線上的滲透測試網站

下面介紹的幾個網站是廠商公布的測試環境，在閱讀本書時，可以利用這些網站做為各章節的練習標的。

testfire

網站 URL：http://demo.testfire.net

由 IBM 提供的測試環境，做為測試自家的 AppScan 弱點掃描工具測試之用，此網站幾乎每年都會改版，後續章節將以此網站做為滲透測試的參考對象，用以展示相關工具的使用方式。

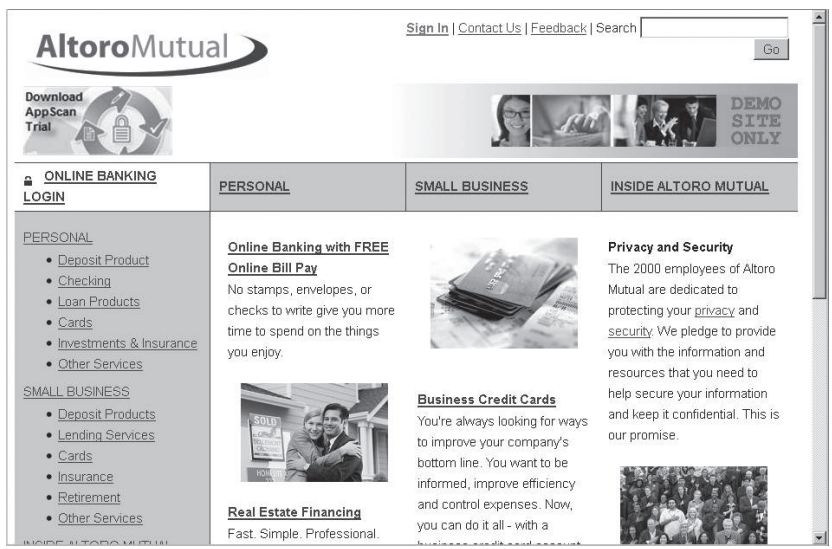

圖 3-1：demo.testfire.net

若想瞭解 AppScan，可以到下列網址下載試用版：

https://www.ibm.com/tw-zh/security/application-security/appscan

但試用版只能用來掃描 testfire 網站。

webappsecurity

網站 URL：http://zero.webappsecurity.com

相似於 testfire，zeroBank 是 Fortify WebInspect 弱點掃描工具的測試評估標的。Fortify 公司成立於 2003 年，主要產品有白箱（Fortify SCA）及黑箱（WebInspect）掃描工具。該公司於 2010 年被 HP 併購，歸入 HP（惠普）的企業產品部門，SCA 與 WebInspect 就變成了 HP 的產品，名稱仍維持 Fortify SCA 與 Fortify WebInspect，而在國內則常將 Fortify SCA 稱為 Fortify 白箱，Fortify WebInspect 叫作 HP WebInspect。

2015 年 HP 將企業產品部門獨立出來，成立 HPE（惠普企業）公司，SCA 與 WebInspect 隨之冠上 HPE 商標。而 2016 年，HPE 將軟體資產事業（包括 Fortify）與 Micro Focus 合併成立獨立的 Micro Focus 公司，因此，Fortify 相關產品也移轉到 Micro Focus。

所以，當讀者看到 Fortify WebInspect、HP WebInspect、HPE WebInspect 或 Micro Focus WebInspect，其實是同一套工具。

