Chokegati: a browser extension to mitigate resource hints vulnerabilities in HTML5

MSc Internship
Cyber Security

Roshan Rangwani
x17131120

School of Computing
National College of Ireland

Supervisor: Rohan Singla

National College of Ireland
Project Submission Sheet – 2017/2018
School of Computing

Student Name:	Roshan Rangwani
Student ID:	x17131120
Programme:	Cyber Security
Year:	2017
Module:	MSc Internship
Lecturer:	Rohan Singla
Submission Due Date:	17/09/2018
Project Title:	Chokegati: a browser extension to mitigate resource hints vulnerabilities in HTML5
Word Count:	5934 words

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:	
Date:	14th September 2018

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
3. Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Chokegati: a browser extension to mitigate resource hints vulnerabilities in HTML5

Roshan Rangwani
x17131120
MSc Internship in Cyber Security

14th September 2018

Abstract

There has been drastic improvement in the speed of web over the years. Because of the introduction of resource hints feature of HTML5, web has become much faster by utilizing the predictive loading of resources. This feature has become the principle road for the attacks on users privacy and security. Attacks also include popular cyber-attacks like Cross Site Request Forgery, Cookie stuffing. Mitigation to these attacks are slow and easily bypassed. Our research focuses on reviewing the present defenses and provide a novel approach to the defensive challenges. In this paper, we present CHOKEGATI, a chrome browser extension based on URL filtering approach using DOM model. By doing so, we are able to mitigate risks involved.

1 Introduction

Usage of web has become an essential part of daily life. Recent years have notice a tremendous growth in the usage, due to its easily accessible and faster, growing demand of web has increased the requirement of faster web. Hence, to fulfill growing demands of faster web HTML5 is been introduced. Its some features like geolocation, local storage, speed and platform independency have made HTML5 widely used and adopted. In a recent study conducted by stats incore reveals about 153 companies out of top 500 US Fortune companies have preferred HTML5 over other technologies as their website technology[1]. To easily access all the functionality of web, a user agent is required by the user which acts as a bridge between web and the user. User agent often also called browser stores information about the user which include browsing history, user sessions, cookies, cached data and user behavior. A user holds all the responsibilities of actions and requests initiated by the browser (Vlajic et al.; 2017a). Many of the tasks performed by the user agent are conducted in background which include cookies send, page loading, and connecting domain name servers. Browser require multiple types of resources in order to load a web page completely for user. While calculating performance of a user agent, resource loading time, rendering time and script execution speed are important factors to consider. A company providing survey result, when conducted, 1% sale lose to

[1] Statistics provided by Incore on HTML5 adoptation: `https://www.incore.com/Fortune500HTML5/`

Amazon when web page delay increased by minute 100msec, while the revenue of Google will be affected by 20% when delay increased by 500msec this shows how much of revenue depends on loading time of the web page. Loading time can be decreased by server and client side. User agent being the client side use techniques, which includes prefetching, caching to improve web experience and performance. Web speed is always considered to be important factor. To reduce delay and increase speed HTML5 came up with resource hints feature in 2016. Resource hints uses <link >of HTML and based on the idea of preloading with predicting user future need in order to provide better experience and speed. Aim of implementing resource hints is to provide zero delay web page loading as resources are loaded in the background. Preconnect, prerender, domain name server prefetch and prefetch are the four methods used for resource preloading by resource hints. Execution of resource hints doesnt require any user involvement, are executed by the user agent as background process. Figure 1 explains the working of resource hints in chrome browser

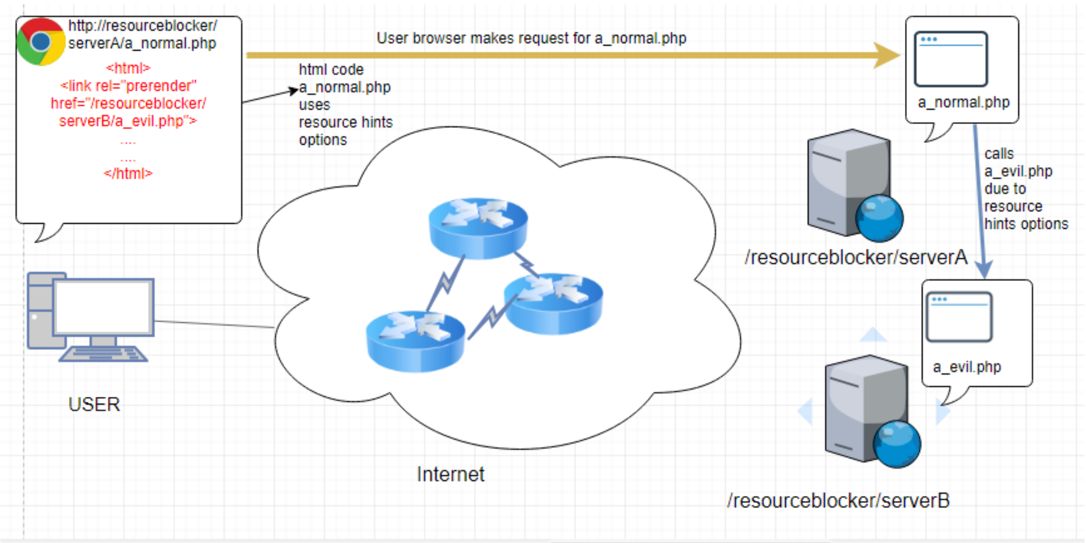


Figure 1: Execution of resource hints in chrome browser

Apart from positive impacts, a researcher Vlajic, Shi, Roumani and Madani (2017b) experimented and concentrated on negative impacts of the resource hints. Their research provides evidences of the attacks possible because of resource hint. Attacks include Cookie stuffing, Cross-site request forgery (XSRF), Etag tracking, Framing Attack, targeted DOS attack, Data Analytics Pollution attack. These attacks have very high impacts on privacy and security of the user data which may lead to complications, ideally risks must not be neglected. In the recent years, many of the attacks are made possible because of the resource hints feature, we have discussed some of the cases involved child pornography, ebay affiliated cookie loss and cross site request forgery attacks, in later part of the paper, under the case study scenarios.

