基於瀏覽器之分散式阻斷服務攻擊防禦技術研究

劉祉君
國立台灣大學資訊工程研究所
p05922001@ntu.edu.tw

摘要

近幾年來分散式阻斷服務(DDoS)攻擊事件仍然層出不窮，例如 2016 年 Mirai 殭屍網路操控物聯網設備發動約 620Gbit/s 的 DDoS 攻擊及 2017 年台灣券商遭自稱 Armada Collective 駭客集團發動 DDoS 攻擊。除此之外，越來越多新型態 DDoS 攻擊手法出現，例如 2015 年 The Great Cannon 攻擊 Github 的事件。此種攻擊手法利用瀏覽器執行惡意的 JavaScript，讓網頁不斷送出 HTTP 連線請求給受害網站，他可以輕易地發動比一般 DDoS 還要大規模的攻擊。鑑於此種特殊形態的 DDoS 與一般 DDoS 有許多不同之處，在本篇報告中我們深入介紹此種基於瀏覽器式的 DDoS 攻擊(Browser-Based DDoS)，討論針對此種攻擊的防禦方式與一般 DDoS 防禦有何不同，並分析比較各種過去已被提出的防禦方式，最後我們探討防禦此種攻擊的未來的挑戰及瓶頸，以提供後續在此技術領域研究之參考。

關鍵詞：分散式阻斷服務攻擊、基於瀏覽器分散式阻斷攻擊

Mitigating Browser-Based Distributed Denial of Service

Chih Chun Liu
National Taiwan University
p05922001@ntu.edu.tw

Abstract

Distributed Denial of Service (DDoS) attacks continue to threaten the Internet in recent years. For example, the Mirai IoT botnet launched an unprecedented 620Gbit/s DDoS attack in 2016, and a DDoS threat was made to several brokerages in Taiwan by a self-proclaimed group of cybercriminal calling themselves Armada Collective in 2017. Besides, new DDoS attack methods appeared and rendered existing defenses ineffective. A blatant example is the China's Great Cannon first observed in 2015. By injecting malicious JavaScript in the web browsers of unwitting users, this attack caused thousands of HTTP requests per second to victim sites and can easily scale up the attack volume because of its special attack method. Because such browser-based DDoS attacks exhibit several distinct features compared to previous DDoS

attacks, we argue that a systematic investigation of traditional DDoS mitigation techniques against browser-based DDoS attacks is needed. Hence, in this survey paper, we introduce browser-based DDoS attacks and examine potential mitigation techniques against such attacks. The aim of this survey is to gain insights into current research on the defense of this attack by analyzing their effectiveness. This survey also discusses various technical challenges that need to be addressed and provides recommendations for future research directions.

Keywords: Distributed Denial of Service, Browser-based DDoS

壹、前言

本篇報告主要針對特殊型態的攻擊手法「基於瀏覽器之分散式阻斷服務攻擊」進行深入研究。此種攻擊出現在 2015 年，根據 Citizen Lab 在 2015 年的研究報告[5]中指出，中國利用 The Great Firewall 衍伸出的 The Great Cannon 對 Github 上與中國相關的 2 個專案連結發動 DDoS 攻擊[11]，自此以後才較廣為人知。除了 The Great Cannon 之外，同一年 Cloudflare 發現他們的使用者在某個時間點收到大量的 HTTP POST 封包，這些封包與一般使用者的封包並無不同，擁有正常的 HTTP 表頭，經分析後發現這些封包皆源自於一個被插入有問題的 JavaScript 的行動版網頁廣告[12]。

Browser-based DDoS 主要是利用網頁內容插入 JavaScript 程式碼，讓使用者在連線到被植入 JavaScript 的頁面時，會非自主的透過瀏覽器不斷的傳送 HTTP 連線請求給受害網站。被植入的頁面有可能本身就是攻擊者網站將惡意的 JavaScript 放在網頁頁面中，或是遭攻擊者插入惡意 JavaScript，亦或是受到中間人攻擊竄改網頁回傳內容給使用者。

