使用 URL Routing 與 Web Storage 防禦跨站冒名請求攻擊

劉書宇　　吳其哲　　丁建文
國立高雄應用科技大學 資訊管理研究所
{f126882231,jerryclass, jenwen.ding}@gmail.com

摘要

近年來網際網路資訊的發展與行動設備的普及，生活中越來越多的資訊與活動都在網路上流通與進行，以網站為基礎的網站應用程式(Web Application)更是蓬勃發展，但由於 HTTP 有著無狀態(stateless)的特性，因此易造成伺服器與用戶端(Broswer)之間的關聯薄弱與驗證不足等，使得網站經常為駭客攻擊的目標，其中的跨站冒名請求攻擊(Cross site request forgery)為最常見的攻擊方式之一，駭客將惡意程式碼嵌入攻擊者或被攻擊者的伺服器網頁中，使得瀏覽者遭到如 session riding 與 session hijacking 等冒用使用者身分的攻擊。

在本論文中將深入討論當前和未來網路的安全性，並針對跨站冒名請求攻擊提出了一套針對使用者的網址變化機制，研究中利用 HTML5 新制定的 Web Storage 來儲存多組使用者登入成功時所取得的多組 UUID API Key，並在使用者操時時將網址與 UUID API Key 結合後作為請求網址發送至伺服端，當伺服端接收到後便可透過 URL Routing 之機制將鍵值取出並與 Server 端的資料做比對，以此達到使用者 Cookie 與 UUID API Key 的雙重認證機制。

關鍵詞：跨站冒名請求、網址路由、通用唯一識別碼、HTML5、Web Storage

Abstract

In recent years, the development of Internet information and the popularity of mobile devices, more and more information and activities are working on the Internet and besides, web applications are booming in our life. However, HTTP is a stateless protocol. Therefore, the linking between the server and the client verifying is weak and inadequate to make the hackers target these sites. One of the most common attacks is Cross Site Request Forgery; Hackers would embed malicious code into the attacked person's web server. Then, viewers are attacked such session riding and session hijacking and other fraudulent attack by using of user identity.

In the research, we would discuss deeply the current and future network security and propose a set of URL change mechanism focusing on the Cross Site Request Forgery attack. Using newly Web Storage to store multiple UUID API Keys from many sets of successful user-login is based on HTML 5 in the research. Besides, while users operate to login into the sites, combining the site and UUID API Key is as a request URL which is sent to the server.

When the server receives the URL, it can compare the data from the server through URL Routing mechanism to achieve the two-factor authentication mechanism based on user cookie and UUID API Key.

Keywords: Cross site request forgery, URL Routing, UUID, HTML5, Web Storage

1. 前言

根據網路安全專家研究，跨站冒名請求攻擊(Cross site request forgery)為過去十年中最為嚴重和常見的威脅之一，並分被標註在 CWE(Common Weakness Enumeration)清單與 OWASP(Open Web Application Security Project)清單中(圖 1)，且多個主要網站包含 Facebook、Twitter、eBay、Google 和 McAfee 皆已成為跨站冒名請求攻擊的目標。

跨站冒名請求攻擊是屬於許多網頁應用程式所固有的弱點，主要發生原因為網站應用程式對於使用者的請求缺乏檢查或認證不足，且在任何不同的語言如 PHP、JSP 和 ASP.NET 等都存在著，攻擊者通過這些漏洞注入惡意代碼，而導致意想不到的惡意命令從客戶端瀏覽器執行[1,2]。

雖然許多研究人員已經提出了多種跨站冒名請求攻擊(Cross site request forgery)的緩解機制，包含請求方法的更改、使用者來源的認證與代理伺服器的使用等，但由於網頁應用程式開發者缺乏對跨站冒名請求攻擊的理解與技術上的侷限，跨站冒名請求攻擊的漏洞依然廣泛的存在。