Motivation: While surfing blogs on cyber-attacks and cases, we went through some cases about resource hints. Further, working in the direction we examined that there lies enormous opportunities to new research work. Previous research work, are focused on the defense mechanism which are less efficient to mitigate risks, which provides us with opportunity and motivation to conduct research. **Research question We enhance**

HTML5 security with the help of browser extension using document object model based filters to mitigate risk involved due to resource hints features. The rest of this paper is structured as follows. We provide an overview of related work in context to the current work as Related Work having 3 subsections as (2.1) Research related to prefetch and dns prefetch (2.2) Privacy implications due to resource hints (2.3) Present defenses to the attacks caused due to resource hints feature. In section 3 we discuss about the methodology which will be followed. Section 4 and 5 elaborate the implementation and evaluation. We draw some discussions in Section 6 by concluding our research and providing future scope.

2 Related Work

Resource hints is an emerging concept, and recently cyber-attacks have emerged because of its usage. This needs to be improved with proper mitigation techniques. There are few papers on such an emerging concept already published which are discussed in the following sub sections.

2.1 Research related to prefetch and dns prefetch

In order to gain better understanding about working of resource hints, we will discuss about the browser and loading resource relationship, then later in this section we will discuss about the techniques, used to load resources and decrease load time with their privacy issues. To load web page, browser needs to follow a procedure either dynamic or concurrently which must be executed. Each component of the web page, forms a document object model and internally represent as DOM tree, render tree is formed by using DOM tree once it gets formed and by painting above render tree layout is formed. Figure 2 explains DOM tree and rendering of the objects by browser.

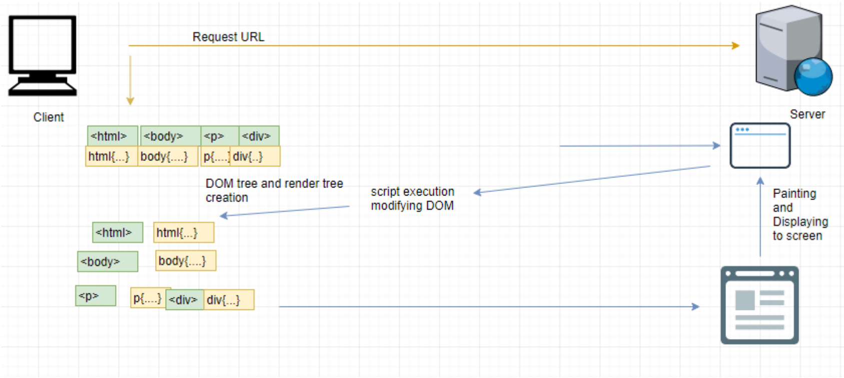


Figure 2: DOM tree and rendering of objects by browser

Many resources are required by user agent, for loading a web page. When the browser request for the foremost resource i.e. main source, it may also request for the remaining other resources which are required to load the content (Wang et al.; 2012). A decent amount of time, is required by the browser to request for the resources and loading it as a web page. Researchers have used various techniques, in order to decrease this load time and make web faster which also involves predictive gathering of resources in background once user gets busy reading the loaded page. One such technique which works on the same technique and principal and also an integral part of Resource Hints is prefetching. Mainly, to increase speed and reduce latency of web this technique is used and is well researched in literature. A researcher Padmanabhan and Mogul (1996) in their research, mainly focused on the idea of predictive loading in server side. Researchers learned and analysed, behavior of the clients by sending client information to the server with help of client cache. Yang, Zhang and Li (2001) used dependency graph method to maintain pattern of client and achieved decrease in latency, although a huge number of packets were increased in the network. Yang, Zhang and Li proposed and used an algorithm Greedy Dual Frequency Size (GDSF), in which a predictive factor was added to reduce the usage of cache, which is required for prefetching. Author for predicting web surfers future requests, collected and analyzed web logs. Apart from improved performance the system also supported present architecture, but failed to support when cache size increased. Another researcher De La Ossa, Gil, Sahuquillo and Pont (2007) pointed, that much of the work done concentrated on predictive algorithm. Author focused on making web prefetching efficient with improved performance. Author followed an approach combining dependency and double dependency graph. However, authors were able to reduce latency by a peripheral proportion. A research conducted by Fan, Cao, Lin and Jacobson (1999) followed an approach of prefetching in browsers and cache proxies. In their approach, they utilized network idle time, efficiently for prefetching between the requests made by the user. Research provided a lot of improvement in the latency with approximately zero extra traffic in the network. However, to improve results many non-standardized assumptions made, which may not be suitable in each case. Throughout the years much work has been done for enhancing execution of prefetching, however Jourdan (2007) identified the negative impacts of the prefetching on the users privacy. Their research provided the evidences of prefetching method responsible for creating security issues in the system. Also, prefetching indirectly download pages, for which request are never made which is always a point of concern over privacy. Author also found prefetching, making difficult for website owner to differentiate between, the real and fake request made by the user. Some evidences were found in case of automatic prefetch links, lead to web applications crash making it unavailable for the user. Moreover, attacks can be prevented by making some change in default settings of the user agent, which are expected to be altered rarely. Inorder, to protect user and web, instead of ignoring the above evidences and leaving users with choices, steps can be taken which may protect privacy of the users.

Another popular technique and most used to reduce web latency named as DNS prefetching. A study done by Cohen and Kaplan (2002) provide us the concept of pre-resolving, pre-connecting and pre-warming. Their research also provide evidences about the TCP connections made by the user agent, in order to support DNS prefetching and effectiveness of the techniques. In their analysis, done on the server response to translate DNS lookups, the authors noticed and identified near about 10% servers surpasses the limit time of 3 secs, considered to be a huge amount of delay. Translation time of the servers are mostly dependent on the locality. Author with their research proposed an approach by combining pre-resolving, pre-warming and pre-connecting named as pre-transfer in order to minimize latency. A significant amount of long waiting times were reduced, when the above approach used. However, the research limited to high overhead introduced to the user agent. In a study done by Krishnan and Monrose (2011) on

DNS prefetching claims, providing a significant quicker experience to user. Author also provide evidences of top search engines like google using prefetch technique, to support instant search feature. Usage will significantly increase in near future. Although, post analysis result shows relative overhead time increment for the server. Their research also provided evidences of violation of user privacy and security issues in the system. The author Krishnan and Monrose (2010) found DNS prefetching violating privacy, by pre-resolving each and every hyperlinks at loading time of the web page. Author also questioned about the implementation of the technique and risks involved. In the next subsection, we will further discuss about the other popular used techniques and security issues with resource hints features.