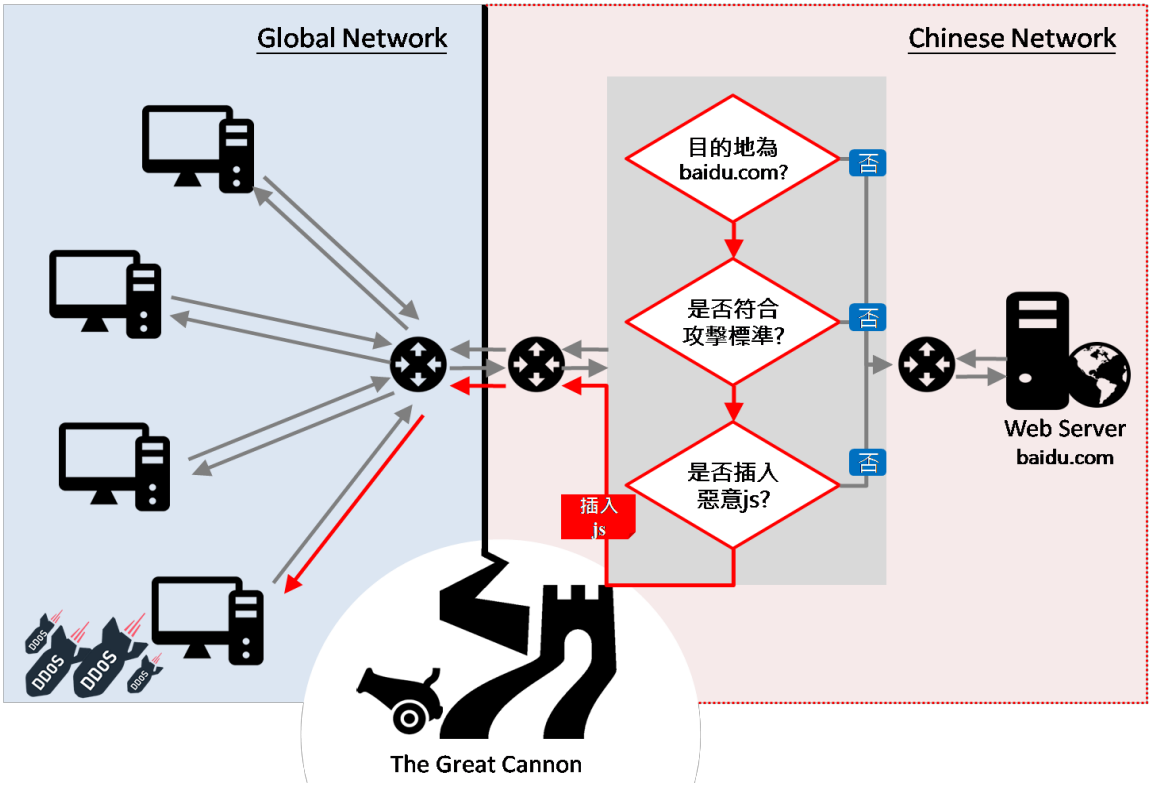
本篇論文針對此種攻擊手法進行深入探討，我們在第二節介紹了 Browser-based DDoS 攻擊的流程，比較其與一般 DDoS 差異之處。第三節介紹相關的研究。第四節以不同面向討論這些已提出的防禦方式。第五節討論以防禦一般 DDoS 的技術來防禦此種攻擊的有效性，第六節提出未來展望與挑戰。最後在第七節總結。

貳、基於瀏覽器之分散式阻斷服務攻擊

本節將深入介紹 2015 年 The Great Cannon 發動的 Browser-based DDoS 攻擊的案例及詳細流程，最後與一般 DDoS 分析並討論相關研究的防禦機制。

2.1 攻擊案例

2015 年 3 月 Github 上面 2 個與中國相關的專案遭到 DDoS 攻擊，攻擊過程是在中國境外的訪客連線到境內的某些網站時，位於 The Great Firewall 附近的伺服器隨機選擇竄改百度統計網站(Baidu analytics 網頁流量分析網站)的程式碼，讓網頁載入惡意的 JavaScript，該 JavaScript 執行後使訪客在不知情的狀況下不斷連向 Github 的專案網站，讓 Github 流量暴增而無法正常使用。(圖一)



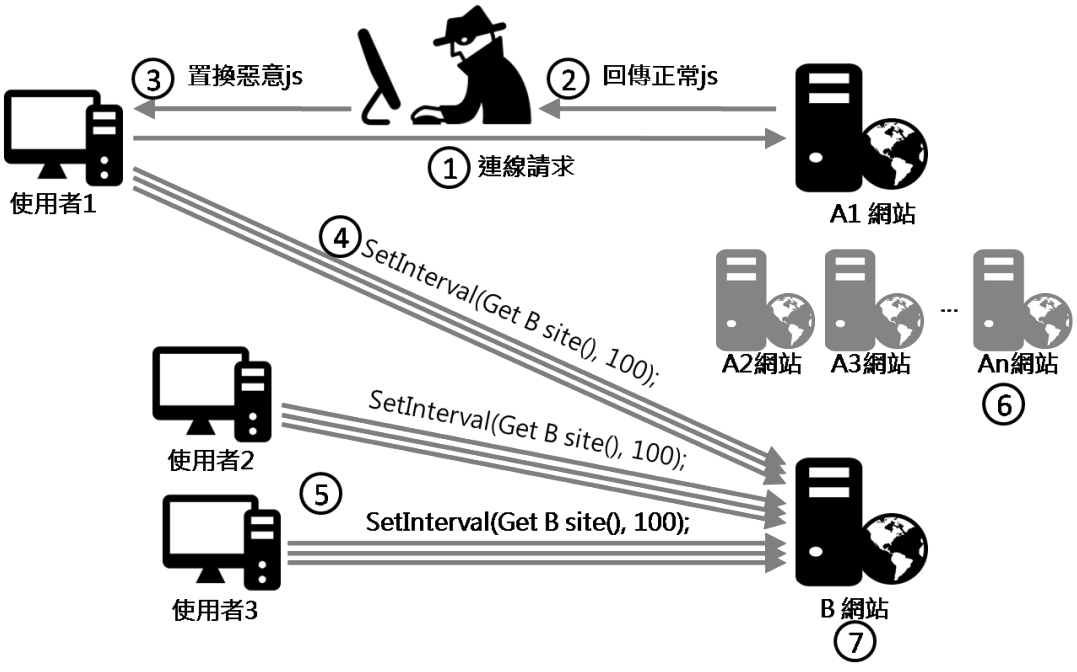
圖一： The Great Cannon 對境外連線請求插入惡意 JavaScript