本研究提出的安全機制主要以 HTML5 的 Web Storage 與伺服端網址路由(URL Routing)的技術為基礎，利用 HTML5 Web Storage 相較於 Cookie 擁有更高安全性與更大儲存空間的特性來儲存多組獨一無二的 UUID API Key，並與請求網址(URL)做結合以產生使用者專用的請求連結位置，以達到每位使用者的網址獨特性，最後再透過 Server 端的網址路由(URL Routing)技術取出 UUID API Key，達到 UUID API Key 與 Cookie 的雙重認證，使得使用者的網路環境更加安全。

OWASP TOP 10 – 2013
A1 – Injection
A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References
A5 – Security Misconfiguration
A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control
A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards

圖 1 OWASP 十大網路威脅[3]

2. 文獻探討

2.1 Cookie

由於 HTTP 是基於請求與回應的無狀態協定，因此當客戶端每一次的請求與伺服端的回應結束之後伺服端便會遺忘曾與客戶端間有過的任何互動，若沒有 Cookie 與 Session 的概念存在的話客戶端每次連線到伺服端都會被當成是一個新的連接，因此 Cookie 與 Session 的發明便是為了保持客戶端與伺服端間的連續性與狀態儲存[4，5]。

如圖 2 所示，當客戶端造訪至一個擁有 Cookie 機制之網站時，伺服端便會透過 HTTP 標頭與瀏覽器將 Cookie 儲存至客戶端的硬碟或記憶體之中，接著在客戶端之後的 HTTP 請求中，伺服端便會透過瀏覽器送出的 HTTP 標頭檢索客戶端擁有之 Cookie 並取得相關訊息以進行後續之服務，而不要求相同之訊息。

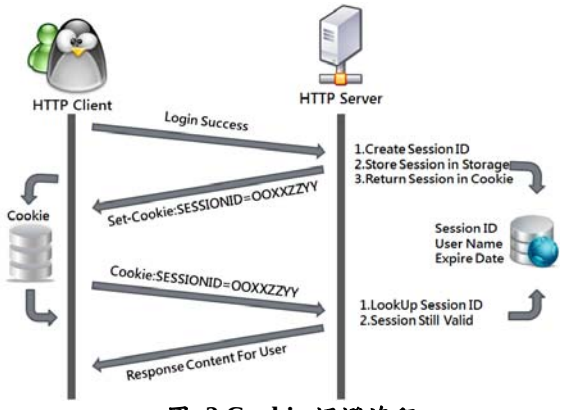

圖 2 Cookie 認證流程

2.1.1 Cookie 威脅

Cookie 的威脅主要有三種類型：網路威脅、系統威脅與竊取威脅，且三種威脅都相當容易實現[6]。首先網路威脅是由於大部分之 Cookie 是以明文的形式在網路上傳輸的，因此 Cookie 是相當容易遭到監聽與解讀，雖然 SSL[7]的出現能增加 Cookie 之安全性，但此安全性也僅限於在網路上傳輸時。其次系統威脅是由於 Cookie 是儲存在客戶端電腦之硬碟或記憶體中，因此夠在系統能夠輕易地修改內容或將其複製至其他電腦中，這使得攻擊者將能偽造訊息或冒充他人身分。最後竊取威脅為若攻擊者在網站中嵌入惡意之請求，該攻擊者便可透過此攻擊竊取使用者之 Cookie 以達到身分偽造之目的。

2.2 跨站冒名請求攻擊

跨站冒名請求攻擊(Cross site request forgery)是相較於跨站腳本攻擊(Cross site scripting)是屬於比較新的網路安全議題[8]，跨站冒名請求攻擊(Cross site request forgery)透過網站對客戶端下達任意有漏洞的 HTTP 請求以達到攻擊的目的，若此時受害者已通過該 HTTP 請求之認證，跨站冒名請求攻擊(Cross site request forgery)便能繞過該 HTTP 請求之身分認證機制並進行攻擊，根據攻擊的方式，將可能會造成受害者在不知情的情況下發布消息或郵件，而更嚴重時甚至會更改受害者之資料如姓名或密碼等，但不幸的是相較於跨站腳本攻擊(Cross site scripting)與隱碼注入(SQL Injection)攻擊，跨站冒名請求攻擊(Cross site request forgery)之防禦是網頁開發者中鮮少會去注意到的[9]。