2.2 Privacy implications due to resource hints

We will first discuss about preconnect and prerender in this section, later we will discuss about privacy implications of resource hints. TCP preconnect is a prefetching technique based on connections on TCP protocols. Deng and Manoharan (2015) performed analysis on compatibility of pre-connect with protocols HTTP1.0 and HTTP 1.1. According to authors, research pre-connect is limited to single resource under HTTP1.0 per TCP connection. However, to achieve multi fetching resources HTTP 1.1 can be used to reduce web latency. According to Deng and Manoharan (2015), preconnect create a persistent connection with the server leading to increase overhead of the server. Newton, Jeffay and Aikat (2013) when analyzed web traffic, provided empirical evidences of increase in the number of unused connections and web activity when preconnect feature is used by the user agents. Also author stated that preconnect feature may lead to Denial of service attacks in the near future. Another popular prefetching technique, used by modern browser is prerender. Browser utilizes pre-fetcher module to pre-render web resources according to a study conducted by Gudla, Sahoo, Singh, Bose and Ahamed (2016). Rendering resources also include JavaScript, HTML files and some graphical files like images. Pre-fetcher use internal engine of the user agent to ensure prior and speedily loading of the web page. Pre-render with the use of pre-fetcher, helps to reduce launching time of the web. The author also provided evidences of consumption of huge amount of power and memory when unhandled properly. Author Deng and Manoharan (2015) in their study states that pre-render follows similar procedure like browser follows to load a site page. Procedure also includes creation of DOM objects, by parsing HTML code with an addition of various scripts like CSS and JavaScript. Author, also performs experiment and states prerender works similar to browser with a slight difference, pre-rendered page appear to be in hidden tab, as if the request for page is done explicitly by entering URL in the address bar. Another experiment conducted by (Vlajic et al.; 2017a) provides significant changes in the browser cache when a page is rendered, havent appeared in the browsing history. The author also focuses on the possible attacks when pre-render feature is used. We have discussed four types of prefetching techniques in above sections. In 2016 a draft presented by w3 combined all four techniques, in a single feature with the introduction of HTML5, to speed up the loading time of the web, which referred as resource hints. Although, the implementation of resource hints is yet to be standardized, and havent been researched in the literature, but because of the speedy features and supported by 90% of modern browsers, the usage of resource hints have gained enough popularity. However, a researcher Vlajic, Shi, Roumani and Madani (2017b) while conducting experiments found web applications security issues when resource hints features are used. Security issues also include pruning system to cyber-attacks like XSRF, Framing Attack, Targeted DOS, Data Analytics Pollution Attack, Etag Tracking Attack and Cookie Stuffing attack. Author also states, about the possible security and privacy breach of the user when feature is misused. Author also discussed about the defense mechanisms about the attacks and result of vulnerability scanners which are used at present. However, research also provide evidences of scanners failing to detect possible attacks because of resource hints. While the researchers bring a light to possible security issues and attacks done using resource hints, but failed to provide a defense mechanism against the following. In next section we will understand about the present defense mechanism of these attacks.

2.3 Present defenses to the attacks caused due to resource hints feature

We have discussed above about the possible attacks when resource hints features are used. In this section we will study and discuss about the present defense mechanisms of the attacks. Cross site request forgery attack (XSRF) takes place, when user agent sends cookies or the cached credentials automatically, when a request from cross domain arrive according to (Maes et al.; 2009). XSRF attack is one of the top 10 OWASP[2] vulnerabilities and mainly perform requests like funds transfer or changing other personal details like email. In their research author proposed a solution to mitigate XSRF focusing on client side. Author, also analysed cross domain requests which are evolved . Following the above, approach much of the false positives requests were reduced by relaxing server policies. Over the years, so many defense mechanism have been proposed. Author Jovanovic, Kirda and Kruegel (2006) while conducting research provided evidences of XSRF occurring and cases of people getting harm. Author also proposed server side defense methods and automatic mechanism to prevent XSRF. Another researcher Barth, Jackson and Mitchell (2008) provide defenses by blocking referrer and secret tokens using same origin header. They were able to reduce privacy concerns of user. Defense mechanism only work when JavaScript is enabled, is a limitation to the research. (Vlajic, Shi,

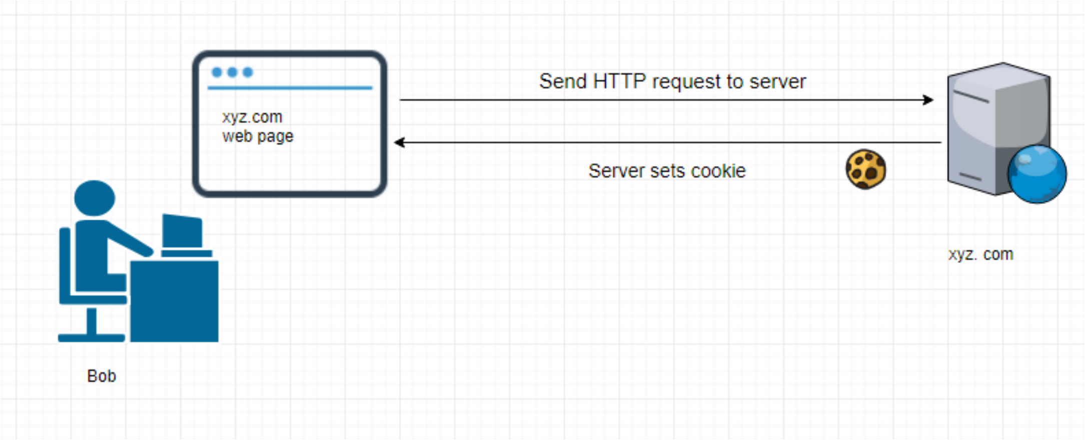


Figure 3: Bob request for xyz.com and gets cookie set in the browser

[2] Owasp `https://www.owasp.org/index.php/Main_Page`

Roumani and Madani; 2017a) also conducted studies how XSRF can occur by exploiting vulnerability using resource hints feature of HTML5.In the figure 3 and figure 4 below explains how XSRF can occur with use of resource hints.