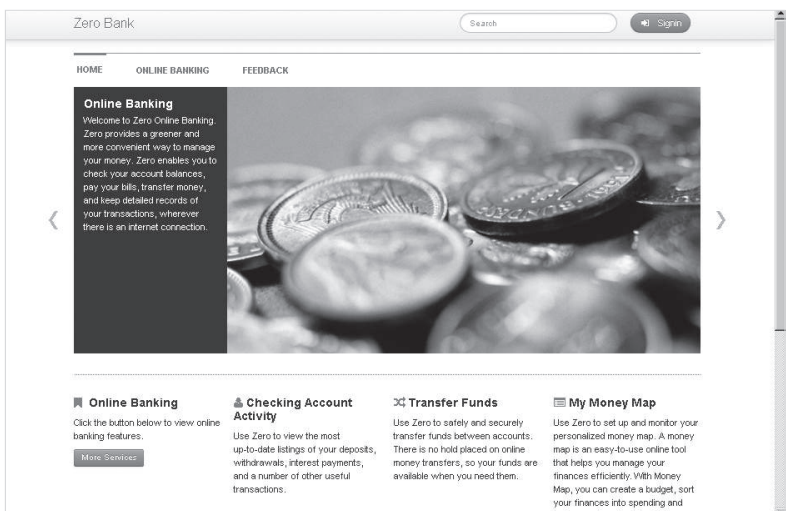

圖 3-2：zero.webappsecurity.com

和新版 WebGoat 不同，5.4 版的帳號和密碼皆預設為「guest」，登入成功後，即可看到 WebGoat 測驗網站了（圖 3-9），按下「Start WebGoat」鈕即可挑戰 WebGoat 的各個關卡！

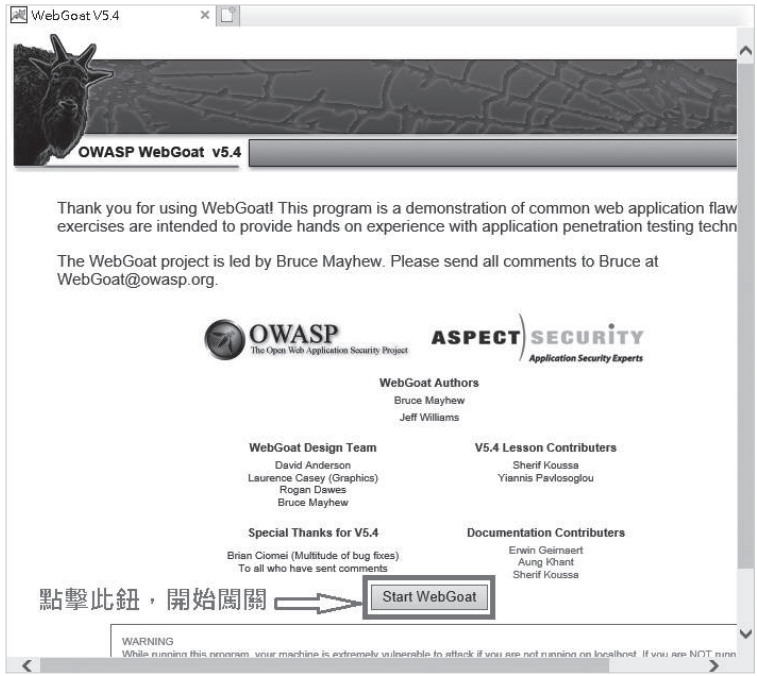

圖 3-9：開始 WebGoat 闖關

4. 如何開始

進入 WebGoat 挑戰頁面，第一頁是作業說明，這裡會有每項功能的詳細介紹，要挑戰的關卡羅列在左邊的清單中，題目前有圓形綠底打勾者，表示已經完成（過關），仔細看 Introduction 下的關卡只要點擊即可過關。如果要將已過關狀態回復成初始狀態，可以選擇該項關卡後，再點按右上角的「Restart this Lesson」（圖 3-10）。

