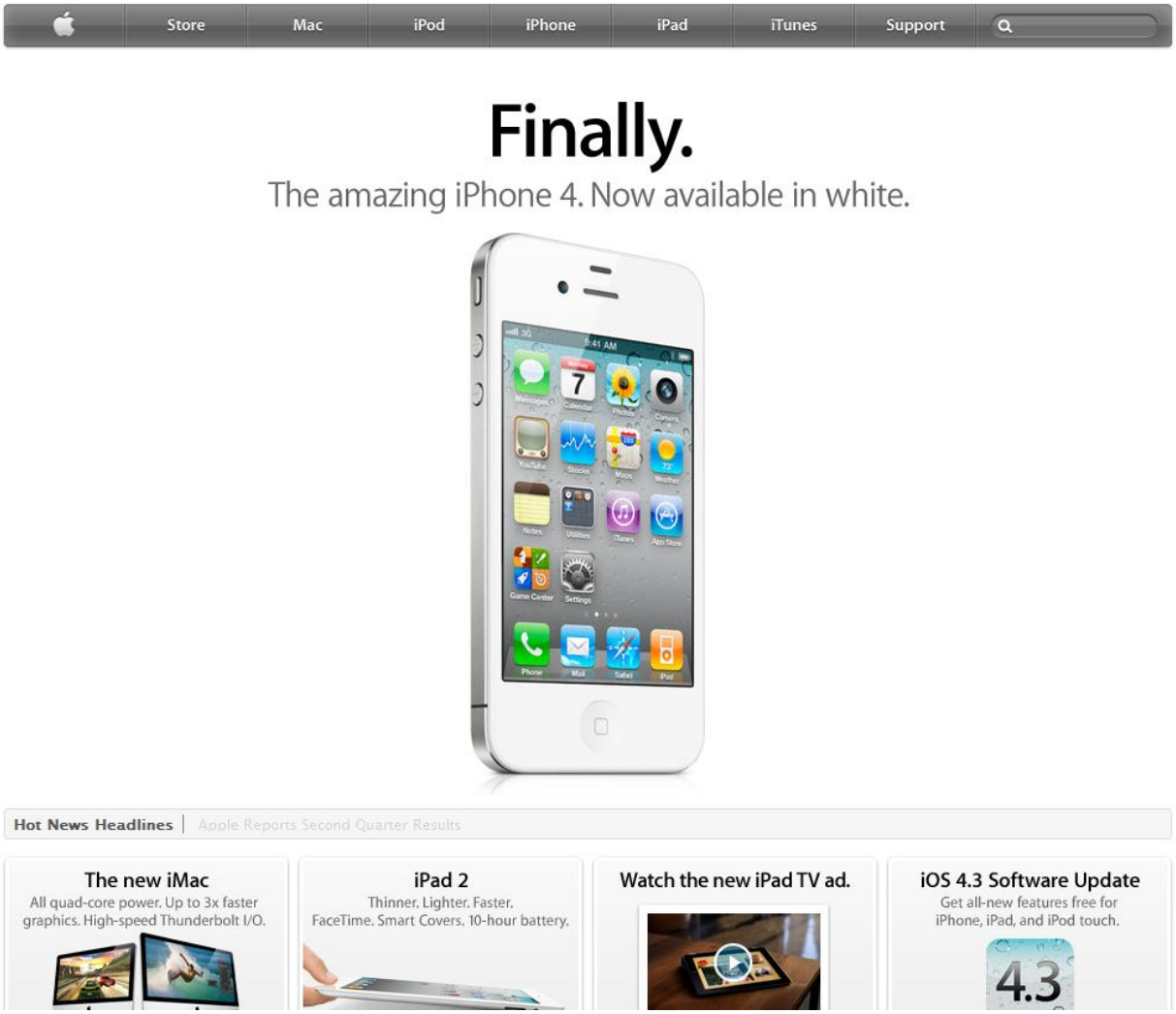



**Figure 79. Screenshot of www.apple.com**

Despite most high traffic websites delivering dynamic responses, there are still many popular sites deliver all static content. Figure 79, above, shows a website that appears to be very robust and dynamic. However, analysis of the website showed that it was composed entirely of static components.

http://images.apple.com/global/elements/arrows/morearro... *Variations: 1*
http://images.apple.com/global/nav/images/globalheader.... *Variations: 1*
http://images.apple.com/global/nav/images/globalnav.png *Variations: 1*
http://images.apple.com/global/nav/images/globalnav_tex... *Variations: 1*
http://images.apple.com/global/nav/images/globalsearch_... *Variations: 1*
http://images.apple.com/global/nav/images/globalsearch_... *Variations: 1*
http://images.apple.com/global/nav/scripts/globalnav.js *Variations: 1*
http://images.apple.com/global/nav/styles/navigation.cs... *Variations: 1*
http://images.apple.com/global/scripts/ac_blackout.js *Variations: 1*
http://images.apple.com/global/scripts/apple_core.js *Variations: 1*
http://images.apple.com/global/scripts/browserdetect.js *Variations: 1*
http://images.apple.com/global/scripts/lib/prototype.js *Variations: 1*
http://images.apple.com/global/scripts/lib/scriptaculou... *Variations: 1*
http://images.apple.com/global/scripts/promomanager.js *Variations: 1*
http://images.apple.com/global/scripts/search_decorator... *Variations: 1*
http://images.apple.com/global/styles/ac_blackout.css *Variations: 1*
http://images.apple.com/global/styles/base.css *Variations: 1*
http://images.apple.com/home/elements/home-bg-choosecou... *Variations: 1*
http://images.apple.com/home/elements/ticker_btm_grey.g... *Variations: 1*
http://images.apple.com/home/elements/ticker_top_grey.g... *Variations: 1*
http://images.apple.com/home/elements/worldwide_us.png *Variations: 1*
http://images.apple.com/home/images/promo_imac_20110426... *Variations: 1*
http://images.apple.com/home/images/promo_ios_4_3_20110... *Variations: 1*
http://images.apple.com/home/images/promo_ipad2_2011042... *Variations: 1*
http://images.apple.com/home/images/promo_ipadad2011050... *Variations: 1*
http://images.apple.com/home/images/white_iphone_hero_2... *Variations: 1*
http://images.apple.com/home/images/white_iphone_title_... *Variations: 1*
http://images.apple.com/home/scripts/promotracker.js *Variations: 1*
http://images.apple.com/home/scripts/ticker.js *Variations: 1*
http://images.apple.com/home/styles/home.css *Variations: 1*
http://images.apple.com/metrics/scripts/s_code_h.js *Variations: 1*
http://metrics.apple.com/b/ss/appleglobal,applehome/1/H... *Variations: 1*
http://www.apple.com/ *Variations: 1*
http://www.apple.com/hotnews/feeds/ticker.rss *Variations: 1*

## http://www.apple.com/

**Figure 80. Analysis of resources from www.apple.com**

A large benefit of having static content is the ability to cache responses for longer periods of time. If there is dynamic content in every response, it is not possible for intermediate devices to cache the response and save bandwidth on later retrievals. However, if all the content is static, it is possible for a device to return a cached response in place of getting the response from the server.