遭到鎖定攻擊的專案分別是中國網路審查及紐約時報中文網(https://github.com/greatfire/、https://github.com/cn-nytimes/)，經分析後發現惡意程式碼是由中國聯通的伺服器發起的攻擊。在中國百度網站使用的流量統計 JavaScript 功能基本上跟 Google analytics 是一樣的，單純用來協助管理者管理網站的流量，所以很多網站內部都會內嵌此百度流量統計的 JavaScript，但此次攻擊僅針對境外使用者隨機攔截，故雖然有大量的網站使用百度統計，卻只有約 1.75% 的訪客會收到竄改後的惡意 JavaScript，然而這僅有的 1.75%訪客便可造成癱瘓 Github 的有效攻擊，不難推測如果有必要的話，The Great Cannon 甚至還可以提升至 5、60 倍的 DDoS 攻勢。

2.2 攻擊流程

Browser-based DDoS 攻擊流程如下圖二，詳細流程如下列步驟：

(1) 一般使用者 1 連向 A1 網站，向 A1 網站送出 HTTP 連線請求。
(2) A1 網站回傳正常的網頁及正常的 JavaScript。
(3) 在使用者與 A1 網站間的連線受到中間人攻擊或是以其他種攻擊形式，讓原來應回覆的正常 JavaScript 被置換成有惡意連線請求的 JavaScript，此包含惡意程式碼片段的 JavaScript 會不斷向 B 網站送出連線請求。(惡意程式碼範例如下圖三)
(4) 使用者 1 在不知情的情況下不斷向 B 網站送出連線請求。
(5) 同時可能有使用者 2、使用者 3 或更多其他使用者亦接受到遭置換過後的 JavaScript，不斷向 B 網站送出連線請求。
(6) 攻擊者端可能同時可以控制 A1, A2, A3 …An 個網站，讓更多使用者因連向這些網站而向 B 網站發動攻擊。
(7) 因接收來自不同使用者大量的連線請求，B 網站便有可能因此癱瘓。



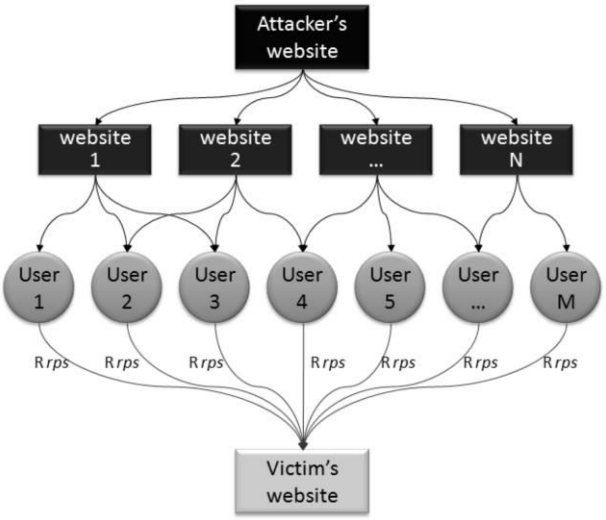
圖二： Browser-based DDoS 攻擊流程

```
1 function HitTarget () {
2   varimage = document . createElement ("img");
3   image .src = " http :// www.B.com/img.jpg";
4   image .id = " image ";
5   document . getElementsByTagName (" head ")[0].
        appendChild( image );
6 }
7 setInterval( HitTarget () , 100) ;
```

圖三： 惡意持續連向外部網站的 JavaScript 範例

2.3 Browser-based DDoS 特徵

此種攻擊的特性是利用連上特定網頁的使用者當作傀儡殭屍(bot)來對目標網站進行 DDoS 攻擊，通常被利用的使用者不知道自己正在對別人發動攻擊，且由於攻擊者可同時利用多個網站並引誘大量使用者來連線，如圖二共有 3 個使用者被插入有問題的 JavaScript 同時連向 B 網站，故相較於一般 DoS 靠單一源頭發動攻擊或是 DDoS 靠多重源頭來同時發動攻擊，Browser-based DDoS 多了一層被利用的使用者(圖四)，在單一源頭發動一樣頻率的攻擊時，Browser-based DDoS 可產生更大量級的攻擊。如下表一，DoS 由單一攻擊者產生 R bps 的攻擊流量，DDoS 由 N 個攻擊者操控的 bot 為源頭各產生 R bps 的攻擊，總量為 N*R bps，而 Browser-based DDoS 攻擊由攻擊者對 N 個網站進行中間人攻擊置換 JavaScript，每個網站經過 M 個人瀏覽，產生 R bps 的流量，總共 N*M*R bps 的攻擊流量。



圖四： Browser-based DDoS 利用網站及使用者增大攻擊量級

表一：Browser-based DDoS 攻擊量級比較

攻擊種類	攻擊源頭	攻擊媒介	攻擊頻率 R	總攻擊量
DoS	1	-	R	R
DDoS	N	-	R	N*R
Browser-based DDoS	N	M	R	N*M*R

除了攻擊量級相對大量之外，Browser-based DDoS 的攻擊架構也由於多一層不知情的使用者而增加防禦的複雜程度，但同時在防禦的節點上也有較多選擇。對於被中間人攻擊的網站，他們需要避免網站被中間人攻擊，對於無辜的使用者，他們需要避免載入有問題的 JavaScript，並防止瀏覽網站向外部存取造成自己成為無辜的幫兇，而受害網站要能辨識此種流量與一般使用者的差異。(表二)

