你的網站在裸奔嗎？一個 SQL Injection 實例的啟示

作者: 李明儒

SQL Injection(資料隱碼攻擊)問題早就不是什麼新聞，但前陣子在一個頗具知名度的活動網站上，赫然發現它大剌剌地現身! 原來，只要程式人員的一個"疏忽"(一個就夠了，真的!)，就足以讓任何網站的安全防線瞬間瓦解、門戶大開。本文便將以此實例探討 SQL Injection 所可能衍生的風險及因應對策。

若要給 SQL Injection 下個定義，它意指因爲程式人員對使用者輸入資料的驗證處理不當，致使有心人可透過輸入資料的機會，在輸入資料中夾帶 SQL 指令，以達到竊取資訊、篡改資料或入侵系統等惡意企圖。關於 SQL Injection 的細節過去已有相當多的討論，在此不多著墨。推薦幾篇不錯的文章，作爲實例探討前的背景知識基礎(請見參考資料 1、2)。

既然略去了 SQL Injection 的基本介紹，本文將以案例所曝露的缺失及如何改善作爲探討重點，希望這番老調重彈能再次突顯 SQL Injection 問題的嚴重性，也算爲整體資訊安全環境的提升進分心力。(自從見證了這個案例後，我已開始每天祈禱，希望往來銀行的所有資訊人員及委外廠商，都已完全了解 SQL Injection 的風險並隨時銘記在心。)

陰魂不散的 SQL Injection

很少有資安問題會像 SQL Injection 一樣，在發現多年之後，依然能三不五時來個奇襲，搞得一群人雞飛狗跳。若諸位有留意一些駭客入侵，竊取資料的新聞，就會發現原來它一直是駭客們的最佳拍檔，不曾絕跡。(例如: 大考中心的資安事件 http://mag.udn.com/mag/campus/storypage.jsp?f_ART_ID=11599)

近年來會鑽系統漏洞的病毒、蠕蟲誕生速度愈來愈快，以往安全性漏洞揭露後幾個月後才會冒出依附該漏洞的惡意程式，到現在幾天內就得面臨大軍壓境的危機。但是，只要確保軟體修補及防毒軟體更新的即時性，這些問題都可以被防範。然而 SQL Injection 問題最神妙之處在於「帶原者」是人而非程式，不限資料庫廠牌、程式語言、開發工具、作業平台，到處都有機會中獎。而我們無法以防火牆、防毒軟體的方式從外圍防堵，也不能比照軟體修補的方式，讓有 SQL Injection 問題的程式師在一夕之間開竅。尤其可怕的是，只要程式師還沒有建立正確的觀念，隨著換工作、接新專案，問題會呈現隨機式傳播。

如果 SQL Injection 的帶原者是人，那麼人與人之間會不會互相傳染? 答案是: 會的! 過去看過不少有 SQL Injection 問題的程式碼，找來程式作者詢問，答覆常常是: 程式碼是由參考某某人的寫法。如果沒有人提醒他其中隱藏的風險，我相信這個好用範例很快會用在其他類似需求的專案上，然後很不幸地被另一位程式菜鳥學去用。天哪! 又一個天真無邪的程式師變成了帶原者。

帶原者多半只因不了解 SQL Injection 的嚴重性，在知道真相的當下，震撼之餘，就能在瞬間產生免疫力。說到這裡，不禁回想起我接種疫苗的那一天… 當時，我還是個天真無邪的菜鳥程式師。在一個研討會中，講師介紹了 SQL Injection，並打趣地說，在竹科的場次，講完這一段，通常就有一批聽眾神色緊張地衝出會場，趕回公司改程式去了。當時，台下的我，雖然克制住打道回公司的衝動，但接連而來的是三天少吃少喝少睡的日子，將過去做過專案裡所有的 ASP、VB COM 都掃過一次，之後再找個理由通知客戶要更新程式，才算結束了這場驚魂。

在往後的工作經歷中，遇過許多程式人員，其中還還是有相當的比例尚未對 SQL Injection 建立正確的認識。

因此，在知名網站上看到 SQL Injection，讓人驚心，卻不意外。網站專案外包是目前業界的常態，而在價格取勝的制度下，低價得標的 SOHO 族、學生兼職族是否對"功能可以正常執行"以外的知識有所涉獵? 專案經理是否有相關的資安意識? 更不要提逐行對程式做 Code Review 是唯一可以確保系統不存在 SQL Injection 問題的方法。(這跟 Y2K 問題還真有幾分神似，只是當年有人發展出了半自動的 Y2K 問題掃瞄工具、也有人提供以行計費的 Code Review 服務，SQL Injection 問題則沒有這方面的選擇) 稍有不慎，就會給予 SQL Injection 可趁之機。而資安危機的共同特色是，一處漏洞與一百處漏洞的結局相同，系統機密終將被入侵者一覽無疑。