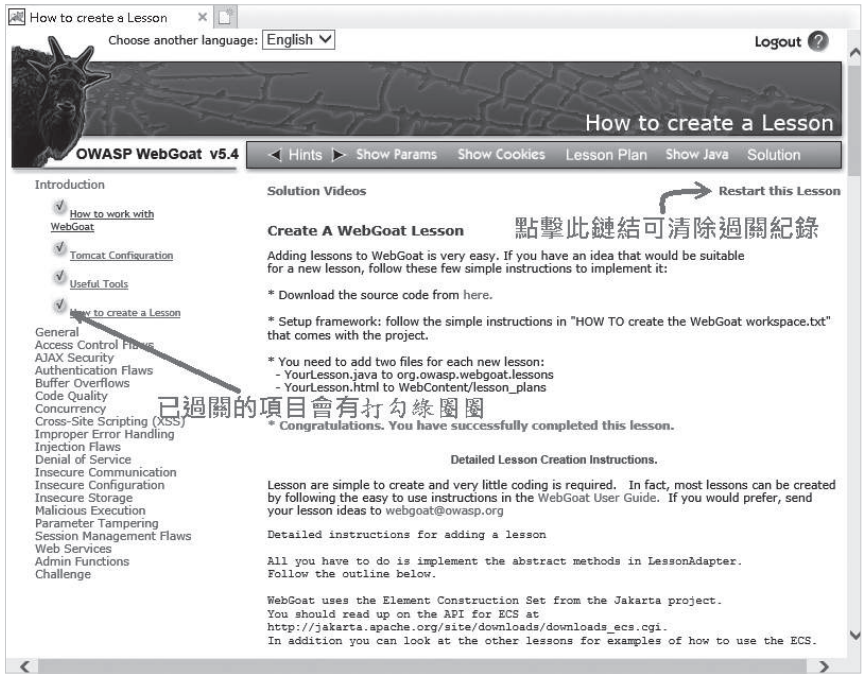

圖 3-10：仔細閱讀 WebGoat 說明

由於 WebGoat 當初設計時，OWASP 正在推展 WebScarab 工具，在 General 關卡的第一題「Http Basics」過關條件即要求使用者利用 WebScarab 攔截往來的 Request ／ Response 資料。

至於 WebGoat 設計的關卡有些甚至不算弱點，用不著每一題都做。筆者建議不必嘗試用人工解題，除非讀者想要印證自己的觀念、理解程度或純粹為了學習，不然，應該看完本書後，利用書中介紹的工具解題，這樣才能熟練、有效地完成滲透測試作業。

5. 設定團隊連線

WebGoat 5.4 預設只能透過本機連線，如果想團隊合作練習，就必須開放從別的電腦連線，因此需要手動修改設定，在 WebGoat 目錄下可發現 tomcat\conf 資料夾，裡頭有三支 XML 檔案，分別為：server.xml、

圖 3-36：將資料夾轉換為應用程式，並選用 .NET v2.0 集區

3. 點擊「轉換成應用程式」後會出現「新增應用程式」對話框，此處要變更**應用程式集區**（圖 3-36），請將它改成「.NET v2.0」以符合 HacmeBank 執行所需的環境版本。

備註

若轉換應用程式時忘了選擇正確的**應用程式集區**，事後仍可透過應用程式的**基本設定**去修改。

4. 利用檔案總管賦予 IIS_IUSRS 帳號對「C:\inetpub\wwwroot\HacmeBank_v2_WS\App_Data」資料夾擁有**完全控制**的權限，否則會出現資料庫無法存取的錯誤（作業系統錯誤 5: 存取被拒）。請在 App_Data 資料夾點擊滑鼠右鍵，從彈出選單選擇「內容」，再切換到對話視窗的**安全性**頁籤。點擊**群組或使用者名稱**清單右下的「編輯」鈕開啟權限編輯對話框（圖 3-37），選擇「IIS_IUSRS」並勾選下方「完全控制」項。