表二： Browser-based DDoS 與一般 DDoS 比較

攻擊種類	數量級	角色	目標	攻擊方式	其他
一般 DDoS	較小 (N*R)	Attacker, Bot, victim	網站 一般使用者	通常不會有中間人攻擊	-
Browser-based DDoS	較大 (N*M*R)	Attacker, compromise website, general user, victim website	網站	中間人攻擊	通常需要 ISP 協助

參、文獻探討

本節討論近年來對於 Browser-based DDoS 攻擊的相關研究[8]。

3.1 Same-Origin Policy (SOP) [13]

瀏覽器防護機制中常被使用的同源政策被用來設計避免網頁請求從 DOM、ajax 或網路存取外部的資源，將資源限制在同一個網域、協定及 port，若來自不同來源的資源便稱之為跨站。然而 SOP 雖然限制了資源存取的來源，卻沒有辦法阻止網頁請求透過

瀏覽器送出，只能避免瀏覽器收到跨站的回應。簡單來說 SOP 的用意是當使用者同時連向有帳戶資訊的網站及惡意網站時，SOP 可以避免惡意網站透過瀏覽器存取到使用者的帳戶資訊。因此對於 Browser-based DDoS 來說，僅在被利用的使用者端使用 SOP，使用者依舊可以持續對受害者送出請求封包。

3.2 Subresource Integrity (SRI) [2]

SRI 在請求 JavaScript 的 URL 後面加上 integrity 屬性算出 hash 值(如下圖五)，瀏覽器拿到網頁內容後，會用 integrity 欄位中指定的 hash 算法計算結果，只有經檢查過符合 hash 值的 script 才可以被執行。SRI 能確保使用者存取的第三方 JavaScript 完整性，但是對於動態來說較沒有彈性，且更新困難，故不是一個能廣泛部署的解決辦法。對 Browser-based DDoS 來說，雖然伺服器端能透過 SRI 讓使用者驗證 JavaScript 完整性，但卻無法防止中間人攻擊直接竄改網頁內容。

```
<script src="https://code.jquery.com/jquery-2.1.4.min.js"
        integrity="sha384-R4/ztc4ZlRqWjqIuvf6RX5yb/v90qNG
                    x6fS48N0tRxiGkqveZETq72KgDVJCp2TC"
        crossorigin="anonymous">
</script>
```

圖五： Subresource Integrity 利用 hash 驗證資源完整性

3.3 Cross-Origin Resource Sharing (CORS) [4]

有別於前述同源政策，CORS 較有彈性可開放部分的跨站來源請求，透過加入新的 HTTP 標頭的方法，提供了網頁伺服器跨網域的存取控制，讓伺服器能夠描述來源資訊以提供瀏覽器讀取。而部分開放指的是只開放那些來源是在同一個母公司或是管理者下的跨站資源，例如 Google 跟 YouTube 就是屬於同一個母公司 Alphabet Inc.便可利用此種方式設立跨站存取權限。但此種方法並不適用於租用外部 CDN(Content Delivery Network)的伺服器，可能會讓 CDN 的存取受限制，也無法有效防禦 Browser-based DDoS 攻擊。

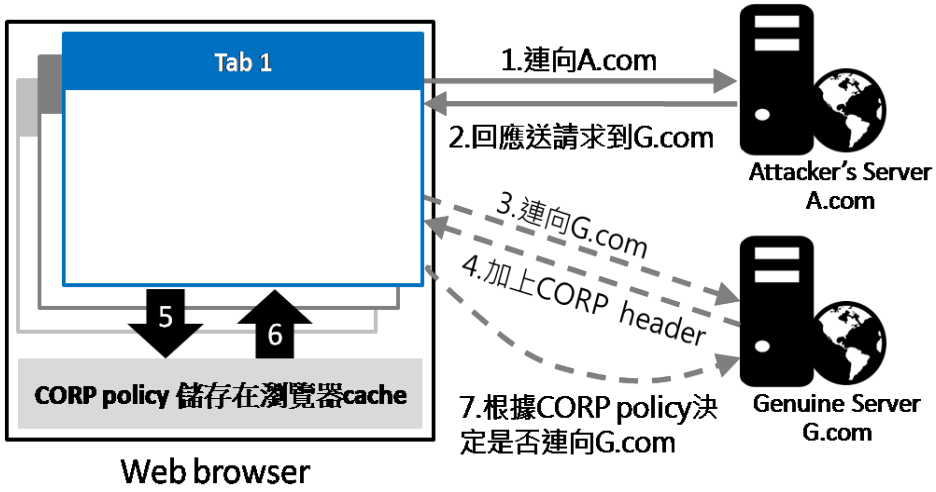
3.4 Server-Side Filters

在受攻擊的伺服器端設立過濾機制，伺服器端辨識出異常流量後封鎖流量源頭，但由於 Browser-based DDoS 攻擊行為與一般使用者幾乎無異，因此單純從伺服器接收

的流量來看，很難有效分辨攻擊者與一般使用者。

3.5 CORP[1]

CORP 於被保護的伺服器所發出的 HTTP 回應標頭中，加上針對不同連線封包的 CORP 規則，這些 CORP 規則可設定流量由何種方式啟動以及來源及是否阻擋/允許 (who what how Access/Deny)，瀏覽器收到回應後會透過暫存器紀錄並利用 plug-in 進行阻擋/允許動作(下圖六)。然而 CORP 除了在被保護的伺服器本身要支援這個服務之外，在使用者端瀏覽器也必須使用 plug-in，才能有效阻擋來自特定啟動方式的流量。除此之外，若伺服器有提供 CDN 的服務，CORP 機制便可能無法有效執行。最後則是 CORP 規則很難準確辨識哪些封包是正常使用者、哪些封包是攻擊連線請求。