2.2.1 跨站冒名請求攻擊方式

跨站冒名請求攻擊(Cross site request forgery)主要是利用伺服端透過 Cookie 儲存在客戶端的 Session ID，其原因為瀏覽器會自動在 HTTP 請求時加入 Cookie，因此在不論是在超連結的點擊或是網頁的重新導向情況下，該 HTTP 請求皆會正確的被觸發，相對的若遭跨站冒名請求攻擊的客戶端不包含驗證所需的 Cookie，則該跨站冒名請求攻擊(Cross site request forgery)便會失敗。

其中最常見的跨站冒名請求(Cross site request forgery)攻擊方式是利用 HTML 中如<Script>、<Image>或<iframe>等有能力發出 HTTP 請求之標籤，如圖 3 所示跨站冒名請求攻擊(Cross site request forgery)強迫使用者的瀏覽器自動對於該目標網址發出請求，導致該頁面的瀏覽者受到資料竄改(Web Content Manipulation)等攻擊，其他典型的跨站冒名請求攻擊(Cross site request forgery)方式還有透過超連結網址(URL)、表單(Form)、JavaScript、VBScript 和 Flash 等[1]。

與跨站腳本攻擊(Cross site scripting)相較之下，跨站冒名請求攻擊(Cross site request forgery)攻擊的目的不是為了取的使用者的資料，而是利用伺服端無法分辨用戶的請求真假特性，欺騙伺服器而導致使用者的行為與權限遭到濫用。

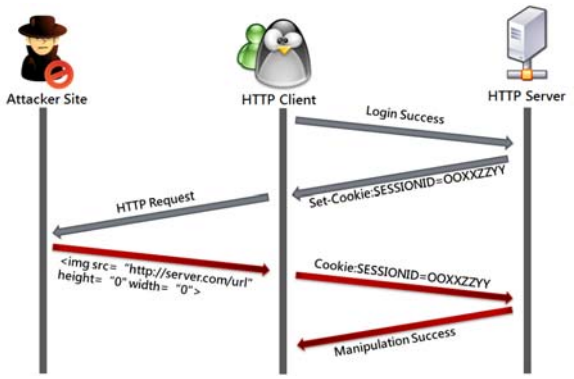

圖 3 跨站腳本攻擊流程

2.2.2 現存防禦跨站冒名請求攻擊機制

防禦跨站冒名請求攻擊(Cross site request forgery)最為簡單的方式便是將 HTTP 請求的方法由 GET 改為 POST，然而這種防禦的方式經實驗證明，只是提高了跨站冒名請求攻擊(Cross site request forgery)的攻擊門檻，如避免一些特定的攻擊方式如利用<image>、<iframe>標籤等，此外若想將網站的 HTTP 請求方式完全改為 POST 是不可能的，因為這將導致網站的程序複雜與網頁開發者的難以實現[9]。

另外防禦跨站冒名請求攻擊(Cross site request forgery)的方式還有透過驗證瀏覽器所產生的 HTTP 標頭 Referer，Referer 的內容為客戶端在發出此 HTTP 請求時瀏覽器所在的頁面網址，因此透過此 Referer 伺服端便可進一步判斷是否要接受來自此 Referer 的 HTTP 請求，以銀行的交易網站為例，若該 Referer 的網址不再伺服端接受的白名單內，則會拒絕此 HTTP 請求並中斷交易，但不幸的是由於此 Referer 是很容易偽造並有可能會洩漏個人隱私資料的，因此大部分的瀏覽器為了不洩漏該使用者的隱私資料，當客戶端發出 HTTP 請求時大多會以空值送出，導致開發者若採用此跨站冒名請求攻擊(Cross site request forgery)防禦機制，勢必會造成客戶端瀏覽器的限制，進而影響使用者的操作意願[11,12]。