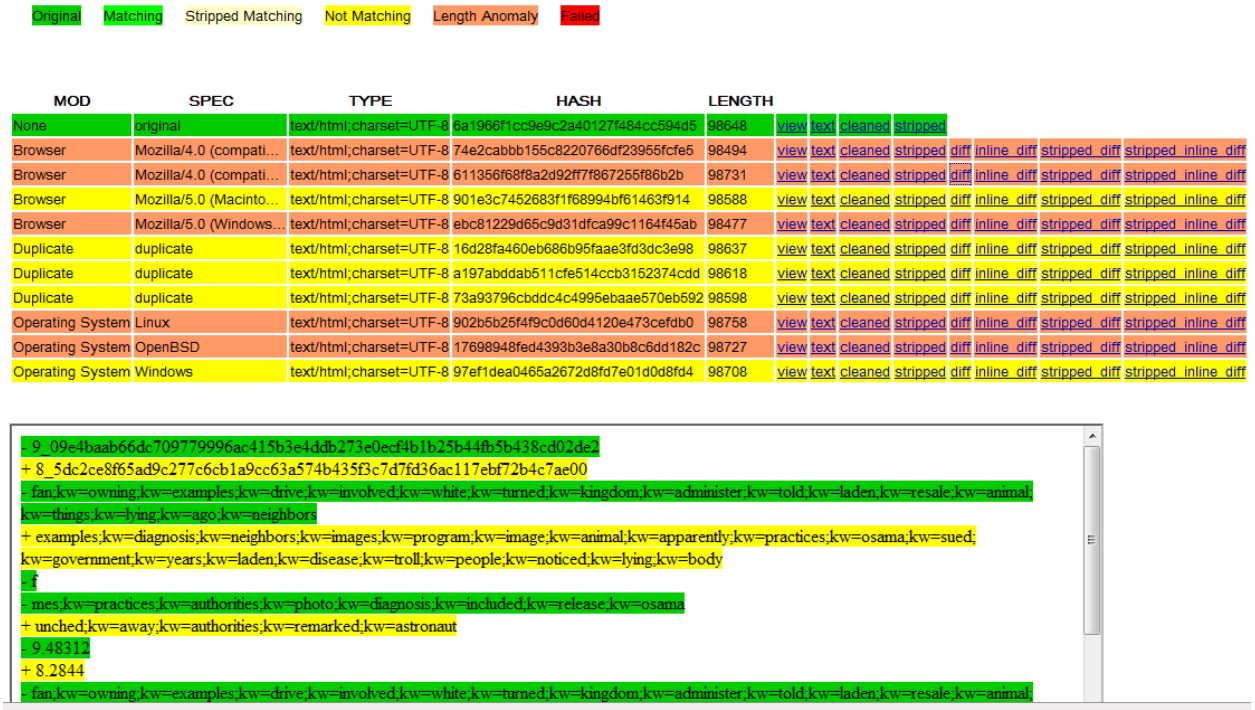
While the multiplexing proxy was unable to find interesting dynamic content on this website, it was able to reveal information about the site being completely static when, by just looking at it, I would have guessed it to be dynamic. This in itself is a powerful capability. Not only is it useful to know when there is instantly dynamic or tailored content, but it is also useful to have an assurance of static content. This allows assumptions to be made when developing tools and services that integrate with the website.

Additionally, if a server delivering static website content was hacked, the multiplexing proxy would more easily detect many of the malicious behaviors previously discussed in the cyber analyst case studies.

**Misclassification**

***www.digg.com / www.ebay.com***

**http://digg.com/**



Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html;charset=UTF-8 | 6a1966f1cc9e9c2a40127f484cc594d5 | 98648 | view | text | cleaned | stripped | | | | |
| Browser | Mozilla/4.0 (compati... | text/html;charset=UTF-8 | 74e2cabbb155c8220766df23955fcfe5 | 98494 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/4.0 (compati... | text/html;charset=UTF-8 | 611356f68f8a2d92ff7f867255f86b2b | 98731 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/5.0 (Macinto... | text/html;charset=UTF-8 | 901e3c7452683f1f68994bf61463f914 | 98588 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/5.0 (Windows... | text/html;charset=UTF-8 | ebc81229d65c9d31dfca99c1164f45ab | 98477 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html;charset=UTF-8 | 16d28fa460eb686b95faae3fd3dc3e98 | 98637 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html;charset=UTF-8 | a197abddab511cfe514ccb3152374cdd | 98618 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html;charset=UTF-8 | 73a93796cbddc4c4995ebaae570eb592 | 98598 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Operating System | Linux | text/html;charset=UTF-8 | 902b5b25f4f9c0d60d4120e473cefdb0 | 98758 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Operating System | OpenBSD | text/html;charset=UTF-8 | 17698948fed4393b3e8a30b8c6dd182c | 98727 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Operating System | Windows | text/html;charset=UTF-8 | 97ef1dea0465a2672d8fd7e01d0d8fd4 | 98708 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |



**Figure 81. Poor results from a very dynamic website**

While analysis by the multiplexing proxy provided interesting results for many websites, there were inevitably some that it was unable to correctly understand and classify. Figure 81 shows how Digg returned a similarly-sized response for each of the duplicate requests, but had more heavily varying lengths for many other responses. This caused a large number of responses to be classified as tailored. However, after analyzing many of the responses against the original, it appeared to be random, instantly dynamic data with no signs of tailored content.

This scenario is similar to the occasional anomaly seen in the empirical classification accuracy data where a website has only a few different response variations, and all the responses to duplicate requests happen to receive the same response. However, in this case study, every response is different, but the lengths of all the duplicate requests happened to be similar to the original response. Therefore, several responses with a more anomalous length were improperly classified as tailored.

Shown below in figure 82, the multiplexing proxy similarly struggled to provide meaningful response analysis for Ebay.