圖六： CORP 運作機制

3.6 Local Proxy[7]

因為 Browser-based DDoS 只能透過瀏覽器達到攻擊目的，一種防禦方法是在被利用的使用者前方加上具有偵測功能的本地端 proxy server，當使用者透過瀏覽器與外部網站連線時，所有連線皆須經過 proxy，而 proxy 內建即時檢查模組及決策模組，檢查模組透過 YARA 特徵比對來檢查 script 及 URL，若有惡意網頁行為則將流量封鎖，確保使用者接收到的回應回安全的網頁內容。此種方式對於 Browser-based DDoS 來說對於被利用的使用者可以確保瀏覽網頁時取得的 JavaScript 中沒有惡意行為，但限制是本地端的 proxy 限制是無法使用在行動裝置，再者是 YARA 可能較難偵測出經過較新手

法混淆處理過的 script。

3.7 Behavior-based Detection[9]

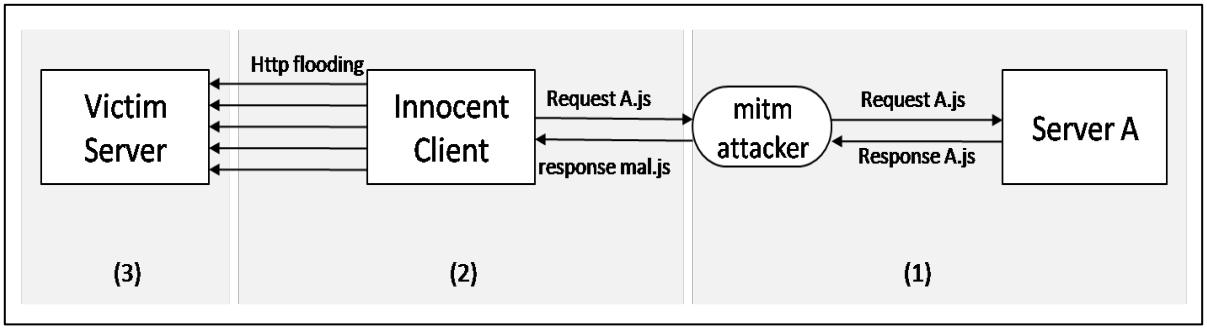
另一種防禦 Browser-based DDoS 的方式，是利用虛擬瀏覽器先行檢查網頁內容。該虛擬瀏覽器為一個安全獨立的分析環境，在實際連向該網頁前先追蹤整個網頁的架構，並將網頁內容分類成 js API, DOM 跟 event，組織成樹狀節點再用 FSM 去檢查定義的惡意行為，相較於特徵比對能更精確地檢測。此種方法透過實際執行網頁內容，確認是否有惡意內容再決定是否要實際連向該網頁，此種方式與 Local Proxy 一樣對於 Browser-based DDoS 來說對於被利用的使用者可以確保瀏覽網頁時取得的 JavaScript 中沒有惡意行為，但其限制是對於有嵌入附加元件的網頁較難執行檢測功能，另外則是效率問題。

肆、Browser-based DDoS 防禦機制比較

4.1 防禦機制建置面向

前述文獻探討中提到 Browser-based DDoS 攻擊現已有多種現成的防禦方式，但這些防禦方式仍舊各自有其限制，本節比較各種方式的優缺點。

首先，在 Browser-based DDoS 中，前面段落提到此種攻擊與一般 DDoS 多了一個不知情的使用者被當成 bot，因此實際可以參與防禦的節點也更有彈性，根據上述各種防禦方式及其建置的防禦位置(圖七)，我們討論這些建置在不同節點上的防禦機制以及建置在這些節點上的先天性限制。



圖七： Browser-based DDoS 節點

以建置在被利用的使用者的防禦機制來說，我們認為使用者多半較在乎自己是否遭受到攻擊而不是在意自己是不是被利用，因此為了保護其他受害者端而要求使用者在自己的瀏覽器或電腦安裝軟體或是 plug-in 並不是一個最直接而有成效的做法；而建置在伺服器端的防禦機制，其一是可以確定連向它的使用者存取的就是伺服器想提供的內容，而非經竄改過的惡意內容，其次是避免使用者因為瀏覽自己的網站而遭受攻擊或是被當成 bot 向他人發動攻擊；最後，建置在受害者端的防禦機制雖不能在接近攻擊源頭的地方就阻止攻擊，但能夠最直接保護受害者端。上述三種防禦機制的位置，建置在伺服器端或是被利用的使用者端或許更能有效阻止 JavaScript 被竄改，但是就佈署層面來說，直接保護受害者端的防禦方法可能才是較實際的作法。

根據相關文獻中提到的防禦機制不同的佈署位置，我們整理出下表三。

表三：Browser-based DDoS 防禦機制建置位置比較

Browser-based DDoS 防禦機制	Innocent client	Server A	Victim Server
Same-Origin Policy	**V**		
Subresource Integrity	**V**	**V**	
CORS	**V**		
Server Side Filter			**V**
CORP	**V**		**V**
Local Proxy	**V**		
Behavior-based Detection	**V**		