而金鑰交換為迄今最常使用的跨站冒名請求攻擊(Cross site request forgery)解決方案，其為透過在伺服端與客戶端兩地儲存由應用程序產生不容易遭到猜測的共同金鑰代碼，並在客戶端 HTTP 請求發生時將此代碼包含在送出的變數中，以供伺服端比對是否與客戶端的金鑰代碼相同，若金鑰相同則執行正確的 HTTP 回應。而此方法的缺點為開發者所需投入的開發成本較高，因為採用金鑰交換機制須建立龐大且複雜的金鑰管理機制，且需修改每個採用金鑰交換機制的 HTTP 處理器，而若開發者還採用更加複雜的一次性密碼(One Time Password)機制，除了開發成本的更加提高外也會大幅將低客戶端操作之彈性[10]。

最後防禦跨站冒名請求攻擊(Cross site request forgery)的方式還有透過代理伺服器(Proxy)，此方式的防禦方法為透過另外一台伺服器去擴展原有伺服端的功能，進而達到伺服端對客戶端進一步的掌握，以阻絕所有開發者認為不正當或惡意的操作可能，而此減輕機制最為重要的條件為開發者需要建立代理伺服器與伺服端間的完整的關聯，因此相較於前幾項跨站冒名請求攻擊(Cross site request forgery)的防範機制，使用代理伺服器勢必也將造成開發者投入的開發成本更為增加[10]。

2.3 HTML5 Web Storage

HTML5 中提供了一個類似於 Cookie 的 HTML5 Web Storage 儲存機制，提供伺服器端能夠在客戶端儲存鍵值與變數，並且提供了簡單的函數來達成新增、修改、刪除和查詢之功能。然而不同於 Cookie 的是 HTML5 Web Storage 並不會透過 HTTP 請求而自動送出，如圖 4 所示 HTML5 Web Storage 所有的控制與儲存皆是透過 Javascript 腳本做操作。另外相較於 Cookie，最大的優點就是不需要持續地發送至伺服端做驗證。

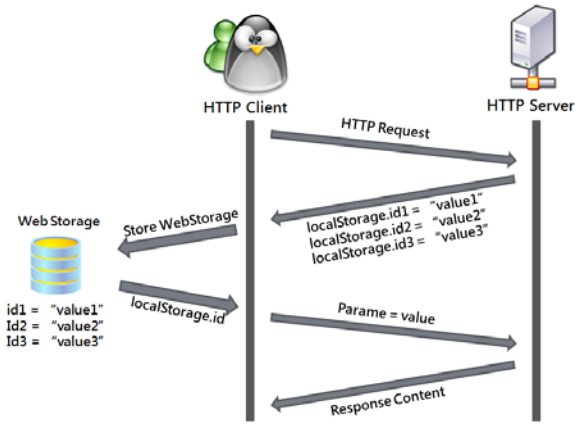

圖 4 HTML5 Web Storage 流程

2.3.1 HTML5 Web Storage 隱私

由於持久性的 Cookie 造成有關隱私上的相關問題，因此 HTML5 Web Storage 規範提供了相似於 Cookie 的替代品 Web Storage 使得客戶端能對資料能有更佳的控制。以網頁中呈現的背景顏色為例，若在沒有使用 HTML5 Web Storage 的情況下，伺服端需在資料庫中或 Cookie 中儲存使用者之偏好之背景顏色，並且在 HTTP 請求中不斷的送出與驗證該客戶端背景顏色之資料，此時便會有洩漏資料之

問題，但若改為使用 HTML5 Web Storage 來存取該背景顏色資料時，便能將修改與驗證之機制都留在客戶端處理，同時也能減少資料走漏之風險 [13]。

2.4 網址路由

網址路由在伺服端中主要有兩個目的，分別是比對並處理客戶端透過瀏覽器送出的 HTTP 請求與回應適當的回應給瀏覽器，以 MVC 架構為例，瀏覽器會將 HTTP 請求轉換成封包並發出 HTTP 請求到伺服端上，此時伺服端之網址路由便會取得該請求之路由值，並由上而下比對網址路由表中所有路由規則來選擇最適合之路由規則後交由網址路由模組判定此請求要交由哪個 HTTP 處理器處理，如伺服端判定是由 MVC 處理器來處理，那就會讓 HTTP 請求進入 MVC 的執行週期，最後再提供適當的回應給客戶端[14]。