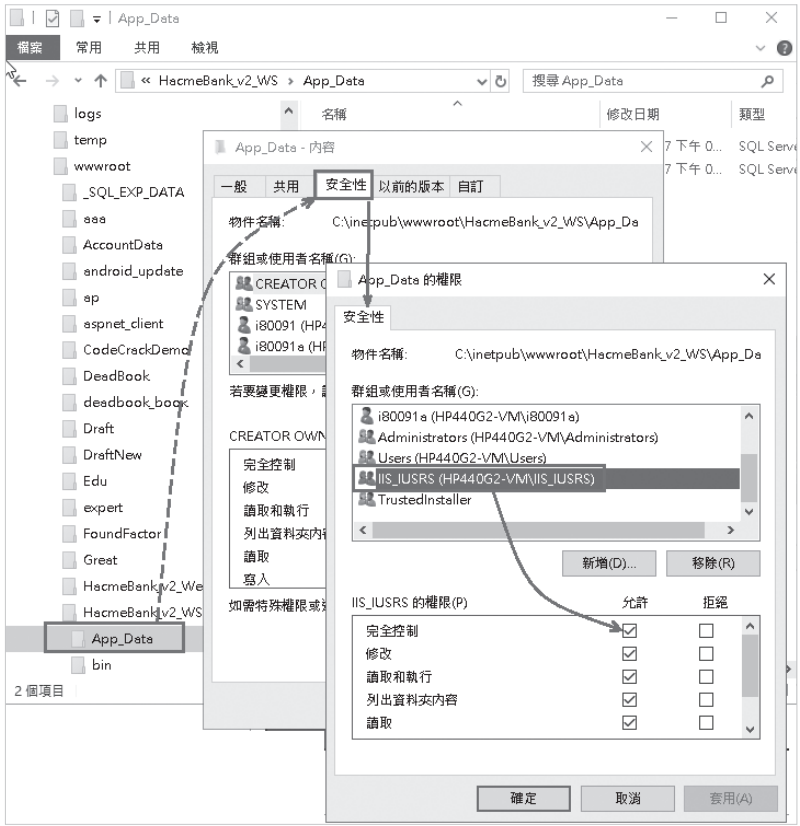

圖 3-37：賦予 IIS_IUSRS 對 App_Data 資料夾具有完全控制權限

5. 修改資料庫連線字串

當完成上述各項設定後，懷著愉悅的心情開啟「http://localhost/HacmeBank_v2_Website」，終於看到 Hacme Bank 2.0 首頁，但嘗試登入時卻出現連接 SQL Server 的錯誤訊息（圖 3-38）。

圖 3-40：因日期的中文上、下午造成應用程式錯誤

只要修正 HacmeBank_v2_WS 的 Web.config 檔（第 5 點提到）第 74 列，增加「culture="en-US"」屬性，即可解決日期字串問題，修正後如下所示：

```
<globalization requestEncoding="utf-8" responseEncoding="utf-8" culture="en-US" />
```

關於 HacmeBank 的一些資訊

可喜可賀！ HacmeBank 終於架設完成，它是以教學為目的弱點網站，裡頭已事先建立三組測試帳號（如下表的三位銀行客戶），每組帳號都有兩個銀行帳戶，讀者可以玩玩彼此轉帳或貸款的遊戲，但這不是我們的目的，我們的目標是要取得管理員權限，也就是進入「Admin Section」後台管理功能。