Cross Site Framing Attack is another type of attack, which involves planting of the false evidences on a computer by exploiting browser vulnerability or weaknesses of the website. Author Gelernter, Grinstein and Herzberg (2015) conducted research on cross site framing attack on search engines like Google, Bing, Yahoo also found false evidences in result being planted in the computer. Author provided the details regarding the defense mechanism against these attacks and how it can work, with different services like web service, web browsers and other forensics software. While researching on the cross site framing attack, we noticed that some of the innocent people were charged under child pornography due to false evidences planted on their system and later on when investigation was made, the forensic team cleared that it was due to malware frames which were planted resulted in web session redirecting them to such websites.[3]

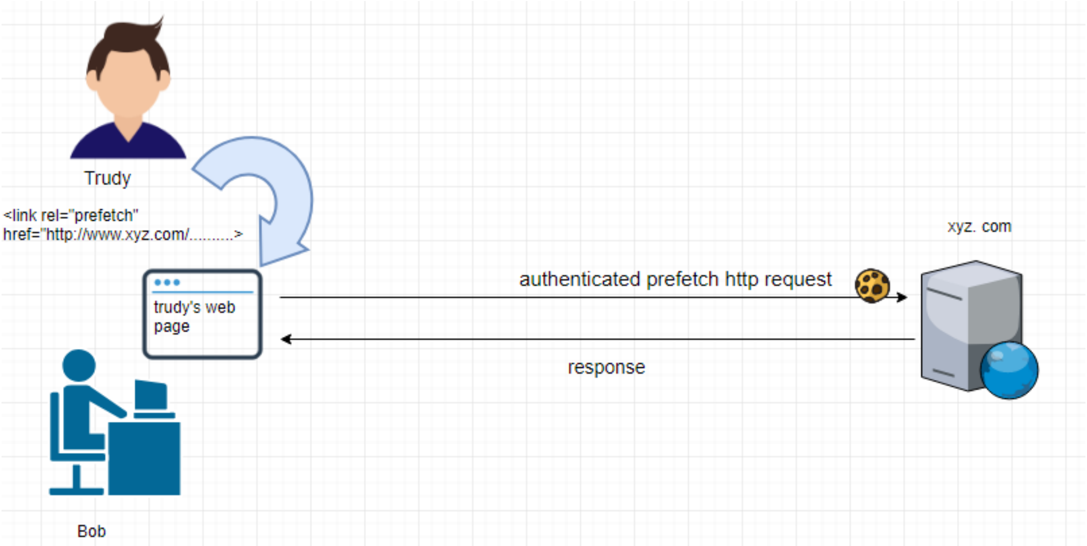

Trudy
<link rel="prefetch" href="http://www.xyz.com/.........>
trudy's web page
authenticated prefetch http request
response
xyz. com
Bob

Figure 4: Bob visits trudy site which is using resource hints and sends cookie authentication

Case study: Data Analytics Pollution Attack Data analytics pollution attack is latest type of attack and havent much addressed in the literature. Under this attack logs of the web server, are polluted in order to create a negative impact on competitors business intelligence. Thus, using resource hints feature of HTML5 one can easily pollute and plant false enquiries of the product which may have high impact on the businesses dependent mainly on online revenues.

Case study: Targeted DOS Attack Under this attack an attacker specifically targets victims frequently visited websites by sending prefetch requests such that victims IP address gets blacklist by the server. Hence, its called targeted DOS attack as its specific to a target. Main aim of the attack is to get victim unable to attend the services. A study conducted on targeted denial of service by Kuzmanovic and Knightly (2006) very efficiently displays network implications and TCP connections. Author used analytical

[3] Article on child pornography case: `https://www.theregister.co.uk/2009/11/09/malware_child_abuse_images_frame_up/`

modeling approach for analysing DOS patterns, they achieved remarkable result and also able to detect and reduce attacks by a notable amount. The defensive approach followed reduces the performance of the system and insufficient to prevent DOS attacks which are categorized as browser based.
Case study: Cookie Stuffing Attack Affiliate marketing is one of the popular form, in which marketers need to pay commission for converting traffic i.e. pay per sale advertising method. The merchant need to pay commission to the affiliate for each and every sale made. The author Chachra, Savage and Voelker (2015) in their research also discuss about the affiliated cookies and cookie stuffing. Author, also focuses on how these cookies are abused and what are the present mitigations of the cookie stuffing. One such fraud case happened with EBay for affiliate cookie fraud of $28M because of the cookie Stuffing technique [4].
Case study: ETag tracking ETag is a part of Hyper Text Transfer Protocol, which allows web caches work more efficiently. ETag field is of 81864 bits long and hence its very easy to store identity of the user, which can be used by the companies to track users. Hence, with very little effort the following ETag can be changed. El Masri and Vlajic (2017) in their research conducted on top5 browser extensions, which are used by most of the users, in order to prevent XSRF and Cross site framing attack. Author, analysed extensions on the basis of the popular HTML tags such as iframe, prefetch, img in code and surprisingly found that, none of the extensions where able to protect, their user when resource hints features were used. Also author in their analysis providing details about extensions, mostly are focused on examining tags such as img, href, anchor tag, while ignoring the link tag which is also the root cause of attacks possible because of the resource hints. We have conducted research on resource hints, while conducting our research, we found that to protect these attacks most of the extensions are focused on tags which are not related to the resource hints. Therefore, unable to protect the users, against the vulnerabilities. Hence, there exist enough scope to research further in order to protect users against these attacks by focusing on the root cause of the attacks i.e. focusing on the tags such as <link>. We discussed in the other sections about the formation of the DOM tree of HTML code. A researcher Gupta, Kaiser, Neistadt and Grimm (2003) provided us the concept of the DOM based extractions and how to use extracted contents. Another researcher Maes, Heyman, Desmet and Joosen (2009)provided proof of concept of detecting malicious web pages on the basis of the DOM objects extracted. The author performed lexical analysis, blacklisting and predictors to detect malicious links. Model proposed is extremely popular and statically blockage ratio of malicious pages are higher than other models, thus we expect efficient results. We therefore, accepted this approach as our model with slightly modifying the input URL as providing link tag URLs. In the next section, we will discuss about details of methodology.