此外網址路由中也可以自行定義路由參數與路由值，如圖 5 所示，開發者在路由表中定義了 Controller、Action 與 Parame 之路由參數，但由於 Parame 在路由表中非屬於 MVC 處理器的路由參數之一，因此 Parame 所屬的路由值"123" 就會自動被轉換成查詢參數的一部分。

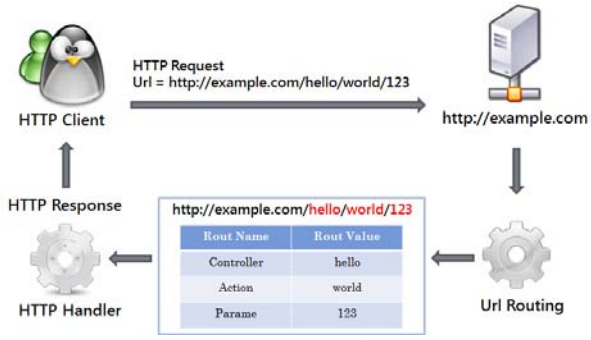

圖 5 網址路由流程

2.5 通用唯一識別碼 UUID

通用唯一標識別碼 UUID(Universally Unique Identifier)也被稱為全局唯一識別碼 GUID(Globally Unique Identifier)，是一種在沒有中央系統的情況下還能保證唯一性之規範。使用通用唯一標識別碼 UUID 主要的原因之一是希望在沒有權利管理其他系統的情況下，能產生獨立之鍵值並用於各種用途如做交易之鍵值等，如此一來每個人都可以不與其它人衝突，進而解決名稱或 ID 重複之問題[15]。

一組通用唯一標識別碼 UUID，是由一串 16 位元組(128 位元)的 16 進位數字所構成，是故通用唯一標識別碼 UUID 的總數為 2 的 128 次方，也就是說若每奈秒產生 1 兆個 UUID，則要花 100 億年才會將所有的 UUID 用完，其中通用唯一標識別碼 UUID 的標準型式包含 32 個16 進位數字，以連號分為五段，形式為 8-4-4-4-12 的 36 個字元如 08dac2f1-e4c4-4cea-a202-7e480affe859，而其他重要的應用，則有 Linux ext2/ext3 檔案系統、LUKS 加密分割區、GNOME、KDE 和 Mac OS X 等。

3. 系統架構與設計

本研究提供一套跨站冒名請求攻擊(Cross site request forgery)防禦機制，以網址路由(URL Routing)與 HTML5 Web Storage 為基礎，首先在使用者登入成功時於伺服端 Session 和客戶端 HTML5 Web Storage 儲存多組使用者專屬之 UUID API Key，這些 UUID API Key 主要目的為供使用者用於敏感性之連結如有關資料庫新增、修改、刪除、查詢等，使用者在後續的操作時將會將此些 UUID API Key 加入 API 網址並作為請求的網址中送出 HTTP 請求，傳送至伺服端後伺服端便可透過網址路由(URL Routing)之機制從路由值中取得使用者之 UUID API key 與所對應之 API，再透過 Cookie 機制取得使用者之 Session 做兩者 UUID API Key 之比對，若使用者 Session 之內容 UUID API Key 與使用者 HTTP 請求之 UUID API Key 相同，便做出正常的 HTTP 回應，反之若兩者不同則 HTTP 回應請求失敗，並於伺服端執行比對失敗時之機制如清除 Session 等。

3.1 客戶端登入機制

圖 6 為當使用者在登入成功後，伺服端會除了建立使用者所需的 Session 外還會建立使用者專用之 UUID API Key 並儲存於伺服端之 Session 與客戶端之 HTML5 Web Storage 中，以供客戶端在之後的 HTTP 請求中做為 UUID API Key 使用。