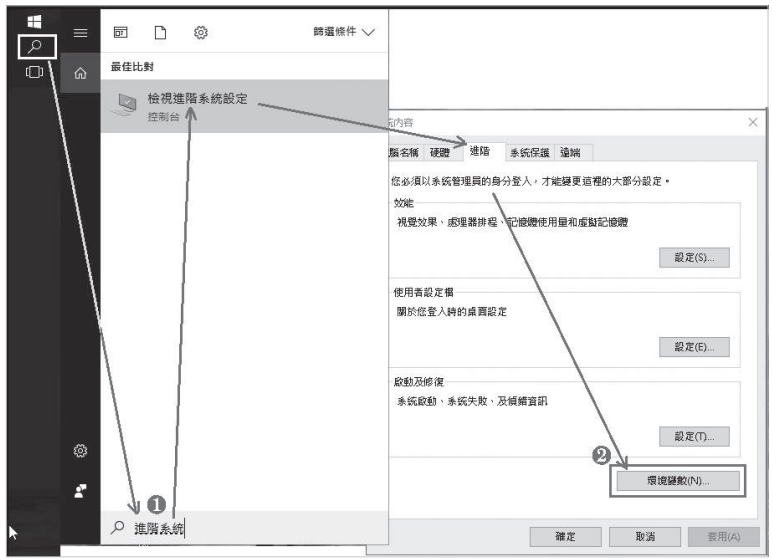

圖 3-43：開啟進階系統設定，以便修改環境變數

2. 透過右下的「環境變數 (N)...」鈕開啟環境變數設定對話框。

3. 找到使用者變數或系統變數的 PATH 項，然後點擊「編輯」鈕。

4. 編輯或新增 PATH 環境變數的內容。（圖 3-44）。

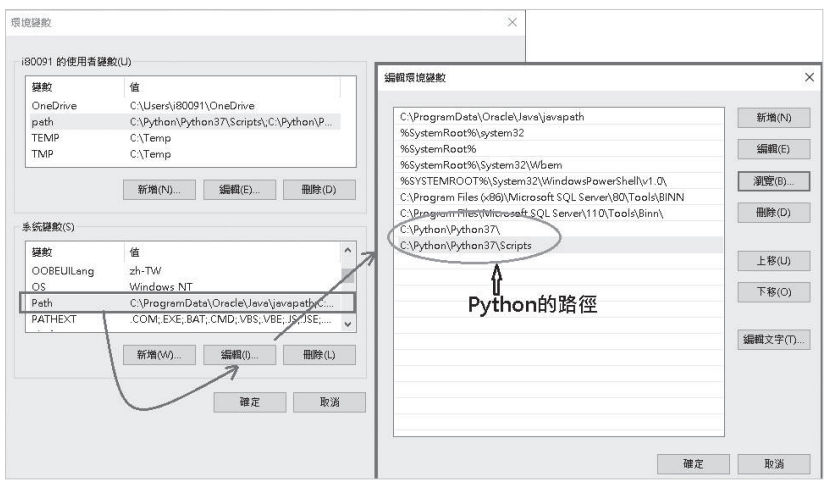

圖 3-44：編輯或新增 Path 環境變數

容，圖 4-2 即利用 Firefox 的**開發者工具箱**觀察瀏覽器和伺服器的互動過程，從左方窗格可看到一張網頁會發起許多組請求，每組請求都是一個獨立的 TCP/IP 連線，由此亦可看出一張網頁是由許多獨立的資源組合而成。瀏覽器藉由**請求標頭**（Request Header）告知伺服器請求內容，伺服器則利用**回應標頭**（Response Header）告知請求是否成功及回應內容的相關屬性。

圖 4-2：利用開發者工具箱觀察網頁的請求與回應

當使用 GET 方法請求資源時，發送給伺服器的參數（資料欄位）直接串在網址的後面（用？隔開），每個欄位是以「名稱 = 值」的形式表示，欄位與欄位之間則用 & 分隔。若採取 POST 方法請求資源，欄位資料仍然是以 & 分隔的「名稱 = 值」形式表示，但會尾隨在請求標頭的最後一列，在標頭資料與欄位資料之間隔著一列空白。圖 4-3 是 GET 請求的標頭內容，圖 4-4 是 POST 請求的內容。

但是如果輸入的資料改成：uid="' *OR 1=1*- "；pwd="a1234567"，結果 sSql 就變成

```
sSql="SELECT * FROM users WHERE id='' OR 1=1-- ' AND password='a1234567' "
```

讀者是否看出問題了，sSql 字串中的「--」會將後接的文字變成註解，而「OR 1=1」表示條件一律成立，上面的式子就變成跟下面的查詢語法等效，限制條件被註銷了。

```
sSql="SELECT * FROM users"
```