3 Methodology

It is very important to follow and understand the flow process and connection between the different components. Also, to understand task done by the specific component. We have researched different methodologies, which can be applied to defend against the resource hints, among all DOM based filter method fits best because of the standardization and

[4] Article on Ebay affiliate cookie fraud: `http://uk.businessinsider.com/ebay-the-fbi-shawn-hogan-and-brian-dunning-2013-4?r=US&IR=T`

easy to understand. Model applied before have achieved high result with better accuracy. DOM based URL filter have been explained by Maes, Heyman, Desmet and Joosen (2009). Figure 5 shows the graphical representation of the methodology

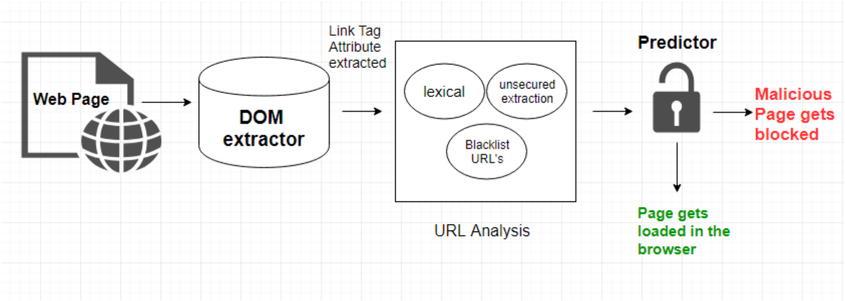
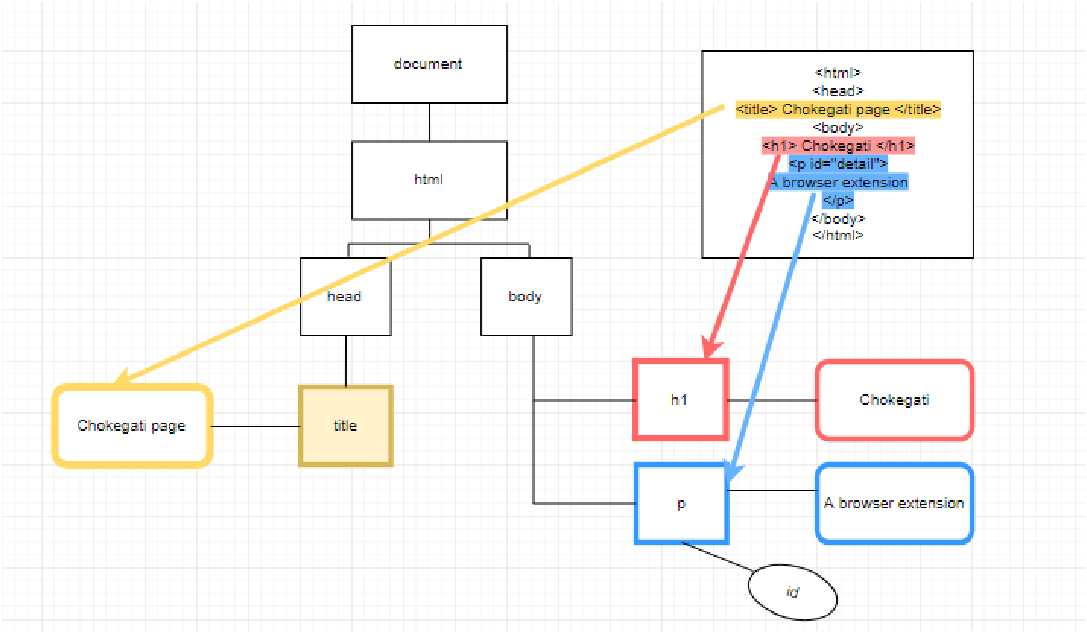

Web Page
DOM extractor
Link Tag Attribute extracted
lexical
unsecured extraction
Blacklist URL's
URL Analysis
Predictor
Malicious Page gets blocked
Page gets loaded in the browser

Figure 5: Flow Diagram of methodology used.

3.1 Extracting elements from DOM extractor

As discussed above in section 2.1 each and every HTML element will be represented as object in DOM tree. We are here performing the extraction of the link element which is also the main source of resource hints. Using HTML DOM model we will access link elements like dns-prefetch, pre-connect, pre-fetch and prerender. In the figure 6. below HTML elements of a web page are represented in a DOM tree.



```
<html>
<head>
<title> Chokegati page </title>
<body>
<h1> Chokegati </h1>
<p id="detail">
A browser extension
</p>
</body>
</html>
```

Figure 6: Representation of DOM objects of a simple web page in DOM tree

3.2 URL filtering and analysis

We have extracted the URL from above DOM extractor and performed analysis on the same. We performed lexical analysis, host based, blacklisting domains and unsecured functions. Working of every tasks are discussed in detailed below:
Lexical Analysis of the URL : Lexical analysis is done on the extracted URL to gather basic information like length of the URL, primary domain, longest word etc. While advanced lexical analysis is performed to gather IP address of the URL and keyword present in the URL. To perform lexical analysis of the URL JavaScript is used as scripting language.
Blacklisting URL: Although, this method is old fashioned but very effective, provides a faster results, saving a lot of time. If the URL is already present in the list of blacklist URL, it gets blocked. However, using alone blacklisting as a filtering option is not suitable to perform appropriate analysis. But once, used with other filtering options act as an asset to the predictor model which improves performance.
Unsecured elements extraction: The URL or the string inside the link element is extracted with the help of DOM extractor. We will extract number of iframes, harmful functions, script and other unsymmetrical or suspicious strings. Analysis is also done on the basis of the keywords present such as banking, username, confirmation code. For conducting we will be using HTML, JavaScript.

3.3 Working of Predictor

As the name suggest predict, predictor acts as the brain and act as decision maker. Predictor is supplied all the input of the previous analysis like lexical, blacklisting, html extractor. Once content are provided as input predictor performs predictive analysis and provides result to allow access link to the browser or to block. Browser is provided with the feedback. Predictor updates database if URL found to be vulnerable.

4 Implementation

To implement the above model we have followed prototyping model approach. Prototyping approach makes us understand better about the system, while in the developing phase, identifying the missing functionalities and detecting errors is easy. We have utilized HTML, JavaScript, CSS, PHP and JSON as programming and scripting languages to develop browser extension for chrome named as chokegati. In the below section implementation of the steps been explained.