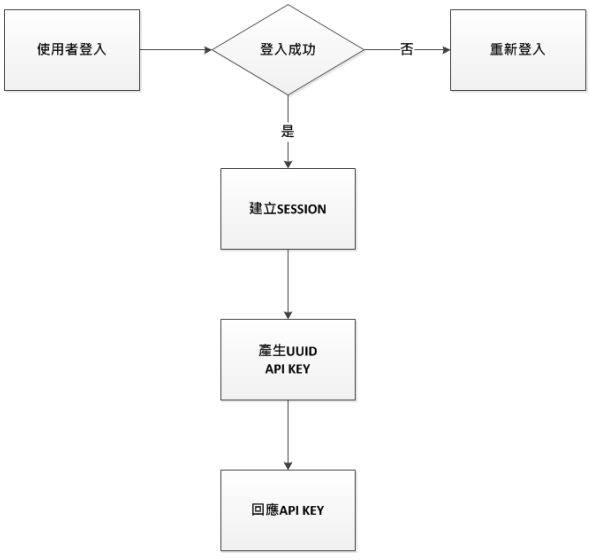

圖 6 客戶端登入流程

3.2 伺服端認證機制

圖 7 為當客戶端發出 HTTP 請求時，伺服端會除了使用 Cookie 驗證使用者的身分外，還會透過網址路由判斷客戶端 HTTP 請求所使用的 API 網址是否包含正確的 UUID API Key 以達到雙重認證的效果，以本流程圖為例，若當使用者的 UUID API Key 未通過伺服端認證時，便會清除該 Cookie 所屬之 Session，並導客戶端導向致重新登入之介面。

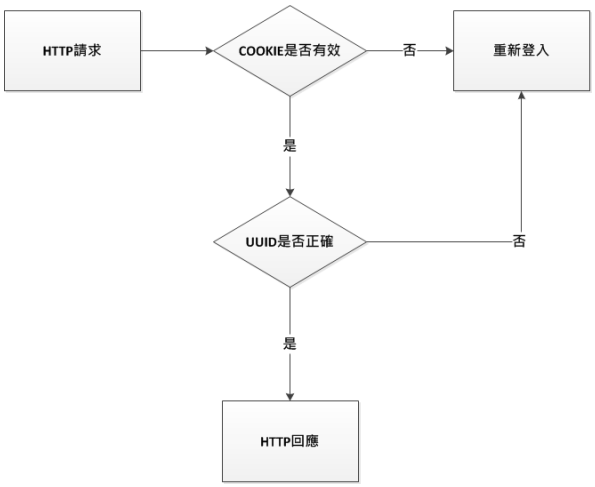

圖 7 伺服端認證流程

3.3 客戶端對伺服端請求機制

圖 8 為在本研究架構下，客戶端登入成功時伺服端除了以傳統 Cookie 認證之機制建立 Session 之外，也將創建符合敏感 API 數量之 UUID API Key 並儲存於伺服端 Session 與客戶端的 HTML5 Web Storage 中，接著在客戶端之後的 HTTP 請求裡，若有用到需要 UUID API Key 之 API 網址時，便會將該 API 網址與專屬之 UUID API Key 做結合做作請求之網址，當該請求被送至伺服端時，伺服端便會透過網址路由之機制，判斷該 UUID API Key 是否為該客戶端所屬，並根據認證之情形回應內容。