4.2 防禦機制執行面向

本段落討論這些防禦機制在實際執行時可能碰到的問題，包含了一般 DDoS 防禦常見的限制，例如建置困難度或是準確度與效率的取捨，我們整理出下表四。

效率上來說，須檢查的資訊越少則效率越好，如(1)(2)(3)(4)(5)僅需檢查少量資訊故對效率影響不大，而(6)(7)因需檢查的資訊較多或是需要額外導向外部的檢查步驟，影響效率相對較大。

建置困難度以需要配合的節點越多則越困難，(1)(3)(4)(6)(7)因只需單一節點即可完成建置故為簡單，而(2)牽涉到伺服器及使用者兩節點，(5)則是透過使用者及被害者兩節點的合作來達成防禦。

準確度以可以分辨一般使用者與攻擊者為前提之下，能夠精確辨識出攻擊流量者如(6)(7)針對網頁行為及封包特徵進行辨識，而準確度較差者如(2)因無法阻止中間人攻擊且無法使用在動態 JavaScript 而受限，(4)(5)因可以檢查的內容有限，在很多情況下可能會有誤判的狀況發生，而(1)(3)則是僅能建立簡單的過濾條件所以準確度與其他方式相比較差。

規則設定方面，越複雜的條件設定越困難，(1)(2)(3)因能設定的條件較少，故困難度相對低，而其他則是相對複雜。

表四：Browser-based DDoS 防禦機制執行面向比較

Browser-based DDoS 防禦機制	**效率**	**建置困難度**	**規則設定**	**準確度**
(1) Same-Origin Policy(SOP)	**快**	**簡單**	**簡單**	低
(2) Subresource Integrity(SRI)	**快**	困難	**簡單**	**中等**
(3) CORS	**快**	**簡單**	**簡單**	低
(4) Server Side Filter	**快**	**簡單**	**簡單**	低
(5) CORP	**快**	困難	困難	**中等**
(6) Local Proxy	慢	**簡單**	困難	**高**
(7) Behavior-based Detection	慢	**簡單**	困難	**高**

4.3 防禦機制改進方法

上面段落我們討論了防禦機制在不同面向的比較，針對表四中準確度較好的 4 種方法，我們提出一些改進的想法。

Subresource Integrity 因受限於動態 JavaScript 及中間人攻擊，我們認為若能針對動態函式進行檢查或是提供網頁本身的完整度確認機制或許能改進只檢查外部 JavaScript 的完整度來的更有效。

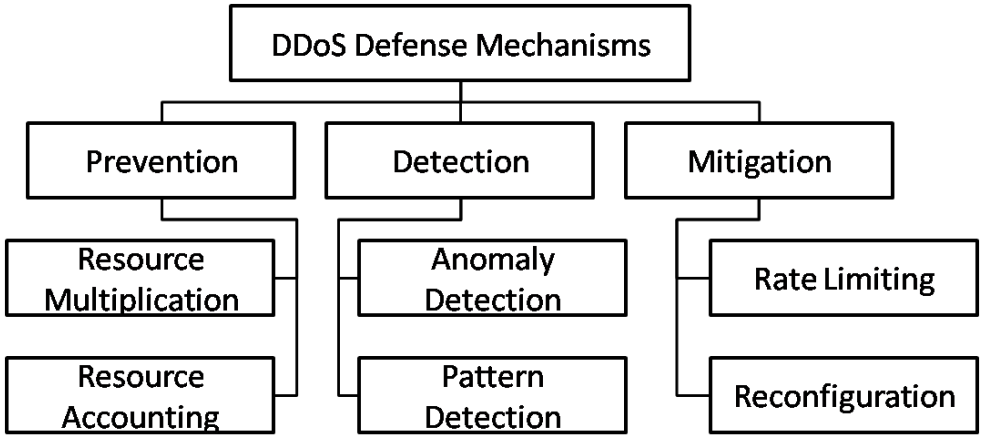
CORP 因檢查的內容僅針對 HTTP 連線請求時的 who / how / what (誰要取得資源 / 以什麼方式取得 / 取得什麼資源)去決定能不能存取，如果能夠再加上更多條件設定或許能提升準確度，此外因為這種方式需要兩個以上節點配合，若能利用單一節點完成

一樣的機制也是可以改進的地方。

Local Proxy 及 Behavior-based Detection 主要缺點是效率差，為解決此問題，Local Proxy 可能可以將 proxy 建置在 CDN 上面，而 Behavior-based DDoS 或許可以瀏覽器擴充功能代替虛擬瀏覽器的檢查作業以提升效率。

伍、一般 DDoS 防禦機制應用

[6]以各種不同方式來分類探討 DDoS 防禦機制，其中依照防禦作業的層級分成預防措施(preventive)及應對策略(reactive)，而應對策略又分成偵測攻擊(detection)及減緩傷害(mitigation)，依照此三種分類，我們挑出較具代表性幾種防禦方式，這些防禦方式皆為建置在伺服器端較普遍使用的防禦 DDoS 的方法(圖八)，下面段落依序進行討論。



圖八： DDoS 防禦方式分類

5.1 攻擊預防的方式中依照類型不同分述如下：

5.1.1 資源取用權限(Resource Accounting)：

這樣的機制限制了非合法使用者的資源取用，通常需配合對使用者進行身分稽核的動作，這裡的使用者範圍包含一個人、一個程式、一個 IP 位址甚至是一群 IP 位址，雖然對於尚未取得合法身分者可能有麻煩的步驟需處理，但通常能保證合法使用者的使用權限不會受影響。然而使用在 Browser-based DDoS 上若僅針對身份進行判定，而未能確認封包是由使用者自主發起的還是由 JavaScript 啟動的，很難有效防禦攻擊。