4.1 Extracting elements from DOM extractor

When a web-page is loaded DOM object is formed for each element. We have implemented DOM extraction to extract link tags which is the main source of the resource hints. Hence after getting link tag extracted we perform further analysis. To extract objects we have programmed scripts which is written using JavaScript. Here is the snippet of the code.

```
<link rel=“prerender” href=“/chokegati/serverB/a_evil.php”>
<script>
function getresourcehint() {
```

```
var x = document.getElementsByTagName(“link”)[0].getAttribute(“href”);
document.getElementById(“resourcehint”).innerHTML = x;
}
</script>
```

4.2 Implementing Lexical Analysis on URL

The filtered string under the link tag is taken into analysis using different JavaScript functions in order to observe information about the attribute URL like length of the URL, length of the primary domain, IP address if any. Below is the sample snippet of obtaining the attribute value for the resource hints option prerender.

```
<link rel=“prerender” href=“/resourceblocker/serverB/a_evil.php”>
document.getElementById(“href”).size
```

4.3 Performing Host Based Analysis

We have observed the hostname provided in the URL. Whether the hostname provided is of the same origin or the cross domain. For example if host is related to music but in the URL its contacting some xyz bank then URL automatically become suspicious. Hence we obtained host details of the URL with following snippet

```
getLocation(“/resourceblocker/serverB/a_evil.php”);
```

4.4 BlackListing Checker

It is an important step if the URL present in database of already blacklisted by the search engines, ad blocks then it will directly be blocked owing to trust the database collect from the search engines. We have collected already blacklisted URL using google search engine, ad blockers and saved in the backend with which each time URL is verified before passing into predictor.

4.5 Unsecured elements extractor

If the attribute doesnt contains any URL it may contain some harmful string or HTML element which may lead to vulnerabilities. Hence, with the HTML element extractor we perform the sanitization of the string which removes all the harmful characters, tags by which vulnerability can occur. We have use different sanitization function in order to sanitize and clean string under the resource hint attribute.One of the snippet is:

```
chokegati.sanitize(‘<img src=xyz onerror=alert(1)//>‘);
```

4.6 Working of predictor

Predictor analyses the result obtain from the previous steps and provide decision to the browser for allowing or blocking the content. Predictor allows if no suspicious URL is found and blocks if any suspicious pattern is found and increase the number of links block. Combining all these technologies and implementation, we created a chrome browser extension and added to the browser.

5 Evaluation

To evaluate our browser extension and collect artifacts we have created sets of pages one set is normal pages and the other set consists of evil pages. Normal set of pages calls evil one using different resource hints options. While performing experiment we initiated request/retrieval of the pages named a_normal.php, b_normal.php, c_normal.php and d_normal.php. The observations in table 1 are summarized as: When a_normal.php and b_normal.php are requested which contains dns-prefetch and preconnect options of resource hints respectively in the code. We havent recognized any change in the chrome browser history, cache or in cookie creation. Hence, these option can be considered to make lower impacts. In figure 7 we have shown the experimental setup to evaluate the effect on chrome browser.

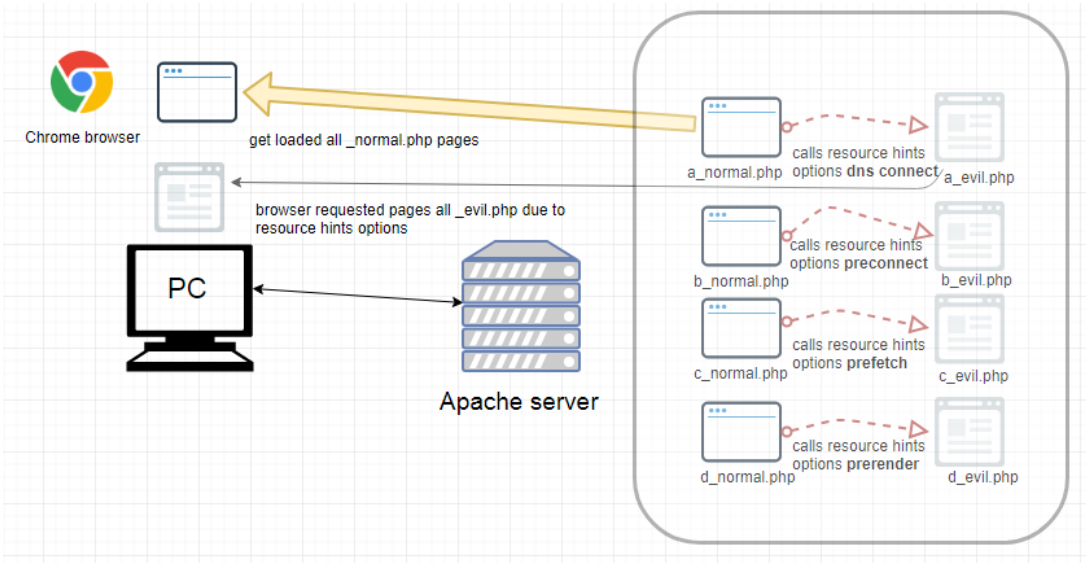

Chrome browser
get loaded all _normal.php pages
browser requested pages all _evil.php due to resource hints options
PC
Apache server
a_normal.php
calls resource hints options dns connect
a_evil.php
b_normal.php
calls resource hints options preconnect
b_evil.php
c_normal.php
calls resource hints options prefetch
c_evil.php
d_normal.php
calls resource hints options prerender
d_evil.php

Figure 7: Experimental setup for evaluation with resource hints options using chrome browser.

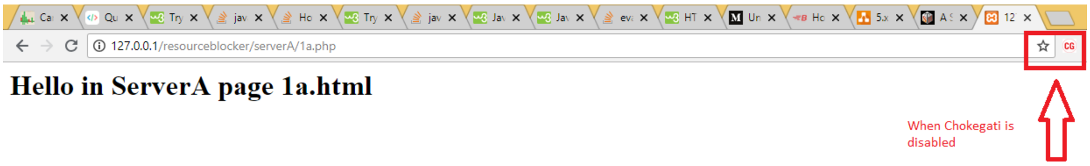
When c_normal.php and d_normal.php are requested by the browser which contains prefetch and prerender options of resource hints respectively in the code. We recognized a number of changes in the browser history, cache and cookie which are related to c_evil.php and d_evil.php are found. We noticed and collected artifacts shown in the table 1 below that cookies created which are associated with the pages . Pages appear in cache even when request is not made explicitly.

Resource hints option	Chrome History Changes	Chrome Cache Changes	Cookies effect
DNS-prefetch	no change	no change	no change
Preconnect	no change	no change	no change
Prefetch	no change	present in cache	cookies are created
Prerender	no change	present in cache	cookies are created

Table 1: Artifacts collected using chrome browser

5.1 Experiment and evaluation with chrome browser

We evaluated chokegati by running our programmed scripts which directly contains resource hints code intentionally. In order, to ensure proper working of the extension we run programmed page multiple times and evaluated performance of the browser. In figure 8,9 and 10 we have shown the experimented performed and how chokegati able to block the URL. In figure 8 we have executed page containing resource hint when chokegati is disabled while in fig9 we executed when chokegati is enabled. Fig 10 shows the demo code of the page executed.