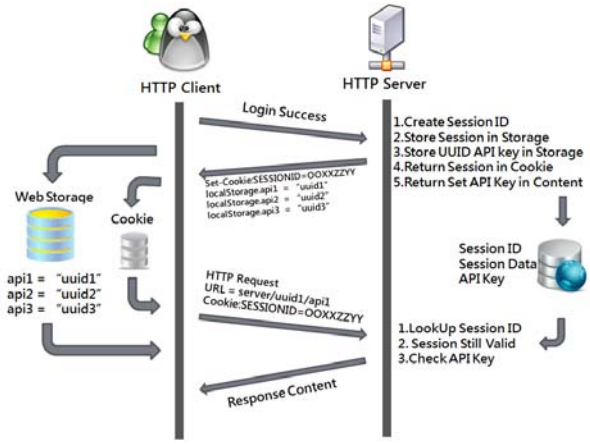

圖 8 HTTP 請求流程

3.4 跨站冒名請求攻擊之防禦

圖 9 為客戶端以正常之方式登入並驗證成功後，取得受認證之 Cookie 與多組 UUID API Key，接著在登入成功並取得認證後雖然遭遇到跨站冒名請求攻擊(Cross site request forgery)，但由於跨站冒名請求攻擊之攻擊者並無法事先得知客戶端之專屬 UUID API Key，而導致雖然 Cookie 認證通過了而 UUID API Key 認證卻失敗的結果，進而導致跨站冒名請求攻擊失敗。

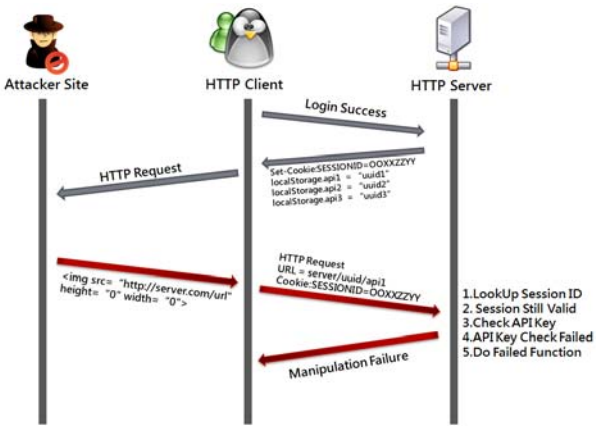

圖 9 跨站冒名請求防禦流程

4. 安全與效能評估

在安全與效能評估中，主要探討在本研究架構與舊有架構相比之下，對客戶端與伺服端所造成之效能影響與安全之改善。

4.1 效能評估

4.1.1 優點

由於 HTML5 Web Storage 不同於 Cookie 不會將資料附加在每一個 HTTP 請求中，而是只在被使用時才被截取並傳送至伺服端，因此相較於將資料存在 Cookie 上的方法對於伺服端與客戶端的流量都將有相當大幅度之改善，另外由於同時也減少伺服端之資料處理量。

4.1.2 缺點

1.由於本研究架構在伺服端加入了網址路由之機制(URL Routing)，因此對於伺服端來說增加了實體路徑搜尋與路由表對照之處理時間，但此時間消耗能透過對於網址路由機制的搜尋順序進行優化。而在本研究中，透過實驗比較正常實體路徑搜尋與網址路由(URL Routing)的機制，在同樣回傳空白網頁的情況下網址路由(URL Routing)機制大約

只比實體路徑搜尋多了 1m 秒的延遲。

2.本研究系統架構在使用者登入成功的當下，將會產生所有需認證之 API 所需的 UUID API Key，因此當採用本機制之 API 網址逐漸增加時，伺服端之運算量也會隨之增加。其圖 8 為本研究實驗在伺服端 CPU 為 Intel(R) Core(TM)2 Quad CPU Q9500 的條件下，當客戶端請求發生時，伺服端產生 UUID 從 0 到 50000 筆並回應客戶端所需之時間。

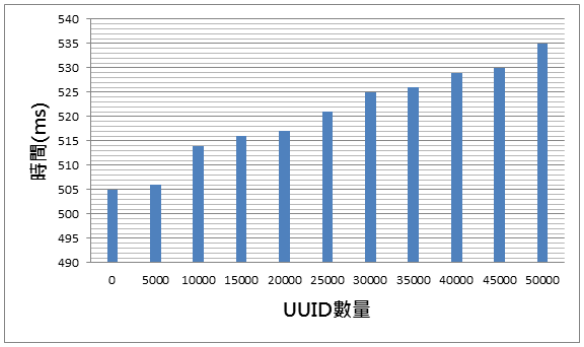

圖 8 UUID 數量與時間之影響

3. 本研究系統架構在使用者登入成功之後，會將所有產生之 UUID API Key 儲存一份在 Session 以供後續之認證與比對，因此當採用本機制之 API 網址逐漸增加時，伺服端 Session 所需之容量負擔也將會隨之成長。