接著我們就將重點回歸到這個網站 SQL Injection 漏洞的實例探討上，看看問題是怎麼被發現的、背後的成因、可能導致的危險以及因應之道。而探討的方式，會將問題、成因、對策三者放在一起討論。

由於我們的重點在於 SQL Injection 問題本身，無意對該網站的資訊安全品質做任何批判，因此雖然所附圖例都擷取自真實的網站畫面，但我已隱藏(馬賽克囉!)與網站相關的文字資料，程式、參數名稱也多用化名，以免徒生不必要的困擾。

問題怎麼被發現的?

雖然我常以骨子裡流著駭客(Hacker，不是Cracker)的血液自傲，但也沒有閒到三不五時流連各大網站找碴，四處找漏洞鑽，會發現這個問題，純屬意外。

話說這個活動社群網站上有個功能，點選文章發表者上的連結，可以檢視其個人基本資料。它採取的是類似 UserInfo.asp?UserId=123 的做法，例如: 123 是會員的流水號。但是因程式沒寫好，UserId 未正確帶出，因此變成了 UserInfo.asp?UserId=。可以想見，ASP 抓到的 UserId 參數將是空值，而網頁則顯示了以下的錯誤訊息: (如圖 1)

圖 1 因 UserId 參數不當而顯示的錯誤訊息網頁

有 SQL Injection 經驗的老鳥看到這行訊息，就像鯊魚聞到血腥味一樣；單憑這則訊息就可以斷定網站存在 SQL Injection 問題，接下來便是大顯身手的時刻了。至此，我們發現了幾個問題:

【問題 1】
在產生查詢發表者個人資料的連結上，未確實代入使用者代號，造成傳入參數不正確，進而導致程式錯誤。
【原因】
程式人員疏忽，程式碼不正確，而測試人員也未抓出此一明顯錯誤。
【對策】
應落實專案管理要求，在驗收、上線前對系統各功能進行完整的測試。

【問題 2】

UserInfo.asp 程式未對參數做檢查，進而導致了資料庫存取的執行錯誤。

【原因】

UserInfo.asp 應先對輸入的 UserId 參數做檢查，當它不是數字或爲空值時，根本不需查詢資料庫。

【對策】

理論上，既然是要查詢某個人員的基本資料，就不該接受沒有指定 UserId 未指定的情境，應顯示參數不正確的例外。可在 UserInfo.asp 加入以下的參數檢查程式碼:

```
Dim strUserId
strUserId=Request("UserId")
If (strUserId="" Or Not IsNumeric(strUserId)) Then
    Response.Write "UserId 參數不正確!"
    Response.End
End If
```

【問題 3】

由錯誤訊息中，惡意使用者可以獲取許多珍貴的情報。圖 1 至少就揭露了以下幾條線索:

1. 程式使用「ODBC」方式連上「SQL Server」。
2. 程式透過共用的 ASP 程式(Connect.asp)連結資料庫。
3. 程式直接將 UserId 接在 SQL 指令中，且沒有外括單引號。當 UserId 爲空字串時，會組出"SELECT … FROM UserInfoTableName WHERE UserId="的不完整指令，形成"'='附近的語法不正確"錯誤。

由於參雜系統情報的錯誤訊息會顯示在使用者的瀏覽器上，後續當駭客要進一步探索更多的資料庫資訊時，也可透過顯示在網頁上的錯誤訊息傳回。

【原因】

程式未自訂錯誤訊息頁面，而將原始的錯誤訊息細節傳回給一般使用者。

【對策】

由於錯誤訊息中常會透露許多系統及程式的細節，在開發測試階段，這些資訊有助於 Debug；在系統上線後，卻變成駭客蒐集系統情報的重要管道。因此，在正式上線後，最好善用 IIS 設定客製化錯誤訊息或 ASP.NET web.config 中 CustomError 設定，對一般使用者隱藏錯誤的細節資訊。

看網站裸奔

由這個 SQL Injection 的發生位置來看，URL 中所輸入的 UserId 參數被用在使用者個人資訊的資料表查詢 SQL 中，而這個資料表最令人感興趣的欄位莫過於「密碼」了。於是我們在 URL 上調整一下，就可以把 UserInfo.asp 當成一個功能受限的 SQL Query Analyzer 使用；故意製造錯誤，再由錯誤訊息中獲得情報。參考資料 1 的文章介紹了透過加註 HAVING 1=1 的做法逐一查出欄位名稱，再使用 UNION 技巧將我們需要的資料僞裝成爲原本要查詢的欄位傳回。但我意外發現，由於 SQL 錯誤訊息的貼心設計，讓我們能更簡單地將密碼取回。