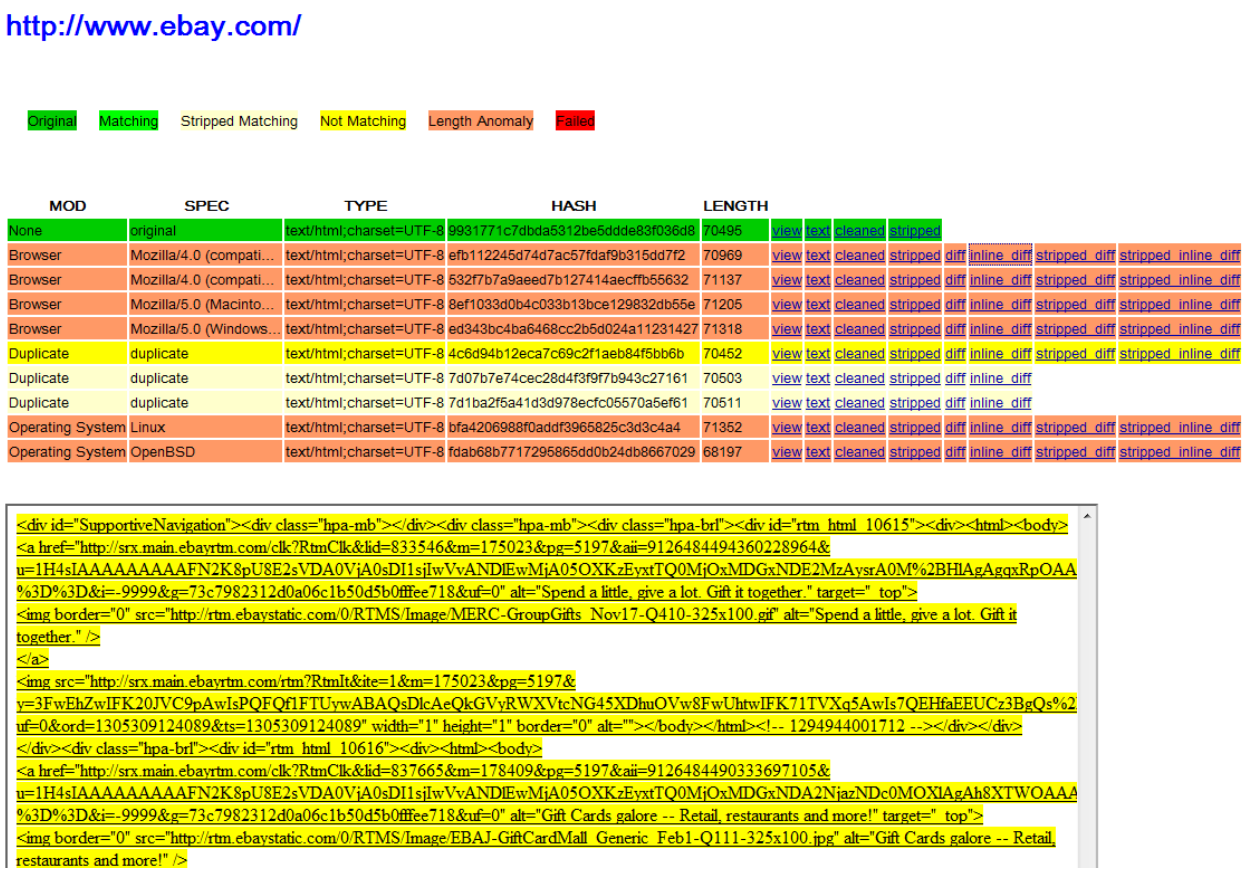



**Figure 82. Misclassification of a dynamic website**

Once again, this can be attributed to an unfortunate case where, by chance, the three responses to duplicate requests returned very similar-length data, causing the instantly dynamic content in other responses to trigger a length-based anomaly for many of the responses. However, further investigation did not show any reason to believe that there was any tailoring involved in any of the responses.

*www.msn.com*

http://www.msn.com/

Original Matching Stripped Matching Not Matching Length Anomaly Failed

| MOD | SPEC | TYPE | HASH | LENGTH | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| None | original | text/html; charset=utf-8 | c306a17495641cb9e43188b605848f6b | 83306 | view | text | cleaned | stripped | | | | |
| Browser | Mozilla/4.0 (compati... | text/html; charset=utf-8 | 2aa05ca8d4dff6a0277c350662dcf66d | 83361 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Browser | Mozilla/4.0 (compati... | text/html; charset=utf-8 | e687fb8b31c3b325c2bf27ca4efa0a82 | 83358 | view | text | cleaned | stripped | diff | inline_diff | | |
| Browser | Mozilla/5.0 (Macinto... | text/html; charset=utf-8 | 561677b399a266ae57311b3cbc0c6543 | 83306 | view | text | cleaned | stripped | diff | inline_diff | | |
| Browser | Mozilla/5.0 (Windows... | text/html; charset=utf-8 | 9c969c5d5f80952a9d17ad752c572ba4 | 83306 | view | text | cleaned | stripped | diff | inline_diff | | |
| Duplicate | duplicate | text/html; charset=utf-8 | a23ed5344672d85642d27621b190c47a | 83361 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Duplicate | duplicate | text/html; charset=utf-8 | b0d35e4c3312b3a65d5e1bfd7e828050 | 83306 | view | text | cleaned | stripped | diff | inline_diff | | |
| Duplicate | duplicate | text/html; charset=utf-8 | be54fe261305fb4c9a294f84c7255fbd | 83306 | view | text | cleaned | stripped | diff | inline_diff | | |
| Operating System | Linux | text/html; charset=utf-8 | d11d253d02d73dd81deb4bb678a039e8 | 83306 | view | text | cleaned | stripped | diff | inline_diff | | |
| Operating System | OpenBSD | text/html; charset=utf-8 | 7bc60a83298f172eeebe487b1f135bb1 | 83361 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |
| Operating System | Windows | text/html; charset=utf-8 | ff4afd4e6ae4e894234932bb2b73a26e | 83361 | view | text | cleaned | stripped | diff | inline_diff | stripped_diff | stripped_inline_diff |

**Figure 83. Results for www.msn.com**

Figure 83 shows responses to duplicate requests classified as both “stripped matching” and “not matching”, resulting in the resource receiving an instantly dynamic classification. However, this is a case where browser-based tailoring was actually missed, and further manual analysis with the multiplexing proxy caught the anomaly!

While looking at the “diff” of “stripped matching” responses simply showed a few request ID modifications for most of the responses, analyzing the response for the Internet Explorer 8 client had an additional line:

*+ /><meta http-equiv=”X-UA-Compatible” content=”IE=7”*

```
+ /><meta http-equiv="X-UA-Compatible" content="IE=7"
- 3932759",requestId:"725e3cca8c464574beb2808b6fe5a473
+ 2432311",requestId:"3831cde5cfdb4e6a89360f06e0e6a8b6
- 725e3cca8c464574beb2808b6fe5a473
+ 3831cde5cfdb4e6a89360f06e0e6a8b6
- 725e3cca8c464574beb2808b6fe5a473
+ 3831cde5cfdb4e6a89360f06e0e6a8b6
```

**Figure 84. Meta element included in the response for Internet Explorer 8**

This additional line tells Internet Explorer 8 to perform rendering under the emulation of Internet Explorer 7. Not being present in the other responses, this would have been very difficult to detect without the multiplexing proxy.

In addition to this anomaly, performing a "stripped diff" on the "not matching" responses provided interesting results.