4.2 安全性評估

4.2.1 優點

1.造成跨站冒名請求攻擊(Cross site request forgery)主要的原因為所有的網站使用者皆使用相同之 API 網址，因此導致駭客能在網頁中嵌入指定之 API 網址以進行特定之攻擊，而本研究系統架構則利用 HTML5 Web Storage 與網址路由企圖改善此本漏洞，期望能阻絕大部分之跨站冒名請求攻擊(Cross site request forgery)。

2.本研究系統架構遭駭客監聽時，由於每組 UUID API Key 是無關聯並且獨一無二的，因此若駭客想取得該使用者完整的控制，勢必需要投入更長的時間與心力。

4.2.2 缺點

由於 HTML5 Web Storage 可供所屬之 Domain 網站存取與控制，因此採用本研究架構之網站若擁有跨站腳本攻擊(Cross site scripting)等重大網頁漏洞，則本研究架構將極有可能遭到破解。

5. 結論

隨著網路技術的快速發展，網路安全的議題也勢必會變得更加重要，而在常見的網路攻擊上，跨站冒名請求攻擊(Cross site request forgery)相較於跨站本攻擊(Cross site Scripting)和隱碼注入攻擊(SQL Injection)，是網站開發者較為陌生或尚未注意的部分，故本研究預期在網路安全越來越重要的未來，跨站冒名請求攻擊(Cross site request forgery)必定會會到更高的重視，因此提出此系統架構。

本研究系統架構相較於其他相關研究的不同主要在於開發者實作的容易性上，大部分過去的相關研究大多都存在的開發成本與實作難度較高的問題，因此本研究從跨站冒名請求攻擊(Cross site request forgery)的原理著手，透過建立每位使用者的網址獨特性，提出了一個架構簡單的解決方案，試圖大幅減少客戶端遭受跨站冒名請求攻擊之機率，希望能為網路的安全盡一份心力。

參考文獻

[1] Siddiqui, M. S., & Verma, D. (2011, May). Cross site request forgery: A common web application weakness. In Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on (pp. 538-543). IEEE.

[2] Shahriar, H., & Zulkernine, M. (2010, November). Client-side detection of cross-site request forgery attacks. In Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on (pp. 358-367). IEEE.

[3] Open Web Application Security Project,accessed on May 2013 from https://www.owasp.org/https://www.owasp.org/index.php/Top_10_2013

[4] Kristol, D. M., & Montulli, L. (2000). HTTP state management mechanism.

[5] Moore, K., & Freed, N. (2000). Use of HTTP state management. RFC 2964, Network Working Group.

[6] Nescape. Persistent Client State: HTTP Cookies. http://wp.netscape.com/newsref/std/cookie\ _spec. html.

[7] Wagner, D., & Schneier, B. (1996, November). Analysis of the SSL 3.0 protocol. In The Second USENIX Workshop on Electronic Commerce Proceedings (pp. 29-40).

[8] Schreiber, T. (2004). Session riding-a widespread vulnerability in today's web applications. Whitepaper, SecureNet GmbH.

[9] Chris S. (2003). Foiling Cross-Site Attacks. PHP Architect Magazine.

[10] Jovanovic, N., Kirda, E., & Kruegel, C. (2006, August). Preventing cross site request forgery attacks. In Securecomm and Workshops, 2006 (pp. 1-10). IEEE.

[11] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (1999). Hypertext transfer protocol–HTTP/1.1.

[12] Johns, M., & Winter, J. (2006, May). RequestRodeo: Client side protection against session riding. In Proceedings of the OWASP Europe 2006 Conference.

[13] West, William, & Pulimood, S. Monisha. (2012). Analysis of privacy and security in HTML5 web storage. J. Comput. Sci. Coll., 27(3), 80-87.

[14] Sanderson, S. (2010). Pro Asp. net MVC 2 Framework. Apress.

[15] Leach, P. J., Mealling, M., & Salz, R. (2005). A universally unique identifier (uuid) urn namespace.