同事曾問我「為什麼知道查詢的語法是「"SELECT * FROM users WHERE id='" + uid + "' AND password='" + pwd + "' "」，當然不知道，從網頁是無法看到後端程式的語法，這條式子是「臆測」出來的，記得第一章提過滲透測試必需要有創意，就是要揣摹設計者的心思。如果網頁欄位有被 SQL Injection 的可能，我們就會假設幾種後端程式可能的寫法，再針對假設的寫法進行驗證。

從上面的例子可以看出駭客精心安排輸入的字串就能改變 SQL 指令的內容，SQL Injection 不只是可以繞過身分驗證，還可以撈取整個資料庫、刪除資料紀錄或整個資料表，甚至可能操控後端伺服器做任何事。

防護建議

防範注入攻擊的手段不外乎驗證及過濾使用者提交的資料內容，但千萬要注意，要在伺服器端執行驗證及過濾，而不是靠瀏覽器端（使用者端）的腳本程式，因為使用者能輕易操控瀏覽器端的機制。

以 SQL Injection 而言，後端程式進行 SQL 查詢時，最好使用 Prepare Command（也有人稱為 Parameters Command），以 C# 為例，將字串拼接方式改成：

3. 使用固定或規則性產生的 SessionId，駭客輕易「猜到」目前使用者的 SessionId 而假冒其身分。

4. 未使用加密機制傳輸資料，簡單說就是網站（頁）沒有啟用 SSL，機敏資料可能被截聽，駭客可以從中取得合法使用者的身分及權限。

5. 密碼資料未以加密格式儲存，駭客若取得帳戶資料，即可用各種身份操作系統。

6. 身分驗證機制設計不良，例如沒有限制登入失敗次數、過於詳細的錯誤提示，甚至允許使用弱密碼等，以致駭客能夠利用暴力破解方式取得使用者的帳號及密碼。

防護建議

1. 網站應該啟用 SSL 機制及加密儲存機敏資料，這也是解決 Sensitive Data Exposure（A3） 及 Security Misconfiguration（A6）缺失的必要措施。
2. 網頁傳送資料儘量以 Post 取代 Get，並在每個頁面加上隨機產生的驗證符記（token），以確保網頁來源的合理性。
3. 網頁標頭（header）要設定 x-frame-options 或 content-security-policy 欄位，以防止網頁被 iframe 挾持。
4. 設定使用者一段合理時間（如 15 分鐘）未向伺服器提出請求時，自動清除 Cookie 及 Session 狀態，並在使用者斷線後，立即將存活時間設為過期。
5. 用來維護連線狀態的 SessionID 應該以隨機方式產生，並在每次請求時重新動態更換，降低駭客猜中 SessionID 的機會或者藉用舊 SessionID 偽冒合法使用者的身分。

IE 也有儲存密碼的功能，但無法直接由瀏覽器的功能觀看已保存的帳號及密碼，如果是 Windows 10，可以利用控制台的**認證管理員**查看 IE 或 Edge 所保存的網站帳號及密碼（圖 4-15），Windows 7 沒有網站認證選項，但可以從網路上找到許多工具（如 WebBrowserPassView）來窺視密碼，因為需要在使用者的電腦上安裝或執行額外工具程式，這已非本書討論範圍。

圖 4-15：利用認證管理員查看 IE 保存的網站登入帳號及密碼

從上面的說明，可知瀏覽器雖然記住密碼，但想看到明文密碼還要取得本機使用者的密碼，這裡介紹一種修改網頁內容讓密碼欄位顯示明碼的手法。網頁上的密碼欄其實是型態為 password 的輸入框，只要動點手腳將 password 刪除或改成 text，密碼立刻變明碼。

使用者若啟用記住密碼功能，我們可以在登入頁的帳號欄利用**向下鍵**快速輸入已保存的帳號，此時瀏覽器會自動帶出對應的密碼，但畫面上密碼是被遮隱的，此時，利用開發人員工具將密碼欄的 type 內容清空或改成 text，遮隱狀態就會變成明碼文字狀態，如圖 4-16 所示。