劫持 JavaScript 程式(下)

文章來源: HP-Fortify 提供　翻譯整理：叡揚資訊 資訊安全事業處

上期文章中介紹了 JavaScript Hijacking【劫持 JavaScript 程式(上)】，本次將繼續說明 JavaScript 的防禦方式：

3. 防禦 JavaScript Hijacking

第一代的網站應用程式是不易被 JavaScript Hijacking 攻擊的，因它們在傳輸資料時是用 HTML 文件的方式傳送，而不是單純只用 JavaScript 傳送。若應用程式無機密資料，對於 JavaScript Hijacking 也是相對的安全。

如果一個網站本就有跨網站指令碼攻擊的弱點(XSS)，那它就無法抵擋 JavaScript Hijacking 的攻擊。因為 XSS 本身會允許執行來自同一網域的指令碼。就算一個網站沒有 XSS 問題，也不代表它不會遭受 JavaScript Hijacking 的攻擊。

因 Web 2.0 會處理一些機密性的資料，以下是兩種用來防禦 JavaScript Hijacking 的基本方法：

(1) 拒絕惡意請求

(2) 避免直接執行 JavaScript 的回應

最好的防範方法就是同時採用上述兩種作法。

(1)拒絕惡意請求

站在伺服器的角度，JavaScript Hijacking 攻擊是非常類似 CSRF 攻擊的，因此若能擋掉 CSRF 攻擊也就能擋掉 JavaScript Hijacking 攻擊。

為了要偵測惡意的請求，每一個請求都必須包含一個很難讓攻擊者去猜測的參數。當伺服器收到請求之後，它就去檢查那個特定的參數是否和 session cookie 中的一致。惡意的程式碼並不能去存取 session cookie(因為同源政策)，所以攻擊者並不容易去偽造一個會通過檢查的請求。除了 session cookie 的方法外，也可以使用一個不同的暗碼。只要你這個暗碼是很難被猜測出來的，而它只會出現在正常的應用程式中，而不能被來自不同網域的應用程式使用，它就能防止攻擊者來偽造合法的請求。

一些架構是只在客戶端作用的。換言之，它們完全是用 JavaScript 來執行，對於伺服器程式如何的運作是一無

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 315 號 13 樓之 3　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

所知的。這指出它們並不知道 session cookie 的名稱。即使不知道 session cookie 的名稱，它仍然能在送請求到伺服器時，加入此類 cookie 的防禦方式。下列的 JavaScript 片斷，列出了"blind client"政策：

```
var httpRequest = new XMLHttpRequest();
...
var cookies="cookies="+escape(document.cookie);
http_request.open('POST', url, true);
httpRequest.send(cookies);
```

在伺服端也可以去檢查 HTTP referer 的表頭，來確認目前的請求是從正常的應用程式或是從惡意的程式而來。但就過往的經驗來說，HTTP referer 的表頭是不可靠的，所以我們不建議用這種方式。

還有一種防禦的作法是伺服端程式只回應 POST 的請求，而不回應 GET 的請求。因為很多使用<script>的攻擊都是使用 GET 請求而載入的。所以程式開發人員若為了安全性考量，那麼省略一些功能是必要的。

(2)避免直接執行回應

為了要讓一個惡意的網站去執行包含 JavaScript 的回應，合法的客戶端程式就可利用這點，在執行前先對其進行修改，因這些惡意程式必定包含了<script>的標籤。當伺服端在同步一個物件時，它必須包含一個字首(prefix)來讓它能夠藉由<script>來執行 JavaScript。一個合法的客戶端程式能在執行 JavaScript 將這些額外的資料移除。實作的方式有非常多種，我們將列出兩種：

字首：

```
while(1);
```

除非客戶端的程式能正確的移除這個字首，否則在評鑑這個 JavaScript 程式後，程式將會進入無窮迴圈。

```
var object;
var req = new XMLHttpRequest();
req.open("GET", "/object.json",true);
req.onreadystatechange = function () {
        if (req.readyState == 4) {
        var txt = req.responseText;
        if (txt.substr(0,9) == "while(1);") {
        txt = txt.substring(10);
        }
    object = eval("(" + txt + ")");
    req = null;
```


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 315 號 13 樓之 3 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

```
}
};
req.send(null);
```

第二，可在 JavaScript 的前後加入註解符號/* ... */，在它去呼叫 eval()之前必須先去移除這些註解符號才能被正確執行。

如下列的 JSON 物件：

```
/*
[{"fname":"Brian", "lname":"Chess", "phone":"6502135600",
"purchases":60000.00, "email":"brian@fortifysoftware.com" }
]
*/
```

客戶端的程式可用下面的方式移除：

```
var object;
var req = new XMLHttpRequest();
req.open("GET", "/object.json",true);
req.onreadystatechange = function () {
        if (req.readyState == 4) {
                var txt = req.responseText;
                if (txt.substr(0,2) == "/*") {
                txt = txt.substring(2, txt.length - 2);
                }
        object = eval("(" + txt + ")");
        req = null;
        }
};
req.send(null);
```

那麼惡意網站就不能再使用<script>來取得這些內含的資料

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 315 號 13 樓之 3 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