方法是這樣的，首先我們要先猜想密碼欄位名稱，一般不外乎 pass、passwd、或是 password。於是我們用 UserInfo.asp?UserId=1+AND+Password=1 讓組成的 SQL 變成"SELECT … FROM UserInfoTableName

WHERE UserId=1 AND Password=1”，其中故意夾帶一個錯誤。由於 Password 欄位多半會以 varchar 類別保存，當我們要求與數字 1 比對時，SQL Server 很聰明地提供了自動將 varchar 轉換成 int 的功能，若轉換失敗時還會貼心地會顯示字串內容解釋轉換失敗的理由。(如圖 2 所示) 哦哦! 使用者密碼也被貼心地夾帶在錯誤訊息中，交給了有心人。

圖 2 因爲轉型失敗，密碼欄位內容被顯示錯誤訊息中

循著這個資訊漏洞，配合會員清單網頁，有心人可以寫個程式試遍所有會員編號，爲會員清單補上密碼欄位。當網站系統最重要的機密被一覽無疑，跟脫光衣服在街上裸奔沒啥兩樣；更要命的是網站管理員、程式開發人員、企業主在此刻往往還渾然不覺。

【問題 4】

UserInfo.asp 將使用者輸入的資料未經檢核處理就附加成爲 SQL 指令的一部分，這就是大名鼎鼎的 SQL Injection(資料隱碼)漏洞。

【原因】

由錯誤訊息來看，UserInfo.asp 應是以 strSQL=”SELECT … FROM UserInfoTableNameWHERE UserId=”& Request(“UserId”)的方式直接組成 SQL 指令，讓入侵者連處理單引號的功夫都省了。

【對策】

在程式設計指南中，動態決定 SQL 指令的方法有兩種: 一種是以 Ad-Hoc 方式組合出 SQL 指令字串，另一種則是將動態變化部分以參數(Parameter)方式傳入，例如: ADO 及 ADO.NET 裡的 Command 物件均有 Parameters 集合屬性。一般會建議使用後者，理由是物件內建了字串的處理轉換功能，二則是資料庫不必每次重新解析、編譯 SQL 指令，對效能也有所助益。寫法如下: (以 ADO.NET 爲例)

SqlCommand cmd=new SqlCommand(“SELECT … FROM UserInfoTableNameWHERE UserId=@UserId”,cn);
cmd.Parameters.Add(“@UserId”,SqlDbType.VarChar).Value=Request[“UserId”];

而 Ad-Hoc 式的動態 SQL 指令字串並非完全不能用，在某些場合中，它的確是較便捷有效率的做法，只是在使用時，務必要加強對使用者輸入的資料的檢核。

以這個案例來看，由於 UserId 被設定是純數字，因此在 SQL 語法中不必使用單引號，但是由於 Query String 傳入的都是字串型態，它有可能是空字串，也可能是非數字，因此較嚴謹的做法是使用 VB 的 IsNumeric 或 Regular Expression 檢查該字串是否爲合法的數字。

若參數欄位的類別是(n)char、(n)varchar 或(n)text，則在組合 SQL 時，要記得用單引號將參數內容字串夾起來，並要留意參數內容中包含單引號時的特別處理，以免組出的語法不合規定或被植入惡意 SQL 指令。

一般來說，用以下的寫法將字串中的單引號置換成兩個單引號即可: (以 ASP 爲例)

strSQL=”SELECT … FROM YourTableName WHERE TheField=’” & Replace(Request(“UserInput”),””’,””””) & “’”

【問題 5】

使用者可透過查詢資料庫取得密碼內容。

【原因】

Password 欄位未使用加密或用雜湊值(Hash)保護，導致有心人可由資料庫中所儲存的欄位內容取得密碼。其中使用雜湊值又比加密法更安全一些，即便駭客已掌握雜湊演算法，也無法由欄位內容反推密碼。

【對策】

針對密碼欄位，可使用 DES、甚至 RSA 等加密演算法或是 SHA1、MD5 等常用的雜湊值取代密碼明碼儲存於資料庫中，如此可減少因資料庫內容外洩衍生的風險。

還會發生什麼事?