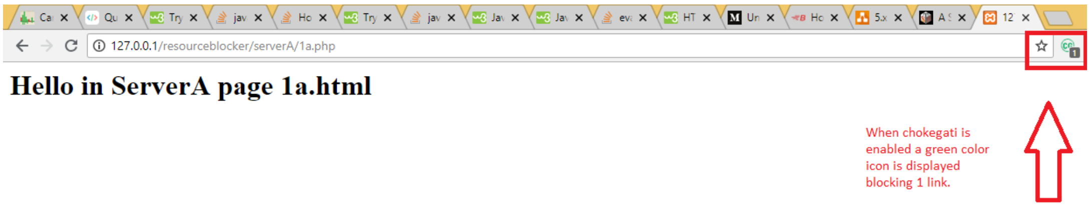


127.0.0.1/resourceblocker/serverA/1a.php

Hello in ServerA page 1a.html

When Chokegati is disabled

Figure 8: Resource hints link executed when chokegati extension is disabled

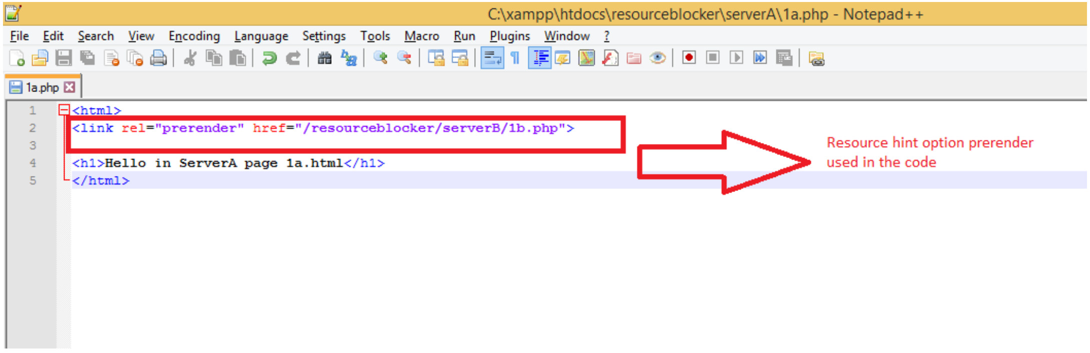


127.0.0.1/resourceblocker/serverA/1a.php

Hello in ServerA page 1a.html

When chokegati is enabled a green color icon is displayed blocking 1 link.

Figure 9: Resource hints link blocked when chokegati extension is enabled



C:\xampp\htdocs\resourceblocker\serverA\1a.php - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
1a.php

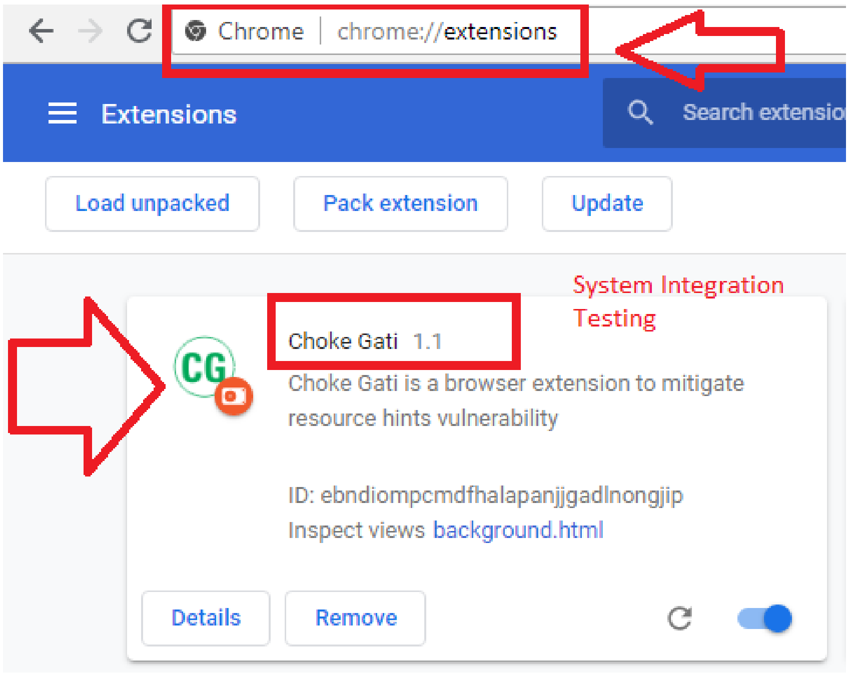
```
<html>
<link rel="prerender" href="/resourceblocker/serverB/1b.php">

<h1>Hello in ServerA page 1a.html</h1>
</html>
```

Resource hint option prerender used in the code

Figure 10: Code snippet of the figure 8 and figure 9 to check functionality test of chokegati

5.2 Test Cases

We have performed three types of testing in the browser extension named chokegati they are smoke testing, regression testing and system integration testing.
Smoke Testing: Whenever a new functionality was added we tested the functionality alone and then we integrated with the existing tested code.
System Integration Testing: We have tested our product with the chrome browser under different cases by enabling the developer mode. We also tested performance changes of the chrome browser. Fig 11 shows us the integration of chokegati with chrome browser.
Regression Testing: We tested whole system when a new functionality was added in order to check working of the browser extension chokegati.



System Integration Testing

Choke Gati 1.1

Choke Gati is a browser extension to mitigate resource hints vulnerability

ID: ebndiompcmdfhalapanjjgadlnongjip

Inspect views background.html

Figure 11: System integration testing of chokegati with chrome

5.3 Discussion

We have experimented various cases of resource hints. We found how easily these attacks can occur, with little effort. We experimented with cases of dns-prefetch, prefetch, preconnect, prerender and found artifacts in the browser. We noticed changes, in the cache and cookies of the browser, when prerender and prefetch options are used. When pages appear in the cache or create cookie, but the page is not present in the history, is automatically a sign of suspicious activity. While most of the request are sent automatically by the browser without the knowledge of user. Hence, it has sparked many issues related to users privacy and security. We studied different cases of resource hints, used for the cyber- attacks like XSRF, Cross site framing attack, E-tag tracking, Cookie stuffing, Data analytics pollution attack and targeted DOS attack. Some of the attacks have very high industrial impacts, as we studied past cases which involved millions of dollars loss to different companies, because of the attacks placed with the use of resource hints. Other attacks involved loss in the data privacy of the user. We evaluated the above situation with our chrome browser extension chokegati and performed testing which include regression, integration and smoke test of the functionalities. We found out chokegati able to mitigate the risks involved due to resource hints features by blocking the URL using DOM based URL filtering.