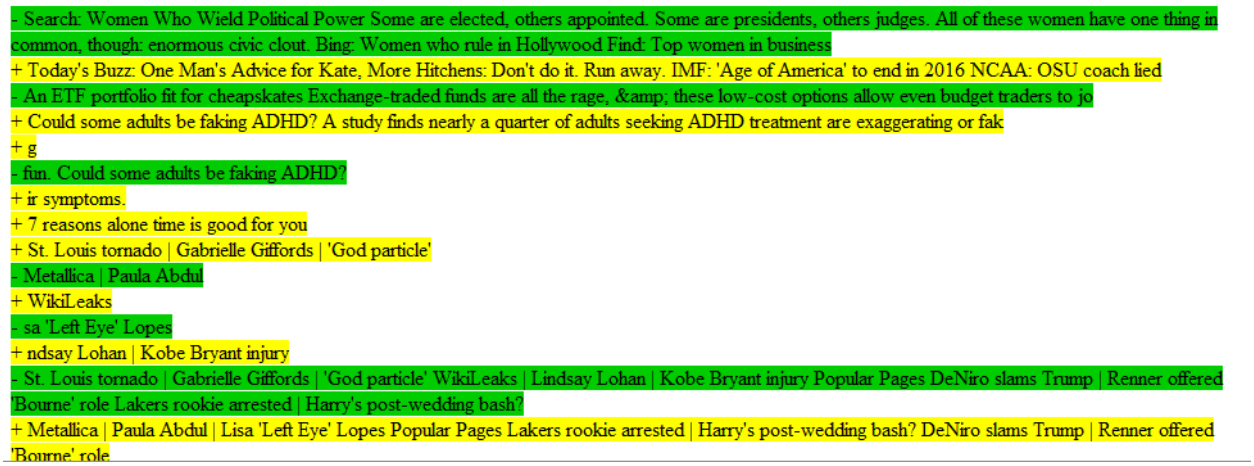



**Figure 85. Stripped diff for www.msn.com**

The dynamically modified data was either a label to or description of a current event or news article. This allows the reverse engineer to see that articles and content are randomly being shuffled, added, and removed between seemingly identical requests.

**Summary**

The multiplexing proxy was very successful in supplying information that would aide a reverse engineer. The response tailoring identified in this study ranged from simple format fixes to complex content and advertisement tailoring. In this case, only browser and operating system client fingerprint modifications were used. However, the information obtained could be augmented with the detection capabilities seen by additionally sending requests with a modified client location. This would provide an even greater amount of information about the behavior of the website under varying client fingerprints.

Not only was the multiplexing proxy able to obtain information about tailored responses, it was also able to identify interesting behavior in the instantly dynamic content. For example, it was possible to detect a randomly injected performance analytics script in responses from www.target.com, and could identify instantly dynamic news content and ads from news.yahoo.com.

By combining the information obtained about both the tailored and instantly dynamic content, a reverse engineer will be significantly further ahead in the analysis of targeted websites. While not all websites were classified correctly, the amount of information that was obtained far outweighed the minimal number of misclassifications.

# CONCLUSION

Both the theoretical and empirical analysis showed that with as little as three duplicate requests, it was possible to provide a sufficient level of classification accuracy for everything but the most critical, automated systems. The discovery and analysis capabilities of the multiplexing proxy far outweighed the small possibility of false positives inhibiting the user, allowing it to successfully detect interesting web content that was tailored toward clients with specific browsers, operating systems, and locations. The ability to detect and classify both instantly dynamic and tailored content provides utility to open source analysts, cyber analysts, and reverse engineers.

The multiplexing system was able to help an open source analyst discover many cases of location-based redirection and censorship. While open source analysts were previously restricted to mitigating tailored web content, this research opens the door to being able to detect when it occurs. The tool proved capable of detecting articles, prices, and other content that was tailored based on geolocation of the client's IP address. Even in the cases where instantly dynamic or tailored content was not present, the tool still provided the analyst with additional trust in the validity of the content. This additional trust is unachievable when receiving a single response based on one particular client location. This research has given sufficient evidence that the tool provides valuable detection capabilities against this type of attack.

This research has the potential to significantly impact how a cyber analysts investigates websites and performs triage of potentially infected machines. The case studies demonstrating tailored exploitation of client fingerprints showed how the multiplexing proxy could provide utility to cyber analysts beyond what is currently available. Not only was the tool able to detect attacks such as injected i-frames and malicious JavaScript in the test environment, but it was also able to detect illegitimate proxies in the wild. The wide array of malicious services on the web capable of delivering tailored content makes this an essential tool moving forward. It has the potential to turn a long, manual process into an efficient and valuable task.

The multiplexing proxy was also very successful in identifying a wide-range of tailored content that would be useful to a reverse engineer. Not only was the tool able to identify simple cases such as format fixes, but it was also able to identify more complex content and advertisement tailoring. Additionally, the tool was shown to have a unique and powerful ability to detect complex instantly dynamic behaviors such as randomly injected performance analytics scripts. In addition to detecting anomalies in the initial resources necessary for rendering a website, it also identified tailored and instantly dynamic content not

present until future AJAX requests were made. The combination of these capabilities provides sufficient evidence of the multiplexing proxy providing utility for a reverse engineer. Further research in this area can greatly impact the ability to thoroughly test, analyze, and characterize websites using a quick, automated approach.

Despite the successes seen in this study, it is apparent that more work will be necessary before a tool such as the multiplexing proxy can be widely usable in a commercial or production setting. Before the proxy is taken out of a research environment, many of the current limitations must be overcome.

Firstly, support for HTTPS will be necessary. This will require the ability of the proxy being able to play man-in-the-middle for SSL/TLS traffic. Unfortunately, this is both technically very difficult, and a privacy issue. The only way for the proxy to gain visibility to SSL/TLS traffic is to hijack the handshake using its own X509 certificate. At this point, the proxy could then open new SSL/TLS connections with the end host, completing secure transactions on behalf of the client.

Secondly, more work will be necessary for detecting tailored content due to HTTP cookies. While the browser, operating system, and location of the client are all modified by the current proxy implementation, a server can still track a user by storing a custom HTTP cookie on the client. While cookies can be deleted and disabled on the client, this may remove vital or interesting functionality from many websites. Instead, it would be interesting to allow cookies to be configurable similar to other client fingerprint attributes.

Although the current implementation worked well for this research, a more robust tool would require the proxy to fingerprint the client operating system and be able to dynamically modify the TCP/IP stack of outgoing requests. For this research, the TCP/IP stack was modified by making the requests through varying operating systems. A better and more widely usable solution would be to create a tool that can dynamically modify the TCP/IP fingerprint as necessary for each request. This would allow the proxy to more accurately handle a wide variety of systems.

Additionally, a more robust collection of remote proxies would be necessary to use this tool in a production environment. In many cases, the remote proxies were not easily collected or reliable. This required frequently searching for and testing new remote proxies while the tool was used for detecting location-based client fingerprint modifications. Because of this, it was not possible to maintain a large

number of remote locations to proxy requests through at any given time. Integration with a wide, secure network of remote proxies would be a necessary step before the proxy became a robust solution.