雖然見識到包含 SQL Injection 漏洞的知名網站讓人興奮，但我無意扮演駭客或藉著入侵系統獲利，在發了封 Mail 通知網站管理人員後，對該網站的「邪惡測試」就此打住。(當然，多少得壓抑滿腔的熱血與蠢蠢欲動的雙手) 然而，如果今天發現漏洞的是個有心人，後面還會有多少故事呢?

1. 如果 ASP 連線用的 SQL 帳號有 dbo 權限。
   入侵者可以蒐集到完整的資料庫情報(Table Schema)，列出所有的資料表，從中擷取更有價值的資料；或是 DELETE FROM TABLE、DROP TABLE，惡搞一番也很過癮。
2. 如果 ASP 連線用的 SQL 帳號是 sa。
   一旦取得 sa 的權限，可玩的花樣就更多了: 可以呼叫 SQL Server 的特殊 Stored Procedure，執行 DOS 指令、讀寫 Registry、將 Web Server 上的檔案打包傳出來，或是刪改網頁內容。再不然植入木馬程式，綁架伺服器作爲其他邪惡計劃的根據地亦相當刺激。

以上這些狀況，絕對是網管人員、程式人員與企業主的夢魘，而問題的源頭，只因爲區區一行程式沒寫好，再加上資料庫帳號的不當授權，當場萬劫不復! (在我的經驗中，用 sa 當成程式資料庫連線帳號的朋友還真不少!!)

2008 更新補充

最近見識過一些新的 SQL Injection 攻擊概念。傳統印象中，SQL Injection 要設法取得欄位名稱訊，以偷出資料或從事破壞爲樂。但是要進行這些操作，通常得仰賴網站傳回錯誤訊息的細節才能提供繼續深入的情報。近年來，很多網站預設都已開啓 Custom Error Page，讓手工操作入侵的難度變高，但並不代表開啓隱藏錯誤訊息就可以高枕無憂。不過，也開發現一些新的攻擊趨勢:

1. 駭客圈已流傳一些現成的 SQL 注入工具，裡面已針對 ORACLE、SQL、MySQL、Access 等各家資料庫寫好預設的多組測試 Script，不需要耐性過人也不必做苦工，交給工具快速試過一輪即可輕鬆得手。再配合 Google 找尋獵物，亂槍打鳥之下，就算你的網站沒什麼名氣都可能中鏢。
2. 除了有心駭客設法要破解網站盜取資訊，還有一種打遊擊式的 SQL Injection 攻擊，把全部的攻擊指令縮成一行 QueryString，四處亂試主機，成功就爽到，失敗了不過浪費幾百個 Bytes 的頻寬，是穩賺不賠的生意。而攻擊指令是假設資料庫的內容會被當成 HTML 顯示在網站上，所以只要找出 SQL 資料庫中所有的 varchar, nvarchar, ntext, text 欄位，在後方加上一段<script src="用來載入木馬的 js 檔案">，就可以將網站當成感染源，達到廣種木馬的目標。(這類木馬的原理可以參考

http://blog.darkthread.net/blogs/darkthreadtw/archive/2007/06/01/816.aspx，通常只要勤做 Windows Update 並避免執行來路不明的程式就可避免)

結論

在 SQL Injection 問題開始被重視的數年後，依然能在實務網站上與它不期而遇，興奮之餘也有幾份擔憂，不禁想問，Internet 上還有多少網站是不安全的? 會不會有我的身家資料就存放其中? 但願這篇文章多少能突顯出它的威脅性及潛在風險，有助於減少類似的情境再度上演。

最後，針對 SQL Injection 的防範，整理幾則設計網站時的注意事項:

1. 在正式環境中，善用客製化錯誤訊息的功能，避免將錯誤訊息的細節直接傳送給使用者，以免系統架構情報外洩。
2. 千萬不要信任使用者所輸入的任何資料，每一個輸入機會都可能爲駭客所用，成爲傳入惡意指令的管道。如果要將其融合爲指令的一部分，必須在格式驗證上加倍小心。
3. 減少組裝 SQL 指令字串(Ad-Hoc SQL)的程式設計方式，以減少發生 SQL Injection 的風險。改用 Parameter 方式傳入動態參數，還兼有提升效能的好處。
4. 在資料庫帳號的應用管理上，避免過度授權，尤其 sa 帳號實在沒理由當成網站的連線身分，授與連線帳號的權限要愈少愈好，以勉強能維持正常作業爲最佳。

參考資料

參考資料 1 - SQL Injection (資料隱碼)– 駭客的 SQL 填空遊戲
http://www.microsoft.com/taiwan/sql/SQL_Injection_G1.htm
參考資料 2 - 『資料隱碼』SQL Injection 的因應與防範之道
http://www.microsoft.com/taiwan/sql/SQL_Injection.htm