6 Conclusion and Future Work

Resource hints create very high impact on the speedy loading of the web page. Today around 90% of the browsers support this feature of HTML5. The attacks occurs due to resource hints are very easy to execute because of the lack of user awareness. We have discussed about the privacy and threats occurred with their implications. The goal of this paper, is to propose a novel approach to mitigate risks involved with resource hints. We use the DOM extractor and URL filtering in order to block malicious web pages. We have implemented this approach to create chrome browser extension named CHOKEGATI and validated it using resource hints options. Our results, shows extension able to block malicious URL where other browser extension were failing to. Hence, CHOKEGATI able to set bench mark in order to mitigate risks involved using resource hints. Increased CPU utilization can be a limitation to the research. Further work, is required to make CHOKEGATI available for other browsers with decrease in CPU utilization and to detect and mitigate other new possible attacks which arises due to its usage.

7 Acknowledgement

We have been supported by National College of Ireland, Dublin to conduct this academic research. I would especially like to thank Mr. Rohan Singla, my mentor for the academic research. As my teacher and mentor, he has guided and encouraged me in every phase of my research. His knowledge and experience was really helpful for me. I would like to thank another person Mrs. Rashmit Singh from Altda Dublin for guiding me with her knowledge which helped my research to meet industry level.

References

Barth, A., Jackson, C. and Mitchell, J. C. (2008). Robust defenses for cross-site request forgery, *Proceedings of the 15th ACM conference on Computer and communications security*, ACM, pp. 75–88.

Chachra, N., Savage, S. and Voelker, G. M. (2015). Affiliate crookies: Characterizing affiliate marketing abuse, *Proceedings of the 2015 Internet Measurement Conference*, ACM, pp. 41–47.

Cohen, E. and Kaplan, H. (2002). Prefetching the means for document transfer: A new approach for reducing web latency, *Computer Networks* **39**(4): 437–455.

De La Ossa, B., Gil, J., Sahuquillo, J. and Pont, A. (2007). Improving web prefetching by making predictions at prefetch, *Next Generation Internet Networks, 3rd EuroNGI Conference on*, IEEE, pp. 21–27.

Deng, Y. and Manoharan, S. (2015). Review and analysis of web prefetching, *Communications, Computers and Signal Processing (PACRIM), 2015 IEEE Pacific Rim Conference on*, IEEE, pp. 40–45.

El Masri, M. and Vlajic, N. (2017). Current state of client-side extensions aimed at protecting against csrf-like attacks, *Communications and Network Security (CNS), 2017 IEEE Conference on*, IEEE, pp. 390–391.

Fan, L., Cao, P., Lin, W. and Jacobson, Q. (1999). Web prefetching between low-bandwidth clients and proxies: potential and performance, *ACM SIGMETRICS Performance Evaluation Review*, Vol. 27, ACM, pp. 178–187.

Gelernter, N., Grinstein, Y. and Herzberg, A. (2015). Cross-site framing attacks, *Proceedings of the 31st Annual Computer Security Applications Conference*, ACM, pp. 161–170.

Gudla, S. K., Sahoo, J. K., Singh, A., Bose, J. and Ahamed, N. (2016). Framework to improve the web application launch time, *Mobile Services (MS), 2016 IEEE International Conference on*, IEEE, pp. 73–78.

Gupta, S., Kaiser, G., Neistadt, D. and Grimm, P. (2003). Dom-based content extraction of html documents, *Proceedings of the 12th international conference on World Wide Web*, ACM, pp. 207–214.

Jourdan, G.-V. (2007). Centralized web proxy services: Security and privacy considerations, *IEEE Internet Computing* **11**(6).

Jovanovic, N., Kirda, E. and Kruegel, C. (2006). Preventing cross site request forgery attacks, *Securecomm and Workshops, 2006*, IEEE, pp. 1–10.

Krishnan, S. and Monrose, F. (2010). Dns prefetching and its privacy implications: when good things go bad, *Proceedings of the 3rd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, USENIX Association, pp. 10–10.

Krishnan, S. and Monrose, F. (2011). An empirical study of the performance, security and privacy implications of domain name prefetching, *Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*, IEEE, pp. 61–72.

Kuzmanovic, A. and Knightly, E. W. (2006). Low-rate tcp-targeted denial of service attacks and counter strategies, *IEEE/ACM Transactions on Networking (TON)* **14**(4): 683–696.

Maes, W., Heyman, T., Desmet, L. and Joosen, W. (2009). Browser protection against cross-site request forgery, *Proceedings of the first ACM workshop on Secure execution of untrusted code*, ACM, pp. 3–10.

Newton, B., Jeffay, K. and Aikat, J. (2013). The continued evolution of web traffic, *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on*, IEEE, pp. 80–89.

Padmanabhan, V. N. and Mogul, J. C. (1996). Using predictive prefetching to improve world wide web latency, *ACM SIGCOMM Computer Communication Review* **26**(3): 22–36.

Vlajic, N., Shi, X., Roumani, H. and Madani, P. (2017a). Resource hints in html5: A new pandora's box of security nightmares, *Proceedings of the 12th International Conference on Availability, Reliability and Security*, ACM, p. 61.

Vlajic, N., Shi, X., Roumani, H. and Madani, P. (2017b). Rethinking the use of resource hints in html5: Is faster always better!?, *Journal of Cyber Security and Mobility* **6**(2): 195–226.

Wang, Z., Lin, F. X., Zhong, L. and Chishtie, M. (2012). How far can client-only solutions go for mobile browser speed?, *Proceedings of the 21st international conference on World Wide Web*, ACM, pp. 31–40.

Yang, Q., Zhang, H. H. and Li, T. (2001). Mining web logs for prediction models in www caching and prefetching, *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 473–478.