Lastly, it would be helpful to improve on the current set of analytics. While they proved to be extremely useful for this research, there was significant evidence that they could be even better if they were augmented with contextual information and deeper analysis of the differences. Using the hash, stripped hash, and length metrics allowed the multiplexing proxy to analyze and cache multiple responses very quickly. However, analyzing the content and differences with more complex measurements would provide greater accuracy. Firstly, the number and length of differences between a response and the original could be used as a metric to determine anomalies similar to how the overall response length is currently used. This would be ideal for detecting tailored content within instantly dynamic data. Additionally, a large amount of the tailored content contained common keywords and patterns that could be easily detectable. For instance, “ie” and “ff” were commonly seen in browser-based tailoring. Additionally, analytics such as the length anomaly detection proved to be vital. Extending the anomaly detection to include some of these more in-depth features and capabilities would result in a reduced number of false positives at the cost of additional computation and efficiency.

Despite these necessary improvements, the concept of distributed analytics and assurance has been shown to be a powerful approach that has the potential to greatly impact many areas of research. While this research applied the multiplexing approach to HTTP traffic, it should also be applicable to other protocols and types of information. For example, it would be possible to analyze content distribution networks or routing anomalies through use of a similarly distributed querying system that aggregates results and analyzes them. Future work will not only focus on HTTP, but will also explore many of these related concepts.

# APPENDIX A

## CONFIGURING THE MULTIPLEXING PROXY

The multiplexing proxy allows each user to set a custom configuration that will be saved and used every time a request is received at the multiplexing proxy for that particular user. Figure 86, below, shows the configuration screen that allows the user to decide how each request should be multiplexed with duplicate and varying client fingerprints.

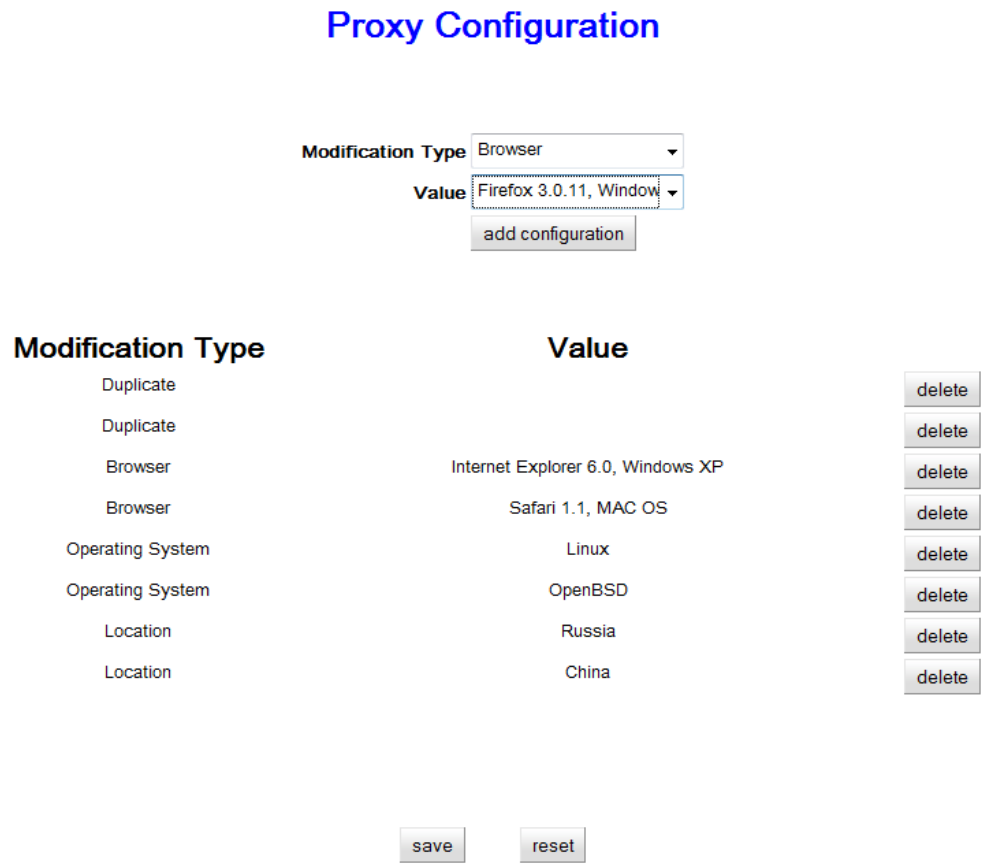



**Figure 86. The multiplexing proxy configuration screen**

The top of the configuration screen provides a selection box that lists the available types of requests that can be added. When a user selects a modification type and value, the additional request may then be added to the list of multiplexed variations. The values for each of the modification types are pulled from a

prepopulated database of interesting values. For instance, the values for the "browser" modification type come from a table of popular user-agent strings for different types of browser. The only exception is "duplicate" requests, which have no value. The ability to modify the number of duplicate requests allows the user to decide an appropriate balance between classification accuracy and speed.

Once the multiplexing proxy is appropriately configured, it can be saved and referenced for all future requests from that particular user. Multiplexed requests may be added or removed at any time by revisiting the configuration screen.

## USING THE BROWSER PLUGIN

Currently, the only plugin that exists is for the Firefox browser. The plugin adds many different visible components that modify the typical browser interface.

Firstly, the multiplexing system plugin adds a custom toolbar to the browser. To the far left, the toolbar provides a new search panel that multiplexes the website entered into the textbox. To the right of the search panel is room that is used to provide overall analysis of the website resources.

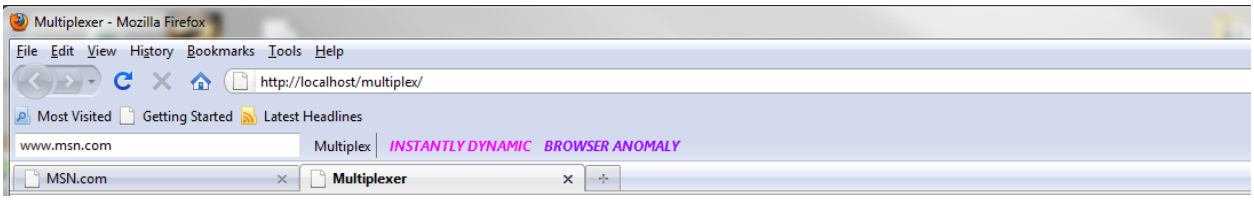



**Figure 87. The multiplexing toolbar**

As responses are received at the proxy, the plugin will query the proxy for a summary of the analysis results that can be displayed to the user. This summary shows whether all the resources were static, or as is seen in figure 87, some of the resources were instantly dynamic or tailored toward a specific client fingerprint. Clicking on the summary results opens a new tab that displays the full analysis for the website.