5.1.2 增加系統資源(Resource Multiplication)：

此種預防機制是增加本身設備資源避免被大量流量阻塞，或是保證在系統壞掉後仍有負載平衡的機制提供服務。缺點是在花費上的開銷較龐大，且對於 Browser-based DDoS 此種量級較大之攻擊方式，僅以大量的頻寬或資源來預防似乎不是一個好選擇。

5.2 攻擊偵測的方式中依照類型不同分述如下：

5.2.1 異常行為偵測(Anomaly Detection)：

根據流量或連線行為進行正常使用者及攻擊者的辨別，透過行為特徵篩濾或即時流量分析偵測攻擊行為。好處是流量或行為特徵明顯的攻擊很容易偵測，限制是無法定義準確的連線檢查，可能會有較高的誤判率。從 Browser-based DDoS 受害端角度來看，其接收到的封包會是相同不斷重複的連線請求，從連線請求數量異常可以很容易判斷遭受攻擊，但若攻擊使用大量瀏覽器每個瀏覽器連線請求量很小時，將很難準確辨識異常。

5.2.2 封包特徵檢查(Pattern Detection)：

此種機制擁有已知攻擊特徵的黑名單或資料庫，針對流經的封包進行特徵篩濾動作。優點是針對能掌握的攻擊特徵皆能有效預防，但限制是無法檢查出不在清單中的攻擊，而對 Browser-based DDoS 的受害者來說，除非是已知某種插入 JavaScript 的攻擊傳送的封包內容都是固定的字串或特徵，否則很難單純從封包內容中判斷出哪些是攻擊封包哪些是一般使用者。

5.3 攻擊減緩的方式中依照類型不同分述如下：

5.3.1 流量限制(Rate-Limiting)：

此種減緩機制在伺服器前的路由器上設立每個來源可以使用的流量限制臨界值，如果每秒要進到伺服器的 HTTP 連線請求大於臨界值，路由器就會封鎖同樣來源的封包。然而這個方法的限制是攻擊者可以用測試的方式測出臨界值，再利用不同的來源，可能是殭屍網路或是偽造 IP 來源，讓不同來源的攻擊皆低於臨界值，同步發起 DDoS 攻擊造成伺服器癱瘓，因此對於 Browser-based DDoS 來說也有其缺點。

5.3.2 資源重新配置(Reconfiguration)：

伺服器遭受攻擊後對系統中的網路拓樸重新配置，或是將受攻擊的伺服器隔離，讓攻擊者找不到目標或是原先的路徑。對於 Browser-based DDoS 中的受害端來說，由於是屬於HTTP的攻擊類型，若要重新配置網頁路徑，亦可能會讓合法使用者無法連線，此部分或許可結合身份稽核制度，在受攻擊時能讓正常使用者進行身份認定並轉往有效連結。

綜整如下表五：

表五：Browser-based DDoS 防禦比較

防禦方式		優點	缺點
預防	**Resource Accounting**	合法使用者不受影響	權限給予方式困難
	Resource Multiplication	保證服務不受影響	建置成本高
偵測	**Anomaly Detection**	容易辨識流量暴增的攻擊	難以防禦大規模分散的DDoS攻擊
	Pattern Detection	針對特徵明顯的封包可以準確偵測	無法單純透過特徵檢查Browser-based DDoS
減緩	**Rate-Limiting**	伺服器較不容易被流量阻斷	難以防禦大規模分散的DDoS攻擊
	Reconfiguration	保證服務不受影響	建置成本高 可能會暫時影響正常使用者

5.4 Cloud-based 防禦機制

除了上面提及較具代表性的幾種防禦方式外，近年來許多研究提出以 Cloud-based 的方式來防禦 DDoS，如[3]中整理了數篇近年以 Cloud-based 建置 DDoS 防禦機制的研究。因應DDoS攻擊的流量不斷增大、難以大規模佈署及在越接近源頭越容易阻止等特性，以雲端服務來提供防禦機制有其顯著的優點，尤其是針對 Browser-based 這種量級更大的攻擊。另一方面來說，雖然 Cloud-based 的建置方式為目前較有效的減緩傷害防禦機制，其仍然無法解決偵測攻擊防禦機制上的缺點。除此之外，Cloud-based 比起其他方法增添不少新的安全因素上的挑戰，例如隱私資訊的可見度、資源共享造成的安

全問題或是雲端服務可能變成攻擊目標等問題。

陸、 未來展望與挑戰

前面段落我們討論了 Browser-based DDoS 現有的防禦方式，並與一般 DDoS 的防禦方式做分析比較。對於 Browser-based DDoS 來說，我們整理出幾個現有防禦的限制性，包含下面所列前 4 點亦是一般 DDoS 的問題，第 5 點是相較於一般的 DDoS 在 Browser-based DDoS 中會更為困難的挑戰，6、7 點則為 Browser-based DDoS 防禦的限制：

(1) 很難大規模佈署：對於 DDoS 這種分散式的來源攻擊，往往牽扯網路中許多節點，因此許多相關研究中的防禦機制需要 ISP 的協助，例如限制流量或是尋找攻擊源頭，否則難以實現有效的防禦。