The right side of the multiplexing proxy's toolbar provides a way for user's to log in and log out of the system. This is a necessary step that allows the multiplexing proxy to know which user's configuration should be used for multiplexing requests.

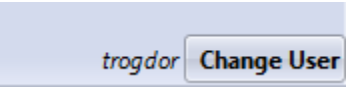



**Figure 88. The multiplexing proxy login**

As figure 88 shows, the currently logged in user is displayed to the left, and a button for logging in with a different user is provided on the right. In this case, we can see that “trogdor” is currently logged in to the multiplexing system.

Lastly, an icon is added to the bottom right corner of the browser interface. This icon is used for quickly toggling the multiplexing proxy on or off. Since it may not be desirable to multiplex certain requests, it is essential to have quick access to a toggle such as this. Otherwise, there may be adverse side effects when dealing with websites that do not have an idempotent behavior.

**Figure 89. Icon showing that requests will be multiplexed**

**Figure 90. Icon showing that requests will not be multiplexed**

Figures 89 and 90 show the buttons that toggle the multiplexing system on and off. By simply clicking the button, the user has full control of whether or not the requests are routed through the multiplexing proxy.

## UNDERSTANDING THE ANALYSIS RESULTS

The multiplexing proxy integrates with a web service that provides enhanced functionality for displaying and analyzing the results of a multiplexed request. The top of the analysis page shows a list of resources from the website sorted by classification and dynamism. This allows the most interesting resources to float to the top of the list.

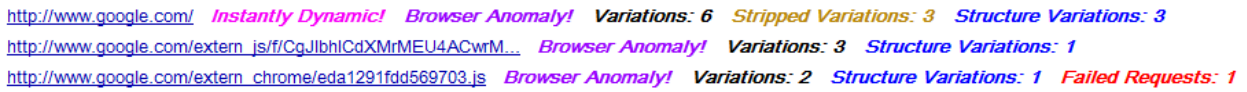



**Figure 91. Example Resource List**

Figure 91 shows dynamic resources listed after visiting www.google.com. Each resource may have multiple classifications listed to the right. For instance, the first resource was flagged as having both instantly dynamic data as well as a browser anomaly. Additionally, the analysis provides a count of the number of variations, stripped variations, and structural variations that were seen for each resource. These counts can be used to better understand the behavior and dynamism of a resource. For example, a resource that had eight response variations is likely much more dynamic and complex than a resource with only 2 response variations. The stripped variations allow the user to understand whether the dynamism is from content visible to the user, or from attributes and scripts working behind the scenes. Structural variations are useful for determining whether or not the actual structure of the website is changing, or whether the changes are likely modifications of small things such as attributes and URLs. Lastly, the analysis provides a count of the number of requests that failed. This is ideal for identifying location-based censorship when using remote proxies. Although all the static resources are initially hidden, the filter can be removed to see a list of every resource requested while visiting the website.

Clicking on one of the resources provides even greater analysis and visualization options.

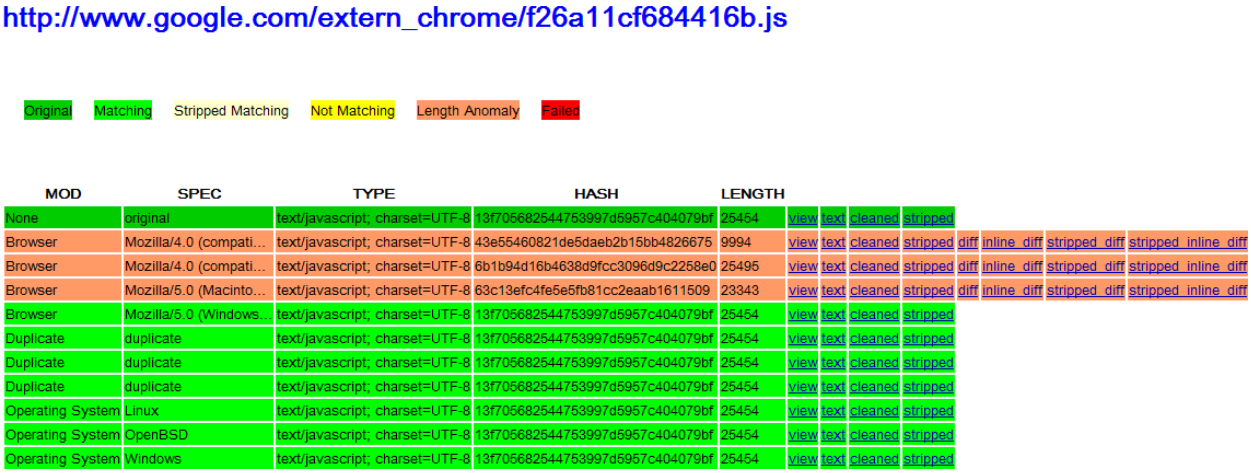



**Figure 92. Resource Analysis**

As figure 92 shows, clicking on a resource provides details about every response that was received for the user's configuration. Each response is augmented with information about how the client fingerprint was

modified for that particular request. Additionally, each response is colored depending on how it compares with the original response. Figure 93, below, provides details about each color.

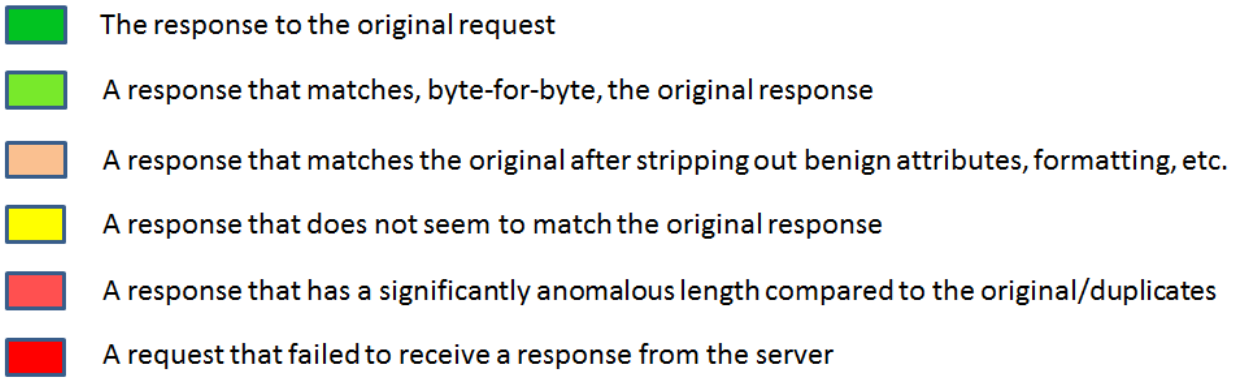



**Figure 93. Response Color Key**

Using this key against the responses seen in Figure 93 tells us that while all nearly all of the responses matched the original response, three of the responses contained a length anomaly. Additionally, looking under the "Mod" column shows us that each of these responses were the result of requests that modified the browser in the client fingerprint. Therefore, the multiplexing proxy could automatically deduce that the resource is tailored toward particular browser fingerprints.

Although some websites are significantly more dynamic, the strategy of color-coding responses still works wells.

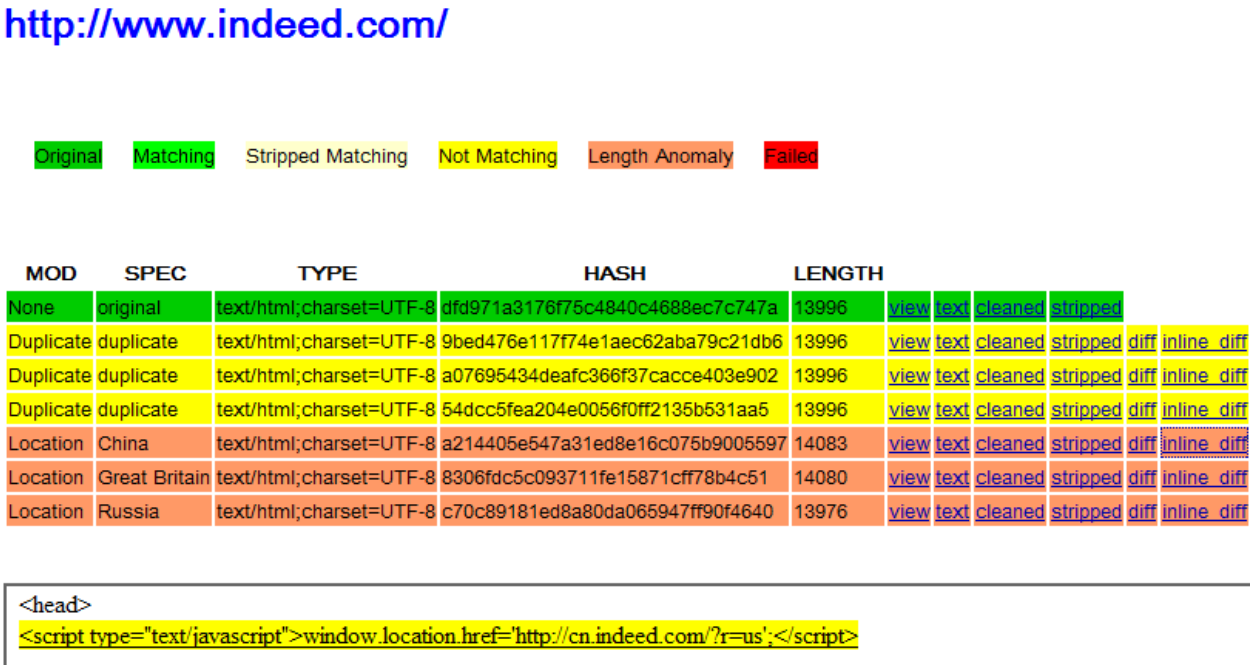



**Figure 94. Dynamic Resource Analysis**

Despite every response to multiplexed requests not matching the original response, the hierarchical analysis and color-coding immediately pinpoint an anomaly in requests that were sent through a remote proxy. Automatically, the multiplexing proxy can classify this resource as being tailored toward specific client locations. Any time there is a consistency in color difference between two different types of request fingerprint modifications, there is sufficient evidence that the server is tailoring the response toward specific client fingerprints.

Depending on the resource type and color, options are provided for analyzing and visualizing the response even further. Typical options include viewing the resource, analyzing the markup or code, and viewing the differences between a response and the original. The differences can either be stripped away from the static content for analysis, or they can be viewed inline with the static content, as is seen in Figure 94.

# WORKS CITED


*Internet Censorship*. (2011, April 23). Retrieved April 24, 2011, from wikipedia.com: http://en.wikipedia.org/wiki/Internet_censorship#Around_the_world

Adar, E., Dontcheva, M., Fogarty, J., & Weld, D. (2008). Zoetrope: interacting with the ephemeral web. *21st Annual ACM Symposium on User Interface Software and Technology*, (pp. 239-248). Monteray.

Adar, E., Teevan, J., & Dumais, S. (2009). Resonance on the Web: Web Dynamics and Revisitation Patterns. *CHI 2009*. Boston.

Adar, E., Teevan, J., & Dumais, T. (2008). Large Scale Analysis of Web Revisitation Patterns. *CHI 2008.* Florence.

Adar, E., Teevan, J., Dumais, T., & Elsas, L. (2009). The Web Changes Everything: Understanding the Dynamics of Web Content. *2nd ACM International Conference on Web Search and Data Mining*. Barcelona: ACM.

*Adblock Plus*. (n.d.). Retrieved from https://addons.mozilla.org/en-US/firefox/addon/1865/

Bolz, F. D. (2005). *The Counterterrorism Handbook: Tactics, Procedures, and Techniques.* CRC Press.

Boyapati, V., Chevrier, K., Finkel, A., Glance, N., Pierce, T., Stockton, R., et al. (2002). ChangeDetector: a site-level monitoring tool for the WWW. *11th International Conference on World Wide Web* (pp. 570-579). Honolulu: ACM.

*Browser Detect*. (n.d.). Retrieved from http://www.quirksmode.org/js/detect.html

Cafarella, M., Madhavan, J., & Halevy, A. (2009, March). Web-scale Extraction of Structured Data. *SIGMOD, 37*(4), 55-61.

*Change Detection*. (n.d.). Retrieved from http://www.changedetection.com

*ChangeDetect*. (n.d.). Retrieved from http://www.changedetect.com

Chawathe, S., & Garcia-Molina, H. (1997). Meaningful Change Detection in Structured Data. *SIGMOD International Conference on Management of Data* (pp. 26-37). Tucson: ACM.

Chen, Y.-F., douglis, F., Huan, H., & Vo, K.-P. (2000). TopBlend: An Efficient Implementation of HtmlDiff in Java. *WebNet2000 Conference.* San Antonio.

Cho, J., & Garcia-Molina, H. (2000). The Evolution of the Web and Implications for an Incremental Crawler. *26th International Conference on Very Large Data Bases.* Cairo.

*Copernic Tracker*. (n.d.). Retrieved from http://www.copernic.com/en/products/tracker/

*Diff*. (n.d.). Retrieved from http://en.wikipedia.org/wiki/Diff

*Diff, Match, and Patch libraries for Plain Text*. (n.d.). Retrieved from http://code.google.com/p/google-diff-match-patch/

Dontcheva, M., Drucker, S., Salesin, D., & Cohen, M. (2007). Changes in Web Page Structure Over Time. *UW, CSE*.

Douglis, F., Ball, T., Yih-farn, C., & Koutsofios, E. (1998). The AT&T Internet Difference Engine: Tracking and Viewing Changes on the Web. *World Wide Web, 1*(1), 27-44.