(2) 經濟效益考量：比起發起一次 DDoS 攻擊，防禦 DDoS 所需成本相對大，且若需要建置一套完善的防禦系統，所需耗費的經費更為龐大，一些中小企業難以負荷這些經費支出，基於成本考量而選擇較簡單的資安系統，畢竟問題沒有發生時，一般人較不願意去投入這樣的金錢。反言之，若要開發一套良好的 DDoS 防禦系統，為了提高實用性及普遍性，成本考量也是重要因素之一。

(3) 缺少細節資訊：精確的防禦來自於深入的了解，但 DDoS 攻擊種類繁多，且不斷出現各種新型手法，若無所有攻擊的防禦細節資訊，包含連線行為、封包特徵、流量變化等資訊，將很難有效抵擋所有攻擊。

(4) 難以有效測試：因大規模的 DDoS 攻擊環境難以實際建置，尤其許多研究又需涉及到 ISP 協助，因此 DDoS 領域大多數研究無法透過實際測試來驗證有效性，而常常透過現成的網路拓樸或攻擊資料庫(例如 CAIDA[10]進行驗證，然而這樣的方式可能造成與實際現況的落差。

(5) 難以分辨正常使用行為與攻擊行為：因為此種攻擊方式與一般瀏覽網頁行為幾乎一樣，包含來源及封包特徵，尤其若能達到單一源頭量少但分散規模大的攻擊時，更無法單純由流量來辨認。其中較有效的方法可能是透過 HTTP 參照位址欄位來統計這些異常流量的啟動來源。

(6) 過度依賴被利用的使用者：因 browser-based DDoS 此種攻擊方式較為特殊，若欲建置在受害者伺服器端其實很難跳脫出現有 DDoS 防禦的限制，但若要避免使用者端成為發動攻擊者，往往需要使用者端的配合，但使用者端並非直接受害者，要求使用者提別人做防禦而加裝一些軟體，可能不是一個非常好的辦法。

(7) 難以追蹤源頭：比起一般因感染遭受控制的殭屍網路通常可透過殭屍電腦去追蹤回攻擊者或中繼站，然而 Browser-based DDoS 透過使用者最多追蹤到受攻擊者操控的網站，除了網站可能非常分散難以一一追蹤之外，若該網站是受到中間人攻

擊的話，便更難找到攻擊者。

綜合整理以上，可以推斷出若要有效防禦 Browser-based DDoS 攻擊，我們應關注在降低建置成本、減少需要主動協助配合的節點、利用 HTTP 標頭分析辨識流量以及需將重點放在受害伺服器的防護上而非被利用的使用者。

柒、結論

本篇論文中我們介紹了 2015 年興起的特殊型態 Browser-based DDoS 攻擊，分析此種攻擊與一般 DDoS 的不同之處。再者探討現有防禦機制的優缺點，首先，我們整理部份針對 Browser-based DDoS 防禦方式的優缺點，再來討論針對一般 DDoS 的防禦直接套用在此種攻擊上面可能會有的限制以及優點，最後我們整理出以現階段所提的防禦機制仍需解決的研究挑戰，希望透過本篇研究的整理與分析意見，提供未來在 Browser-based DDoS 防禦上可以繼續研究的方向。

參考資料

[1] A. Agrawall, K. Chaitanya, A. K. Agrawal and V. Choppella, “Mitigating Browser-based DDoS Attacks using CORP,” *Proceedings of the 10th Innovations in Software Engineering Conference,* ACM*,* 2017.

[2] F Braun, D. Akhawe, J. Weinberger and M. West, “Subresource integrity,” *W3C working draft*, 2014.

[3] B. B. Gupta and O. P. Badve, “Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a cloud computing environment,” *Neural Computing and Applications,* 28(12), 3655-3682.

[4] A. V. Kesteren, “Cross-Origin Resource Sharing,” *W3C Working Draft, Version WD-cors-20100727,* 2010.

[5] B. Marczak, N. Weaver, J. Dalek, R. Ensafi, D. Fifield, S. McKune, A. Rey, J. S. Railton, R. Deibert and V. Paxson, “An analysis of china’s “great cannon”,” *FOCI. USENIX,* 2015.

[6] J. Mirkovic and R. Peter, “A taxonomy of DDoS attack and DDoS defense mechanisms,” *ACM SIGCOMM Computer Communication Review* 34.2, 2004. 39-53.

[7] S. Oh, H. Bae, S. Yoon, H. Kim and Y. Cha, “Malicious Script Blocking Detection Technology Using a Local Proxy,” *2016 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, IEEE, 2016.

[8] G. Pellegrino, C. Rossow, F. J. Ryba, T. C. Schmidt and M. Wählisch, "Cashing Out the Great Cannon? On Browser-Based DDoS Attacks and Economics," *WOO*, 2015.

[9] S. Yoon, H. L. Choo, H. Bae and H. Kim, "Behavior-Based Detection for Malicious Script-Based Attack," *International Conference on Computer Science and its Applications*, Springer Singapore, 2016.

[10] http://www.caida.org/home/

[11] http://www.theregister.co.uk/2015/03/27/github_under_fire_from_weaponized_great_firewall/

[12] https://blog.cloudflare.com/mobile-ad-networks-as-ddos-vectors/

[13] https://code.google.com/archive/p/browsersec/wikis/Part2.wiki#Same-origin_policy