*Femtoo*. (n.d.). Retrieved from http://femtoo.com

Fetterly, D., Manasse, M., Najork, M., & Wiener, J. (2003). A Largescale Study on the Evolution of Web Pages. *World Wide Web*.

*FollowThatPage*. (n.d.). Retrieved from http://www.followthatpage.com

Greensberg, S., & Boyle, M. (2006). Generating Custom Notification Histories by Tracking Visual Differences Between Web Page Visits. *Graphics Interface*, (pp. 227-234). Quebec.

*HogWash*. (n.d.). Retrieved from http://hogwash.sourceforge.net/docs/overview.html

*HTTP:BrowserDetect*. (n.d.). Retrieved from http://search.cpan.org/~oalders/HTTP-BrowserDetect-1.19/lib/HTTP/BrowserDetect.pm

*Infominder*. (n.d.). Retrieved from http://www.infominder.com

*Internet Owl*. (n.d.). Retrieved from http://www.internetowl.com/

*IP Personality*. (n.d.). Retrieved from http://ippersonality.sourceforge.net/

*IP2Location*. (n.d.). Retrieved from http://www.ip2location.com/

*Iplog*. (n.d.). Retrieved from http://ojnk.sourceforge.net/stuff/iplog.readme

*IpMorph*. (n.d.). Retrieved from http://blog.hynesim.org/en/ipmorph/

James, R. (2009, March 18). *A Brief History of: Chinese Internet Censorship*. Retrieved April 24, 2011, from www.time.com: http://www.time.com/time/world/article/0,8599,1885961,00.html

*Jericho HTML Parser*. (n.d.). Retrieved from http://jericho.htmlparser.net/docs/index.html

Kim, J., & Lee, S. (2005). An Empirical Study on the Change of Web Pages. *7th Asia-Pacific Web Conference.* Shanghai.

Koehler, W. (2002). Web Page Change and Persistence: A Four-Year Logitudinal Study. *Journal of the American Society for Information Science and Technology, 52*(2), 162-172.

Kwon, S., Lee, S., & Kim, S. (2006). Effectice Criteria for Web Page Changes. *Eight Asia-Pacific Web Conference.* Harbin.

Liu, B., Zhao, K., & Yi, L. (2002). Visualizing Web Site Comparisons. *World Wide Web*, (pp. 693-703). Honolulu.

Liu, L., & Tang, W. (2000). WebCQ - Detecting and Delivering Information Changes on the Web. *Ninth Internation Conference on Information and Knowledge* (pp. 512-519). McLean: ACM.

Mikhaiel, R., & Stroulia, E. (2005). Accurate and Efficient HTML Differencing. *13th IEEE International Workshop on Software Technology and Engineering Practice* (pp. 163-172). IEEE.

Nadamoto, A., Tanaka, K., & Ma, Q. (2005). B-CWB: Bilingual Comparative Browser Based on Content-Synchronization and Viewpoint Retrieval. *World Wide Web, 8*(3).

Nadamoto, R., & Tanaka, K. (2003). A comparative Web Browser (CWB) for Browsing and Comparing Web Pages. *12th International Conference on World Wide Web* (pp. 727-735). Budapest: ACM.

*Nmap*. (n.d.). Retrieved from http://nmap.org

*NoScript*. (n.d.). Retrieved from http://noscript.net

Ntoulas, A., Cho, J., & Olston, C. (2004). What's New on the Web? The Evolution of the Web from a Search Engine Perspective. *World Wide Web*, (pp. 1-12). New York.

*NVD*. (n.d.). Retrieved from NVD: http://nvd.nist.gov/

Olston, C., & Pandey, S. (2008). Recrawl Scheduling Based on Information Longevity. *World Wide Web*, (pp. 437-446). Beijing.

*P0f*. (n.d.). Retrieved from http://lcamtuf.coredump.cx/p0f.shtml

*Page2RSS*. (n.d.). Retrieved from http://page2rss.com

*Panopticlick*. (n.d.). Retrieved from https://panopticlick.eff.org

Pariser, E. (2011, March). Beware online "filter bubbles".

*Passive Fingerprinting*. (n.d.). Retrieved from http://www.symantec.com/connect/articles/passive-fingerprinting

Patil, V. (n.d.). *IFrame Injection Attack is most command and most basic cross site scripting (XSS) attacks*. Retrieved from http://www.computeruser.com/tutorials/iframe-injection-attack-is-most-common-and-most-basic-cross-site-scripting-xss-attacks.html

*PAW*. (n.d.). Retrieved from http://paw-project/sourgeforge.net/

*Proxy List*. (n.d.). Retrieved from proxy-list.org: http://proxy-list.org

Qiu, J. L. (1999). Virtual Censorship in China: Keeping the Gate Between the Cyberspaces. *International Journal of Communications Law and Policy*, 4.

*RFC2616 - Hypertext Transfer Protocol -- HTTP/1.1*. (n.d.). Retrieved from http://www.w3.org/Protocols/rfc2616/rfc2616.html

*Scapy*. (n.d.). Retrieved from www.secdev.org/projects/scapy

*Security Technologies*. (n.d.). Retrieved from http://www.innu.org/~sean/

*Selenium*. (n.d.). Retrieved from http://seleniumhq.org/

*SFB*. (n.d.). Retrieved from http://code.google.com/apis/safebrowsing/

*TestSwarm*. (n.d.). Retrieved from http://TestSwarm.com

*Top Sites*. (n.d.). Retrieved from www.alexa.com: http://www.alexa.com/topsites

*Tracking Changes*. (n.d.). Retrieved November 2010, from http://www.rba.co.uk/sources/monitor.htm

*Tracking Changes to Web Content*. (n.d.). Retrieved from http://www.irba.co.uk/sources/monitor.htm

*Update Patrol*. (n.d.). Retrieved from http://www.updatepatrol.com/

*Update Scanner for Firefox*. (n.d.). Retrieved from http://addons.mozilla.org/en-US/firefox/addon/3362

*Using Object Detection to Sniff Out Different Browsers*. (n.d.). Retrieved from http://www.javascriptkit.com/javatutors/objdetect3.shtml

Wang, Y., DeWitt, D., & Cai, J.-Y. (2003). An Effective Change Detection Algorithm for XML Documents. *International Conference on Data Engineering*.

*Watch That Page*. (n.d.). Retrieved from http://www.watchthatpage.com

*Website Watcher*. (n.d.). Retrieved from http://aignes.com/

*Websnitcher*. (n.d.). Retrieved from http://websnitcher.com

Xing, G., Xia, Z., & Wang, H. (2008). Xdiff+: a visualization system for XML documents and Schemata. *46th Annual Southeast Regional Conference on XX* (pp. 40-45). Auburn: ACM.

*YUI Compressor*. (n.d.). Retrieved from http://developer.yahoo.com/yui/compressor/